# Software Radio Systems Engineering for URLLC Industrie 4.0 Applications

Dissertation

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt dem Fachbereich 1 (Physik/Elektrotechnik)

der Universität Bremen

von

Johannes Demel M.Sc.

| | |
|---|---|
| Eingereicht am: | 6. Juni 2023 |
| Tag des öffentlichen Kolloquiums: | 20. Juli 2023 |
| Gutachter der Dissertation: | Prof. Dr.-Ing. Armin Dekorsy |
| | Prof. Dr.-Ing. Peter Rost |
| Weitere Prüfer: | Prof. Dr.-Ing. Karl-Ludwig Krieger |
| | Prof. Dr.-Ing. Steffen Paul |

University of Bremen

Bremen, 25. Juli 2023

# Preface

The presented dissertation emerged from my work as a researcher with the Department of Communications Engineering at the University of Bremen. First and foremost I would like to thank Armin Dekorsy for offering me the opportunity to pursue my doctoral degree at the Department of Communications Engineering. His continuous support and professional guidance enabled and encouraged me to successfully publish my results in numerous international conferences as well as to discuss and exchange ideas among peers to grow my scientific expertise as well as to grow personally.

Furthermore, I would like to extend my gratitude to Peter Rost for taking the time to assess this dissertation and for his valuable remarks. The same goes for Steffen Paul and Karl-Ludwig Krieger for spending their time to read through this work and partaking in the colloquium as further examiners. I would like to thank Friedrich K. Jondral for encouraging and supporting me to start this journey to pursue my doctorate. I would further like to thank the German Federal Ministry of Education and Research (BMBF) for funding a large portion of this work as part of the projects HiFlecs, TACNET 4.0, and IRLG.

I am indebted to Carsten Bockelmann for his guidance, broad wisdom and skillful support during these years that made this thesis possible. The countless discussions as well as support and time that enabled me to bring new ideas from inception to publication were immeasurably valuable. My coworkers deserve praise for making this time with the ANT as fruitful and enjoyable as anyone can only wish for.

This multi year effort would not be possible without my friends and family who were always there for me. I am deeply grateful for their support. Finally, I would like to express my deepest gratitude to my girlfriend Steffi who supported me in every imaginable way to follow through with my work.

Bremen, 2023
Johannes Demel

# Contents

# Chapter 1

# Introduction - Industrial radio systems engineering

Communication systems evolve rapidly to cater to new use cases with new requirements. With the advent of digital communication, we observed an exponential growth in possible data rates for human-centered communication needs. Nowadays society is undergoing a fundamental transformation: digitization. We, the people, require higher data rates to watch high resolution videos, transfer large files, be in video chats with lots of people simultaneously, and more recently stream video games. All these use cases can handle latency and occasional outages while being able to resume operation immediately after an error. In most cases such errors are not even perceived by users. Thus, current systems are designed for these enhanced Mobile Broadband (eMBB) use cases that trade in low latency and high reliability for higher overall throughput.

Over the course of the last years, Internet of Things (IoT) applications have become more and more popular. For private consumers, IoT applications include novel smart home appliances and health care utilities. According to some estimates, many more IoT devices than human-operated devices are online nowadays. This massive Machine Type Communication (mMTC) device class exhibits different communication behavior with short packets, low energy consumption, sporadic communication, low reliability, and high error tolerance. Such devices are mostly used for monitoring, such as gathering environmental data or maintaining situational awareness.

A third class of applications is gaining momentum now; the class of mission-critical applications that include autonomous driving, Factory Automation (FA), and Motion Control (MC), is becoming more and more popular [Fet14, AZB+21]. These mission-critical Industry 4.0 (I4.0) wireless communication systems require support for short packet communication with real-time deadlines. Furthermore, they require high reliability with exceptionally high resilience to burst errors, i.e. Consecutive Erroneous Packets (CEPs). Finally, these industrial applications require extremely low latency. These requirements compete with each other in the Ultra Reliable Low Latency Communication (URLLC) use case class.

During the 5th Generation New Radio (5G NR) standardization process, it was recognized that current standards, e.g. Long Term Evolution (LTE) and IEEE 802.11 (Wi-Fi), solely focus on eMBB use cases, while mMTC and URLLC use cases are not addressed sufficiently. Thus, the 5G NR standard includes three use case profiles: eMBB for human-centered high data rate applications, mMTC for IoT devices, and URLLC for mission-critical I4.0 applications [OMM16, DPS18]. We focus on URLLC in our work.

Besides the rise of new application classes with their specific requirements, we observe a trend towards softwarization. To this end, softwarization is a key component to enable Cloud Radio Access Network (Cloud RAN) deployments [CDG+21]. Cloud RAN configurations allow components to be flexibly combined and extended without the need to re-deploy an entirely new system. After an initial Cloud RAN deployment, these systems may receive gradual improvements to better fit the current use case. Furthermore, Cloud RAN deployments provide the flexibility to be adapted individually at every site and Cloud RANs can be reconfigured to the farthest possible extend whenever the requirements change. However, before we discuss technological trends, we first consider industrial application requirements.

## 1.1 Industrial radio requirements

The origins of this work to identify industrial radio requirements can be found in projects Innovative Wireless Technologies for Industrial Automation (HiFlecs) and TACtile interNET 4.0 (TACNET 4.0). Project HiFlecs targeted novel industrial wireless communication systems to meet the requirements of I4.0 Closed-Loop-Control (CLC) applications [BDG+17]. The efforts to identify and refine the requirements for I4.0 wireless communication were picked up in project TACNET 4.0 along with further research towards suitable solutions [GSS+18, GSS+21]. These research efforts heavily impacted 5G NR standardization and requirements identification [3GP19a, 3GP20]. In our work, we focus on PHYsical layer (PHY) and Medium-Access-Control (MAC) technologies for wireless communication systems to improve URLLC systems.

The authors in [DMW+17] built a connection between the Key Performance Indicators (KPIs) in the automation industry, Mean Time To Failure (MTTF), and in the communications industry, Frame-Error-Rate (FER). However, this results in FER requirements that are extremely low and require enormous effort to fulfill. Meanwhile the issue to minimize CEPs, or burst errors, is not addressed directly. In [DBD20], we proposed to shift the KPI to burst error minimization to directly address application requirements. Thus, exceeding the maximum number of CEPs on a single link leads to an emergency halt of the whole system. In particular, mission-critical systems that require ultra reliable communication and suffer catastrophically from burst errors are the focus of our work [3GP20].

The industrial wireless radio communication system requirements, low latency and high reliability, and the small packet constraint with real-time deadlines present contradicting targets for wireless communication systems. Furthermore, these industrial applications expose periodic deterministic communication behavior with short packets and real-time deadlines [3GP19a]. Moreover, smaller packets make it more difficult to satisfy reliability requirements. Lower latency requirements make MAC layer re-transmission mechanisms more challenging or even impossible. Real-time deadlines and latency requirements of less than 1 ms are expected, while communication periodicity is on the same time scale. Thus, we expect one-shot communication systems to be the norm in these scenarios.

Industrial radio measurement campaigns [DHC$^+$19, MBF$^+$05] show that all channels follow a Rayleigh fading model. Distributed Access Points (APs) and antennas improve spatial diversity in an Ultra Dense Network (UDN) and thus, robustness against deep fading events on a single link in these particularly challenging channels. Likewise, a Cloud RAN architecture enables distributed APs to boost reliability by observing the same signal through different channels and joint processing [WRB$^+$14, RBM$^+$15].

Throughout this work, we investigate communication system components and their impact on latency and reliability with respect to URLLC requirements in a Cloud RAN implementation where functional parts are either distributed or centralized.

## 1.2 Softwarization for Cloud RANs

Traditional Radio Access Networks (RANs) are specialized hardware monoliths. Together with ever increasing system complexity, these monoliths tend to be a hindrance to progress [Ray03]. At this point, only the largest companies are able to develop such monoliths. Future RAN deployments are expected to become more flexible to adopt new technologies more rapidly. Commercial-Of-The-Shelf (COTS) hardware, such as General Purpose Processors (GPPs), offer the opportunity to innovate again. Generally, higher RAN layers are more prone to softwarization [AVK19, MMM19]. However, the need for RAN softwarization has been recognized [Lem17, DKP$^+$17] and thus, the demand for software RAN solutions is growing [LA20]. Further, the O-RAN Alliance (O-RAN) is an industry organization that drives RAN softwarization to be able to deploy open systems [O-R21].

A full software implementation offers great flexibility [Gra13]. In our work, we consider a Cloud RAN under software design principles. Previously, hardware design principles, with Field Programmable Gate Arrays (FPGAs) or even Application Specific Integrated Circuits (ASICs), would assume control and thus make it much more difficult to re-organize and leverage flexibility. Hardware development, even if not ASIC focused, is inherently difficult and cumbersome. The author in [Gra13] makes a case for FPGA development in case requirements cannot be otherwise met but recommends to stick to software development. However, specific functionality, e.g. Forward Error Correction (FEC), may be provided by specialized accelerators such as Graphics Processing Units (GPUs) or FPGAs that have to be seamlessly integrated into the software implementation via an Application Programming Interface (API) without interrupting the software system design.

In the wireless communication realm, the paradigm to use COTS hardware to build flexible communication systems is called Software Radio (SR) and was first coined by Joseph Mitola [Mit92]. The concept describes how to use wideband transmitters and receivers and how to carry out all operations in software to gain unprecedented flexibility. It enables myriads of new applications and use cases; however, it is very resource intensive and requires extremely sensitive hardware and thus is often infeasible in practice. Since signal processing can be carried out in the equivalent baseband, we consider a hardware frontend that strikes a balance between flexibility and practicability, the Software-Defined Radio (SDR) concept. A Software-Defined Radio (SDR) frontend is a transceiver for arbitrary complex samples and allows for the control of physical signal properties dynamically through a standardized interface. A Cloud RAN is built on COTS hardware with SDRs.

The principles of agile software development are being introduced alongside the trend towards more flexible software RANs on COTS hardware [CB14, DPL15]. Thus, the advantages of a Cloud Radio Access Network (Cloud RAN), built on cloud computing, are being recognized with appropriate development techniques, thorough understanding for implementation detail, and deployment benefits and procedures [MG11, Lem17, DKP$^+$17, OBJ13, CDG$^+$21]. This flexibility allows us to gradually upgrade systems, evaluate different module combinations, avoid constant hardware changes, and find the best feature set for the use case at hand.

Often, software development for Cloud RAN is carried out with several programming languages including C and C++ for performance critical code as well as Python for support functionality and testing [WO20]. These programming languages may provide a solid base for rapid application development if they are well integrated into a modern, agile workflow that follows State-of-the-Art (SotA) programming idioms and paradigms. One should never underestimate the burden of legacy programming languages with outdated programming paradigms and missing features, such as an appropriate module import mechanism. Further, future development may benefit from more recent programming languages such as Rust.

Previous works in the field of Cloud RAN include ones such as [GM13]. Nowadays, there are two open-source C/C++ LTE SDR implementations known to the authors: srsRAN, formerly srsLTE, and OpenAirInterface (OAI), that have both shifted their focus towards 5G NR lately [GMGSS+16, KSKW19]. These implementations target the LTE and 5G NR standards, and thus, they are specifically designed to optimally accommodate the inherent LTE and 5G NR structure. In contrast, the GNU Radio (GNU Radio) provides the framework to build a wide range of SDR applications [LMA+22]. Specifically, it provides tooling for infrastructure, e.g. multi-thread management, visualization, and hardware interface abstraction among other features, and thus GNU Radio is not tied to any specific communication standard. Additionally, sophisticated methods to orchestrate different parts of a RAN such as application configuration, deployments and update management are out of the scope of this work since we focus on Digital Signal Processing (DSP) aspects [DKP+17].

## 1.3 Objective of this thesis

This thesis aims to address the following research objectives. We investigate which and how SotA technologies in wireless communication may be used in a system that caters to URLLC requirements. These investigations include the identification of suitable technologies for reliable short packet communication as well as latency considerations in software implementations. Beyond that, trade-offs between latency and reliability are discussed. Furthermore, we investigate how a software implementation may enable Cloud RAN deployments.

A Cloud RAN allows for a flexible RAN structure with distributed APs in a UDN that we discuss to improve reliability. Here, we investigate how we can leverage the proposed Cloud RAN structure with improved spatial diversity. Then, we address the challenge to accurately abstract our PHY via Link Abstraction (LA). Furthermore, we use LA to demonstrate Scheduling and

Resource Allocation (S&RA) approaches that minimize burst errors. Finally, we demonstrate in an Over-the-Air (OTA) implementation in GNU Radio that our investigations lead to feasible solutions in a practical application.

## 1.4 Contributions and structure

The structure of this thesis is discussed in the following. Where appropriate, we point towards prior work. Moreover, we briefly discuss our contributions in each chapter.

### 1.4.1 Fundamentals

In Chapter 2, we introduce the basic scenario that we consider throughout this work. Based on this scenario, we introduce our system model as well as the channel model. Thus, this chapter serves as a reference to better discuss the inter-dependencies between chapters.

### 1.4.2 Channel coding

In Chapter 3, we focus on channel coding, or Forward Error Correction (FEC), for short packets. We start with an investigation into possible channel codes. Here, we identify polar codes as a promising candidate that were recently adopted for 5G NR [BCL21]. Thus, we introduce polar codes in-depth and discuss algorithmic and software optimizations. Additionally, we propose a technique to further integrate the security and reliability domain to reduce overhead, especially in short packet scenarios. This proposal resulted in a patent [DBD21]. We finalize this chapter with an extensive analysis of error correction performance, mostly FER, and latency benchmarks. Our contribution is a thorough analysis of the fundamental trade-offs between error correction performance and processing latency in a polar codes software implementation.

### 1.4.3 Symbol mapping

Symbol mapping and bit interleaving are often integrated in Bit-Interleaved Coded Modulation (BICM). In Chapter 4, we discuss highly-optimized and standardized soft-demappers that are required for modern wireless communication systems. The focus of this chapter is on soft-demappers with specializations and approximations for lower latency and higher throughput. However, we investigate the latency impact of interleavers and symbol mappers as well. Since the actual computational burden of these processing steps is lower compared to other chapters, this investigation delivers further insight into Central Processing Unit (CPU) features and how to use them advantageously in a software implementation.

### 1.4.4 Multicarrier modulation

In Chapter 5, we investigate Generalized Frequency Division Multiplexing (GFDM) multicarrier modulation and Orthogonal Frequency Division Multiplexing (OFDM) as a simplified special case of GFDM. First, we present theoretical considerations and algorithmic optimizations. Afterwards, investigations including error rate performance evaluations, and latency benchmarks reveal the performance and latency trade-offs. The result is a thorough understanding of how a software implementation may influence latencies and which parameters are the most influential on latency. This chapter incorporates an extension to our prior works [DBD17a, DBD$^+$17b].

### 1.4.5 Cloud radio access network

In Chapter 6, we investigate a Cloud RAN architecture with distributed APs to improve reliability by leveraging spatial diversity. These considerations include Functional Splits (FSs) to increase flexibility and efficiency. Thus, joint signal processing on a cloud platform enables improved reliability. Furthermore, we extend our proposal for a specific FS to forward quantized Log-Likelihood Ratios (LLRs) over a fronthaul [DMB$^+$20]. Without further measures, we would need to forward LLRs with high resolution and thus high data rates over a fronthaul. The employed Information Bottleneck Method (IBM) method allows us to drastically reduce the required fronthaul data rate by quantization while preserving relevant information. The quantizer design is an offline process, and online quantizer selection and quantization is a lightweight operation. Our contribution is an extensive analysis of the required quantized bit rate, the number of individual quantizers, and the required Signal-to-Noise-Ratio (SNR) range to consider for quantizer design.

### 1.4.6 Medium access control

In Chapter 7, we introduce concepts to improve resilience to burst errors on the MAC layer. Based on our KPIs to minimize burst errors and our real-time deadline constraint, we show how Scheduling and Resource Allocation (S&RA) can greatly reduce burst error probabilities with measures to account for delays. To this end, we start the chapter with Link Abstraction (LA) to accurately abstract our previously introduced PHY for system level simulations. This chapter is an extension to our previous works in [DBD19] and [DBD20].

### 1.4.7 Implementation

In Chapter 8, we discuss our OTA implementation in GNU Radio. The resulting demonstrator is deployed in testbeds and thoroughly analyzed. On the one hand, we demonstrate that our previous work on DSP latency investigations and reliability in a Cloud RAN setup are practically implementable. On the other hand, we investigate software design considerations that are an important factor in a fully optimized system.

Finally, in Chapter 9 we draw our overall conclusion, re-iterate our contributions and gathered insights, and present routes for possible future work.

## 1.5 Notation

Throughout this work, we adopt a notation that we want to outline here.

- Indices start at zero [Dyk82].
- Scalar values are lower case italic symbols $h$, i.e. in regular math font.
- Vectors $\boldsymbol{h}$ are in lower case italic and bold.
- Matrices $\boldsymbol{H}$ are in upper case italic and bold.
- Random variable symbols $\mathrm{h}$ use straight text font.
- Transmit signals $x$ are denoted as is.
- Receive values $\tilde{x}$ that indicate a noisy or soft value use a tilde.
- Hard decision receive values $\hat{c}$ carry a hat that indicates a decision for a possible transmit symbol has been made.
- The set of complex numbers is denoted $\mathbb{C}$, as are other common sets of numbers.
- A specific set of values $\mathcal{U}$ is written in upper case calligraphic font.

# Chapter 2

# Industrial radio system model

Hereinafter, we discuss our industrial radio scenario and reference several other works and ongoing efforts that influence it. Our reference use case is presented that joins distinguishing requirements and constraints of industrial systems. Thus, we address the research question how to boost reliability and how to reduce latency in the context of industrial radio systems with short packets. Further, we present the architecture that we have in mind when we present the details of our system. This includes the major components and their connections. Also, the technological foundation of our work is re-iterated. Finally, we consider our theoretical model both for the PHY and MAC to carry out in-depth investigations. This model serves as a reference for subsequent chapters so the reader may return to this chapter to put things into perspective.

We present our scenario in Fig. 2.1 with several wirelessly connected, remote controlled Automated Guided Vehicles (AGVs) in a factory hall and thus an industrial environment. The AGV controller is part of a private Cloud RAN architecture, potentially an edge cloud, that provides the wireless communication system that meets the strict industrial requirements. In order to boost reliability, we consider distributed Radio Access Points (RAPs) that are connected via fronthauls to a cloud platform to jointly serve the AGVs. In our work, we focus on PHY and MAC layers in a single cell while interactions with neighboring cells or a wider network are out of scope.

"All models are wrong, but some are useful" by G. E. P. Box [BD87].

We know and acknowledge this saying. Still, in this chapter we strive to present our useful industrial radio system model. We present an overview of the communication system we have in mind when we dive deeper into the specifics such that an interested reader may be aided to follow the authors train of thought. A SDR platform enables us to develop all algorithms in the digital domain and only consider an equivalent baseband channel. Therefore, we restrict our channel model to the digital domain and consider analog details and frontend complexity to be out of scope.

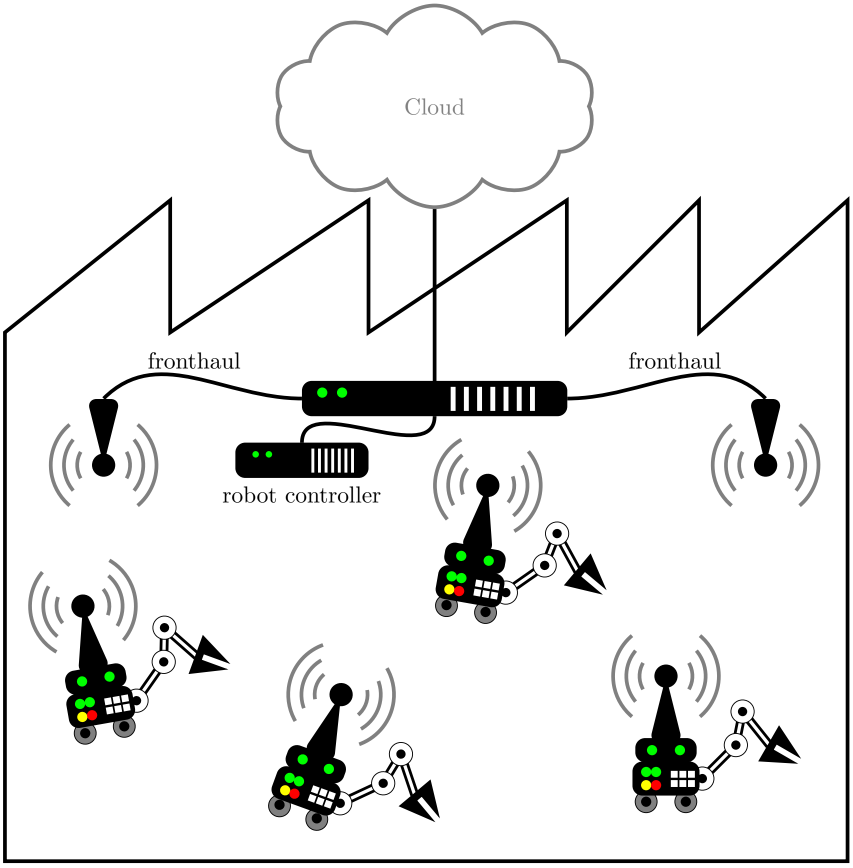


**Figure 2.1:** Scenario: AGVs in a factory hall.

## 2.1 Link model

We want to present a link level model of our system in Fig. 2.2 that represents one link between an AGV and a RAN on the PHY. Here, we discuss the fundamental functionality of each component, while specific details are discussed in subsequent chapters.

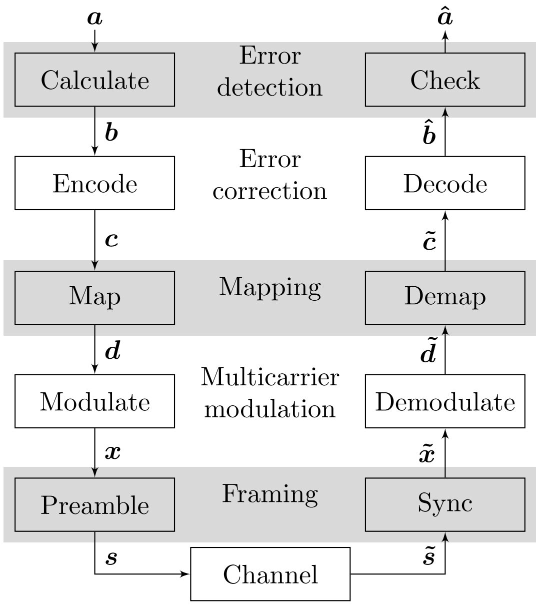


**Figure 2.2:** Link level flowgraph

**Error detection** provides confidence that a received packet is indeed received error free. If a packet does not pass the error detection check, it is discarded. The details are discussed in Chapter 3, specifically Ch. 3.2.3 and 3.3. Generally, a bit vector $\boldsymbol{a} \in \mathbb{F}_2^{N_a}$ with $\mathbb{F}_2 = \{0, 1\}$ from a higher layer enters our system for transmission. A checksum is calculated and appended to every packet and we refer to the output of this block as $\boldsymbol{b} \in \mathbb{F}_2^K$. A receiver uses this checksum to check for correct packet reception. Typically, a Cyclic Redundancy Check (CRC) is used for error detection [ETS20a]. Though, we propose Message Authentification Code (MACode) as an alternative because we see benefits to lower overhead.

**Error correction** or Forward Error Correction (FEC), encompasses all aspects of channel coding such as encoding and decoding but also rate matching, e.g. puncturing.

An information word $\boldsymbol{b} \in \mathbb{F}_2^K$ is encoded into a codeword $\boldsymbol{c} \in \mathbb{F}_2^N$. This codeword carries redundant information that is used to correct errors during the decoding process. We define a coderate

$$R = \frac{K}{N} \tag{2.1}$$

that characterizes the information to codeword size. A low coderate implies that a lot of redundancy is used to improve reliability while high coderates indicate less redundancy at the expense of possibly reduced reliability [RU08]. Chapter 3 discusses FEC in-depth where accurate performance evaluations for specific, state-of-the-art codes are carried out.

**Mapping** describes the process to produce a complex vector $\boldsymbol{d} \in \mathbb{C}^{N_\mathrm{d}}$. A codeword $\boldsymbol{c}$ with $N$ bit is mapped onto a complex vector $\boldsymbol{d}$ with $N_\mathrm{d}$ elements. Every complex symbol $d$ carries $M$ bit of information. We can now define the effective rate

$$R_\mathrm{eff} = M \cdot R = M \cdot \frac{K}{N} \tag{2.2}$$

as the average amount of information bits conveyed by each transmitted symbol in $d$ and $M = \frac{N_\mathrm{d}}{N}$. Chapter 4 discusses symbol mapping in further detail with a focus on Quadrature Amplitude Modulation (QAM) mappings. Especially in case of multipath fading, we want to ensure that received bits are statistically independent. Therefore, Chapter 4 discusses Bit-Interleaved Coded Modulation (BICM) with a bit interleaver before mapping as part of this processing stage.

**Modulation** considers the process to transform all symbols in a packet into a baseband signal that is ready for transmission. We consider multicarrier modulation, specifically OFDM and GFDM, in Chapter 5 to transform a complex vector $\boldsymbol{d}$ into a complex frame $\boldsymbol{x}$ [Gol05, KD18].

**Framing** adds a preamble $\boldsymbol{x}_\mathrm{P}$ to enable synchronization and to obtain initial Channel State Information (CSI). Thus, the complex baseband vector $\boldsymbol{s}$ is comprised of a preamble $\boldsymbol{x}_\mathrm{P}$ and a complex frame $\boldsymbol{x}$. Synchronization is required to recover transmit timing at a receiver. We consider preambles in Chapter 5 and further in Sec. 8.5.2 that match the chosen modulation scheme and require that preambles are more reliably detectable than a data frame because misdetection results in a lost frame.

## 2.2 Small scale channel model

Henceforth we discuss our channel model that we present in Fig. 2.3. It is comprised of several components that we introduce one by one [Rap09, Pro95]. Our focus lies on a software implementation for a Cloud RAN where SDRs provide an equivalent baseband in the digital domain. Thus, we discuss our channel model solely in the digital domain in this chapter. Our channel model implementation [Dem22d] is freely available for the interested reader under the terms of the GNU General Public License v3.0 or later (GPLv3+). This includes, Additive White Gaussian Noise (AWGN), multipath Rayleigh fading, shadowing and temporal or spatial effects. The interested reader is kindly referred to further material for discussion in the analog domain [Rap09, Pro95].

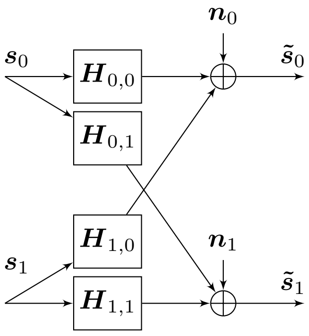


**Figure 2.3:** Time domain 2 × 2 channel model example flowgraph

In Fig. 2.3, we consider a $N_\mathrm{T} \times N_\mathrm{R} = 2 \times 2$ time domain channel example. Thus, we have $N_\mathrm{T} = 2$ transmit antennas and $N_\mathrm{R} = 2$ receive antennas. In general, we transmit a different time domain signal $\boldsymbol{s}_t$ on each transmit antenna with $t \in [0, \ldots, N_\mathrm{T} - 1]$. The signal from $\boldsymbol{s}_t$ propagates through its distinct path with channel matrix $\boldsymbol{H}_{t,r}$, introduced in (2.9), to receive antenna $r \in [0, \ldots, N_\mathrm{R} - 1]$. At every antenna the received signals are super-imposed and corrupted by AWGN noise $\boldsymbol{n}_r$. Finally, we can express the received signal at antenna $r$ with

$$\tilde{\boldsymbol{s}}_r = \left( \sum_{t=0}^{N_\mathrm{T}-1} \boldsymbol{H}_{t,r} \cdot \boldsymbol{s}_t \right) + \boldsymbol{n}_r \tag{2.3}$$

Subsequently we discuss the properties and metrics for $\boldsymbol{n}_r$ and $\boldsymbol{H}_{t,r}$. Properties related to $\boldsymbol{s}_t$ are introduced as a part of our discussions in later chapters.

### 2.2.1 Additive white Gaussian noise

First, we consider the simplified AWGN channel model

$$\tilde{\boldsymbol{s}} = \boldsymbol{s} + \boldsymbol{n} \tag{2.4}$$

where $\tilde{\boldsymbol{s}}$ is the corrupted receive signal. Further, we consider the elements $n$ from the noise vector $\boldsymbol{n}$ that are drawn from a complex Gaussian distribution $n \sim \mathcal{CN}\left(0, \sigma_{\mathrm{n}}^2\right)$ and independent and identically distributed (i.i.d.). The SNR is then denoted by

$$\mathrm{SNR} = \frac{E_{\mathrm{s}}}{N_0} = \frac{E\left\{|\mathrm{s}|^2\right\}}{E\left\{|\mathrm{n}|^2\right\}} = \frac{\sigma_{\mathrm{s}}^2}{\sigma_{\mathrm{n}}^2} \tag{2.5}$$

where $E\left\{|\mathrm{s}|^2\right\}$ is the mean signal power. Moreover, we define the mean signal power for mapped complex symbols $\boldsymbol{d}$ by

$$E_{\mathrm{d}} = E\left\{|\mathrm{d}|^2\right\} = \sigma_{\mathrm{d}}^2 \,. \tag{2.6}$$

Similarly, we may define a mean signal power for other signals when necessary. Further, it is often important to denote a ratio for information bit power over noise power

$$\frac{E_{\mathrm{b}}}{N_0} = R_{\mathrm{eff}} \cdot \frac{E\left\{|\mathrm{d}|^2\right\}}{E\left\{|\mathrm{n}|^2\right\}} = \frac{\sigma_{\mathrm{b}}^2}{\sigma_{\mathrm{n}}^2} \tag{2.7}$$

where $\sigma_{\mathrm{b}}^2$ is the bit power for Non Return to Zero (NRZ) coded bits.

### 2.2.2 Multipath Rayleigh fading channel model

In this section we present our frame-based multipath Rayleigh fading channel model [Pro95]. The channel model in (2.3) considers a $N_{\mathrm{T}} \times N_{\mathrm{R}}$ channel model. First, we discuss the special case of a $1 \times 1$ setup that is trivially extendable to the more general model from (2.3). We consider

$$\tilde{\boldsymbol{s}} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{n} \tag{2.8}$$

as our frame-based channel model that we illustrate in Fig. 2.3. A frame $\boldsymbol{s} \in \mathbb{C}^{N_s}$ is transmitted over a frequency selective Rayleigh block fading channel and corrupted by AWGN. Just like the AWGN channel model in Sec. 2.2.1 we denote the transmit, receive and noise vectors. Thus, we focus on the characteristics of the channel matrix $\boldsymbol{H} \in \mathbb{C}^{N_{\tilde{\mathrm{s}}} \times N_{\mathrm{s}}}$. We assume a frequency-selective Rayleigh fading channel with $N_\mathrm{h}$ taps and consequently the receive frame consists of $N_{\tilde{\mathrm{s}}} = N_\mathrm{s} + N_\mathrm{h}$ samples. The channel matrix $\boldsymbol{H}$ is a Toeplitz matrix and we can denote the channel model in expanded matrix notation

$$\begin{bmatrix} \tilde{s}_0 \\ \tilde{s}_1 \\ \vdots \\ \tilde{s}_{N_{\tilde{\mathrm{s}}}-1} \end{bmatrix} = \begin{bmatrix} h_0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & & \vdots & \vdots \\ h_2 & h_1 & \cdots & 0 & 0 \\ \vdots & h_2 & \cdots & h_0 & 0 \\ h_{N_\mathrm{h}-2} & \vdots & \ddots & h_1 & h_0 \\ h_{N_\mathrm{h}-1} & h_{N_\mathrm{h}-2} & & \vdots & h_1 \\ 0 & h_{N_\mathrm{h}-1} & \ddots & h_{N_\mathrm{h}-3} & \vdots \\ 0 & 0 & \cdots & h_{N_\mathrm{h}-2} & h_{N_\mathrm{h}-3} \\ \vdots & \vdots & & h_{N_\mathrm{h}-1} & h_{N_\mathrm{h}-2} \\ 0 & 0 & 0 & \cdots & h_{N_\mathrm{h}-1} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{N_\mathrm{s}-1} \end{bmatrix} + \begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_{N_{\tilde{\mathrm{s}}}-1} \end{bmatrix} \tag{2.9}$$

where we can identify the channel taps $\boldsymbol{h} = [h_0, h_1, \ldots, h_{N_\mathrm{h}-1}]^T$ whose properties we discuss next.

### Rayleigh fading

Industrial radio measurement campaigns [DHC+19, MBF+05] show that all time-domain channel taps $\boldsymbol{h}$ follow a Rayleigh fading model. Also, the Power Delay Profile (PDP) $\boldsymbol{p}$ of the channel follows an exponential distribution

$$p_k = \frac{e^{-\frac{kT_\mathrm{s}}{\sigma_\mathrm{RMS}}}}{\sigma_\mathrm{RMS}}\,;\; k = 0, \ldots \tag{2.10}$$

with a delay spread $\sigma_\mathrm{RMS}$ in the range 40 ns to 100 ns and the sampling duration $T_\mathrm{s}$. Theoretically, the PDP only decays towards zero, though we assume a maximum relevant channel delay $\tau_\mathrm{max}$, that varies around 200 ns, and $p_k = 0$ if $k > \lfloor \tau_\mathrm{max}/T_\mathrm{s} \rfloor$ [ETS20b]. This relation defines the number of taps $N_\mathrm{h}$ that we consider as well. We obtain the elements of a channel realization vector

$$\boldsymbol{h} = [h_0, \ldots, h_{N_\mathrm{h}-1}] \tag{2.11}$$

that follow

$$\mathrm{h}_k \sim \mathcal{CN}\left(0, p_k\right) \tag{2.12}$$

a complex Gaussian distribution with zero mean and variance $p_k$. Thus, the channel tap amplitude $|\mathrm{h}_k|$ follows a Rayleigh distribution which explains the name Rayleigh fading.

## Temporal and spatial coherence

We focus on short packets, thus we assume a block fading channel, i.e. the channel is constant over the duration of a frame. The channel correlation $\rho$ for consecutive frames can be approximated as

$$\rho = \exp\left\{-23 \cdot \left(\frac{\Delta t v f_c}{c_0}\right)^2\right\} \tag{2.13}$$

depending on time difference $\Delta t$, carrier frequency $f_c$ and relative velocity $v$ between transmitter and receiver [Rap09]. Here, channel correlation $\rho$ quantifies how statistically dependent Gaussian channels are over a time difference $\Delta t$. It is then possible to obtain a channel realization

$$h = \sqrt{\rho}\, h_{\text{prev}} + \sqrt{1-\rho}\, h_{\text{next}} \tag{2.14}$$

from the previous' frame channel realization $h_{\text{prev}}$ and a newly drawn channel realization $h_{\text{next}}$. In case $\rho = 1$, the previous channel realization $h_{\text{prev}}$ solely defines the current channel realization $h$. With (2.13), we expect that $\rho = 1$ only holds for $\Delta t = 0$.

## Frequency domain representation

It is sometimes advantageous to describe a channel model in the frequency domain that represents a frame at the receiver, after demodulation [Pro95]. We describe our equivalent frequency domain channel after Cyclic Prefix (CP) removal and a Discrete Fourier Transform (DFT) for OFDM modulation for a received vector $\tilde{\boldsymbol{d}}$ as discussed in Chapter 5. For GFDM, the channel model holds under the assumption of perfect self interference cancellation. First, we assume that the added CP length $N_{\text{CP}} \geq N_{\text{h}}$ is equal or larger than the number of channel taps $N_{\text{h}}$. Mind how vector length translates to time duration at a given sampling rate $R_{\text{s}} = \frac{1}{T_{\text{s}}}$. Together with the block fading assumption, we can now assume that a frame $\boldsymbol{x}$ is cyclically convolved with a tap vector $\boldsymbol{h}$. The equivalent frequency domain channel

$$\breve{\boldsymbol{h}} = \mathcal{F}_{N_{\text{FFT}}} \boldsymbol{h} \tag{2.15}$$

is obtained where $\mathcal{F}_{N_{\mathrm{FFT}}}$ is a DFT of size $N_{\mathrm{FFT}}$. Now, we denote the equivalent channel model

$$\tilde{\boldsymbol{d}} = \breve{\boldsymbol{H}}\boldsymbol{d} + \boldsymbol{n} \tag{2.16}$$

with a diagonal channel matrix $\breve{\boldsymbol{H}} \in \mathbb{C}^{N_{\mathrm{FFT}} \times N_{\mathrm{FFT}}}$, i.e. all off-diagonal elements are zero, which translates into an element-wise multiplication. The diagonal elements in $\breve{\boldsymbol{H}}$ are the elements from $\breve{\boldsymbol{h}}$. This implies that each received symbol

$$\tilde{d}_k = \breve{h}_k d_k + n \tag{2.17}$$

is transmitted over a frequency flat subcarrier channel plus noise. Finally, the individual frequency channel taps $\breve{h}$ are drawn from a complex Gaussian distribution $\breve{h} \sim \mathcal{CN}(0, 1)$.

Now, we can introduce the per subcarrier Carrier-to-Noise-Ratio (CNR)

$$\mathrm{CNR}_k = \frac{|\breve{h}_k|^2 \sigma_d^2}{\sigma_{\mathrm{n}}^2} \tag{2.18}$$

for a fixed channel realization to characterize the equivalent channel on subcarrier $k$ [DBD19]. Chapter 4 introduces further measures to calculate LLRs with CNRs and Chapter 6 discusses how to combine the information from multiple antennas to improve reliability.

## 2.3 Large scale channel model

The channel model in our work is composed of two components, namely large scale fading and small scale fading. We need both, a small scale and a large scale fading model, to accurately investigate multiple access schemes because wireless communication implies the use of a shared medium. Whenever multiple devices need to transmit, a multiple access scheme is required to organize multiplexing.

Here, we describe large scale fading effects such as our path loss model and shadowing effects [Skl01, MBF$^+$05, ETS18b]. Afterwards, we incorporate these results in our small scale fading model.

It is paramount to investigate which S&RA strategy, and thus multiple access scheme, is appropriate for our use case [DBD20]. This includes an investigation into appropriate metrics that quantify their usefulness for the use case at hand. We will use our model via link abstraction [DBD19] in Chapter 7 to investigate how to grant resources to packets from different users.

We expect industrial radio systems in the 3.7 GHz to 3.8 GHz band within the 5G NR `n78` band [ETS18c]. At least in Germany this band is assigned to campus networks that are expected to play a key role in future industrial radio communication deployments. Mostly, we need to choose between the Time-Division-Duplex (TDD) and Frequency-Division-Duplex (FDD) duplex schemes to organize uplink and downlink transmission. However, the 5G NR standard requires TDD in the expected band `n78` and thus, we focus on this duplex scheme [ETS18c].

For our large scale fading model, we differentiate between Line-Of-Sight (LOS) and Non-Line-Of-Sight (NLOS) channel conditions. LOS channel conditions are often the most favorable because we observe a minimal path loss as well as less stochastic channel effects that deteriorate a signal. NLOS conditions have a more severe impact on a channel. This may result in higher path loss and larger shadowing variability. A system that can fulfill its requirements under NLOS conditions is expected to perform well under LOS conditions. Mostly, we consider industrial radio environments where NLOS conditions dominate [DHC+19, MBF+05].

**Path loss** expresses the received signal power depending on distance $d_\mathrm{g}$, carrier frequency $f_c$ and path loss exponent $\eta$. We denote received signal power with transmit power $P_\mathrm{t}$ and speed of light $c_0$ as

$$P_\mathrm{r}(d_\mathrm{g}) = P\left(f_c, 1, 2\right) - 10 \cdot \eta \cdot \log_{10}\left(\frac{d_\mathrm{g}}{d_0}\right) \text{ [dBm]} \tag{2.19}$$

$$P\left(f_c, d_\mathrm{g}, \eta\right) = P_\mathrm{t} + G_\mathrm{t} + G_\mathrm{r} + 10\log_{10}\left(\frac{c_0^2}{(4\pi f_c)^2 \cdot d_\mathrm{g}^\eta}\right) \text{ [dBm]} \tag{2.20}$$

where all values are in logarithmic units [Skl01]. The path loss exponent $\eta$ depends on channel conditions, e.g. LOS or NLOS [ETS18b]. The reference received power $P\left(f_c, 1, 2\right)$ is defined at a reference distance $d_0 = 1\,\mathrm{m}$ with $\eta = 2$ to ensure that far field assumptions are valid [Rap09, HZA+11]. Also, in most cases we assume isotropic antennas at the transmitter and receiver, i.e. $G_\mathrm{t} = G_\mathrm{r} = 1$.

**Thermal noise** is a receiver characteristic that corrupts a signal. Under the assumption that thermal noise is the source of AWGN, we express the thermal noise power

$$E\left\{|\mathrm{n}|^2\right\} = \sigma_\mathrm{n}^2 = N_0 = 30 \cdot \log_{10}\left(\kappa T B_\mathrm{s}\right) \text{ [dBm]} \tag{2.21}$$

with bandwidth $B_\mathrm{s}$, temperature $T$ and Boltzmann constant $\kappa$ [Skl01].

**Shadowing** considers large scale fading effects that impact propagation conditions. The random variable S follows a Gaussian distribution with $\mu = 0$

and shadowing deviation $\sigma_{\mathrm{SF}}$ dependent on channel conditions [MBF$^+$05, ETS20b]. Thus, we denote

$$\mathrm{S} \sim \mathcal{N}\left(0, \sigma_{\mathrm{SF}}^2\right) \tag{2.22}$$

in the log-domain. Since we consider logarithmic values for $\sigma_{\mathrm{SF}}$ and $S$ for the Gaussian distribution, this may be referred to as a log-normal distribution for the corresponding linear values. Shadowing is a large scale parameter that is more pronounced under NLOS conditions and spatially correlated

$$\rho_{\mathrm{s}} = e^{-\frac{\Delta d_{\mathrm{g}}}{d_{\mathrm{corr}}}} \tag{2.23}$$

with shadowing correlation distance $d_{\mathrm{corr}}$ [ETS20b]. It is possible to convert spatial into temporal correlation with known velocity $v$.

**Link Budget** Finally, we denote our link budget in terms of SNR as

$$\mathrm{SNR} = P_{\mathrm{r}}(d) - N_0 - F - S \ . \tag{2.24}$$

Here, radio receivers are characterized with a noise figure $F$ that further contributes to a higher noise floor. We use this SNR value for our small scale fading model to abstract away physical details such as absolute transmit power.

## 2.4 Chapter summary

In this chapter we laid out our reference scenario with several robots, or AGVs in a factory hall. Based on this scenario, we discussed our system structure and boundaries. This system is based on a solid mathematical foundation for numerical evaluation that we introduced in this chapter as well. The model includes, AWGN as well as small and large scale fading effects. Furthermore, the model is topped off by coherence considerations.

# Chapter 3

# Low-latency channel coding for short packets

Current modern communication systems rely on channel coding to ensure reliable transmission over error prone channels [RU08, ETS20a]. It is, however, notoriously difficult to design a communication system that concurrently fulfills contradicting requirements. While it is theoretically possible to reach the ultimate limit, Shannon capacity, for error free transmission over AWGN channels, it is also infeasible because it requires infinitely long codewords [RU08]. Further, longer codewords increase latency because it takes more time to transmit all coded bits but also because algorithmic complexity tends to increase [Arı09].

Moreover, we focus on mission critical applications where Machine to Machine (M2M) communication is prevalent [DPS18, 3GP19a]. Especially in the realm of URLLC we expect particularly short packets with a low latency requirement and a high reliability requirement [3GP19a]. In this chapter, we consider polar codes as a viable candidate to meet these requirements.

Mission critical applications also require secure communication and authentication to prevent unauthorized third parties from tampering with the system at hand. It is advisable to integrate security into the communication system early on instead of treating it as an afterthought [MB17]. Firstly, encryption secures data packets from inspection by unauthorized third parties. While encryption does not introduce overhead per se, authentication does. Especially systems that convey short packets suffer from overhead which reduces efficiency. In our patented proposal, we focus on the additional overhead added by authentication requirements [DBD21]. A cryptographic checksum, e.g. Cipher-based Message Authentification Code (CMAC) or

Keyed-hash Message Authentification Code (HMAC), provides information in order to reliably identify the source of a packet and thus verifies a packet's authenticity. In this chapter, we discuss our novel approach to merge the security and reliability domains to reduce overhead. Moreover, we contribute a set of experiments to evaluate its feasibility.

We focus on Cloud RAN development. Therefore, we need more insight into software implementation aspects of the discussed algorithms. This should serve as an indicator for expected performance and latency. Further, under a software development paradigm, different algorithms and approaches than under a hardware development paradigm may lead to better results. Hence, we discuss polar codes with all required components.

In this chapter we re-evaluate the findings in [SWJ$^+$16] that polar codes, especially in conjunction with advanced decoders, are an attractive choice for short packet communication systems. We start with an investigation into different channel codes to justify our investigation into polar codes. Furthermore, our open source polar coding implementation [DL22] reveals competitive results in terms of error-correction performance, latency, and throughput in comparison to previous works [Gia16, CHL$^+$19]. These investigations extend towards selected parametrizations with corresponding trade-off discussions. The focus of these investigations is on URLLC with short packets. Also, since most advanced polar decoders make use of a CRC, we propose to fuse the security and coding domains to leverage synergies that help to reduce frame overhead and thus boost efficiency especially for small packet communication. Specifically, we propose a joint MACode-FEC polar code for reduced overhead and a polar decoder that makes use of a MACode checksum. Finally, we present our investigation on error correction performance, latency, and throughput for software polar encoders and decoders.

## 3.1 Codes for short packets

We compare multiple modern codes and their respective performance for short codes [SWJ$^+$16, BCL21, BAL$^+$19]. There exists a software library that supports several FEC technologies named *aff3ct* that we use for several codes [CLT$^+$21, CHL$^+$19]. In accordance with [SWJ$^+$16] we selected three candidates, namely, Turbo, Low Density Parity Check (LDPC), and Polar codes for this comparison.

### 3.1.1 Code candidates

**LTE Turbo codes** are well known in 4th Generation (4G) systems [ETS20c, CHL$^+$19, WP16]. The decoding complexity for this class of codes is steered with the number of iterations $I$. Generally, more iterations allow for better error correction performance at the cost of more complexity.

**LDPC codes** are used in several communication standards such as WiFi and DVB-S2 [Gra19]. Also this code class is selected for 5G NR data payload [ETS20a, BAL$^+$19]. Like turbo codes, LDPC codes are designed for iterative decoding. Thus, their decoding complexity is dominated by the number of decoder iterations $I$. LDPC codes exhibit lower complexity per iteration compared to Turbo codes and thus the number of iterations is not directly comparable [SWJ$^+$16].

**Polar codes** are a new class of FEC first introduced by Arıkan [Arı09] that are discussed in-depth in Sec. 3.2. For State-of-the-Art correction performance, we use a CRC-Aided Successive Cancellation List (CA-SCL) decoder discussed in Sec. 3.2.3. The CA-SCL decoder complexity is steered by its list size $L$. In the current section we want to motivate our choice to prefer polar codes for short packets.

### 3.1.2 Code comparison

We present simulation results over an AWGN channel for several codes in Fig. 3.1 where we assume $K = 256$ information bits and a code rate $R \approx 0.5$. The difference in code rate for Turbo codes is due to implementation specifics [CLT$^+$21]. We use a $(512, 256)$ LDPC code according to the Consultative Committee for Space Data Systems (CCSDS) specification and *aff3ct* for encoding and decoding [CCS15].

In Fig. 3.1, we observe that LDPC codes exhibit approximately 1 dB worse performance than Turbo codes at $\text{FER} = 10^{-1}$, even with a high number of iterations $I$. At $\text{FER} = 10^{-3}$ Turbo codes outperform LDPC codes by approximately 0.8 dB. Further, even a simple polar decoder with $L = 1$, known as Successive Cancellation (SC) decoder, exhibits a 0.5 dB better performance at $\text{FER} = 10^{-1}$ than LDPC codes and the SC decoder is only approximately 0.1 dB worse at $\text{FER} = 10^{-3}$ compared to the LDPC decoder with $I = 100$ iterations. Turbo codes with $I = 32$ iterations almost reach CA-SCL decoder performance with list size $L = 32$ at $\text{FER} = 10^{-3}$. However, CA-SCL decoders with $L = I$ perform approximately 0.2 dB better at $\text{FER} = 10^{-1}$. It should be noted that the CA-SCL decoder achieves this performance at a fraction of the complexity [SWJ$^+$16, GRLa17]. The good

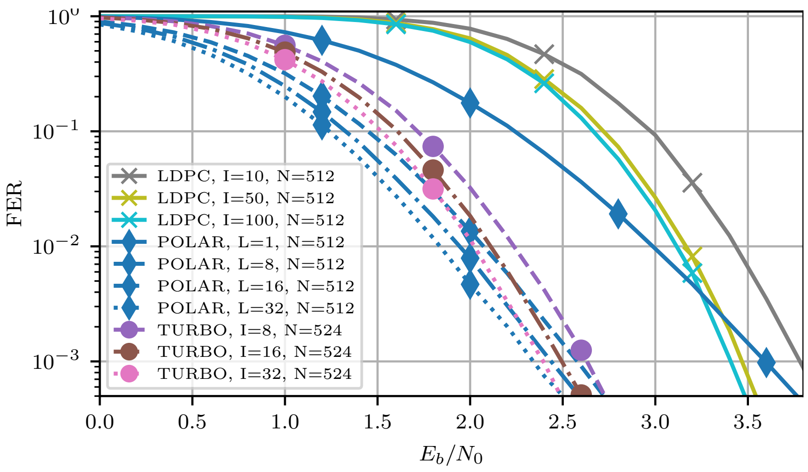


**Figure 3.1:** Short code comparison with $R \approx \frac{1}{2}$: CCSDS LDPC $(512, 256)$, LTE Turbo code $(524, 256)$, and polar codes $(512, 256)$ with CA-SCL decoding.

FER performance in our simulations indicate that polar codes are suitable candidates for short packet transmission. Moreover, the complexity of polar codes is much lower to achieve the FER performance of their corresponding LDPC and Turbo codes [SWJ+16, GRLa17].

With lower code rate $R$ as shown in Fig. 3.2, we corroborate our conclusion that polar codes are good candidates for short packet transmission. At FER $= 10^{-1}$ polar codes with $L = I$ perform approximately 0.2 dB better than Turbo codes while at FER $= 10^{-3}$ the performance is on par. For URLLC applications we generally expect lower code rates to improve reliability and the results in Fig. 3.2 are of particular interest. Finally, for Cloud RAN systems polar decoders are of special interest because of their lower decoding latency compared to Turbo code and LDPC software implementations [Gia16, LGJ20, Gra19, XWXG19].

## 3.2 Polar codes

Polar codes were first presented by Arıkan in 2009 [Arı09]. The name polar code originates from the channel polarization effect that is exploited for polar codes. They are the first class of codes to asymptotically achieve capacity of any binary input symmetric Discrete Memoryless Channel (DMC) with a SC decoder [Arı09, TV13]. Moreover, a polar encoder and decoder exhibits a low

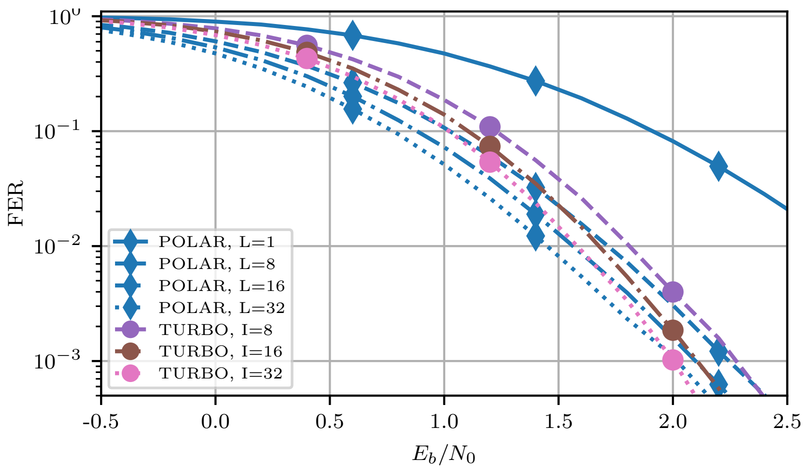


**Figure 3.2:** Short code comparison with $R \approx \frac{1}{3}$: LTE Turbo code $(516, 168)$, and polar codes $(512, 168)$ with CA-SCL decoding.

complexity $\mathcal{O}(N_c \log N_c)$ and may be further optimized via Fast Simplified Successive Cancellation (Fast-SSC) and more specializations [Gia16, HA17]. In the finite length regime, polar codes require more sophisticated decoders to achieve competitive error correction performance. First, Successive Cancellation List (SCL) decoders were proposed in [TV15] that maintain a candidate list. Further, the authors proposed to check the candidates in the list with a CRC, sometimes referred to as a geany, to produce the first candidate that passes this check as the final codeword. Other decoder strategies emerged as well such as Soft CANcellation (SCAN) via Express Journey for Belief Propagation (XJ-BP) and Successive Cancellation Flip (SCFlip) [LSL$^+$16, XCC15, CSD18].

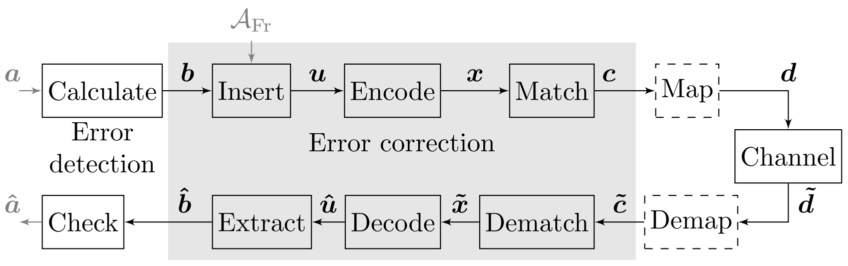


**Figure 3.3:** Channel coding flowgraph

Fig. 2.2 presents a system overview. Here, Fig. 3.3 focuses on a more detailed coding flowgraph to clarify notation and code components. A bit vector $\boldsymbol{a}$ enters the encoder chain and a checksum may be added to be able to verify correct reception [KC04, ETS20a]. At the end of the receiver chain, we obtain the bit vector $\hat{\boldsymbol{a}}$ that is a obtained after hard decision. Thus, we obtain an information bit vector $\boldsymbol{b} \in \mathbb{F}_2^K$ of size $K$ to enter the polar code specific processing chain. The polar receiver chain yields the bit vector $\hat{\boldsymbol{b}}$ after decoding. While error detection and correction serve different purposes, they are closely intertwined and many advanced polar decoders use some error detection, mostly a CRC, to improve their error correction capabilities.

Next, we combine $\boldsymbol{b}$ with frozen bits in the vector $\boldsymbol{u}$. Essentially, the frozen bit position set $\mathcal{A}_{\mathrm{Fr}}$ indicates all positions in $\boldsymbol{u}$ that are set to a fixed value. While it is possible to choose an arbitrary frozen bit vector, conventionally the all zero bit vector $\mathbf{0}^{N_\mathrm{c}-K}$ is used [STG+16, ETS20a]. This choice enables decoder optimizations, e.g. Fast-SSC, without further complication. The exact choice for $\mathcal{A}_{\mathrm{Fr}}$ depends on the chosen channel construction that will be discussed in Sec. 3.2.2. It may further depend on the chosen rate matching strategy that will be discussed in Sec. 3.2.4. Channel construction is an offline task that focuses on error correction performance by finding the best possible set of frozen bit positions $\mathcal{A}_{\mathrm{Fr}}$ [TV13, VVH15]. Naturally, encoding and decoding are online tasks where research efforts focus on decoder error correction performance, latency, and throughput improvements. The number of information bits in a polar codeword may be in the range $K \in [0, N_\mathrm{c}]$. While theoretically possible, the options $K = 0$ and $K = N_\mathrm{c}$ are useless because the former does not convey information while the latter cannot provide error correction.

The encoder transforms the bit vector $\boldsymbol{u} \in \mathbb{F}_2^{N_\mathrm{c}}$ into a codeword $\boldsymbol{x} \in \mathbb{F}_2^{N_\mathrm{c}}$ as discussed in Sec. 3.2.1. A polar code decoder expects soft information $\tilde{\boldsymbol{x}} \in \mathbb{R}^{N_\mathrm{c}}$, i.e. LLRs, for decoding with error correction as discussed in Sec. 3.2.3. The dashed blocks are discussed in Chapter 4 along with further LLR considerations.

Polar codewords are originally only defined for powers of 2 with $N_\mathrm{c} = 2^{n_\mathrm{c}}$ and $n_\mathrm{c} > 0$; $n_\mathrm{c} \in \mathbb{Z}^+$. Thus, a flexible polar code requires rate matching to yield codewords $\boldsymbol{c}$ of arbitrary size $N$. Puncturing to achieve arbitrary code sizes is discussed in Sec. 3.2.4 and may have an impact on frozen bit positions.

Polar encoder, decoder, and channel construction are entangled. Encoder and decoder require the results of channel construction to be parameterized. In turn, channel construction operates on the assumption of specific encoder and decoder structures as discussed in Sec. 3.2.2. We try to disentangle all three components and assume in the discussion for one component that the other components are given.

### 3.2.1 Encoder

First, consider a bit vector $\boldsymbol{u} \in \mathbb{F}_2^2$ of size 2 with $\mathbb{F}_2 \in \{0,1\}$ and $\boldsymbol{u} = (u_0, u_1)$ and compute a codeword

$$\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G} \quad \text{with} \quad \boldsymbol{G} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{3.1}$$

over the Galois field GF(2) [Arı09]. Now, we extend this scheme to bit vectors $\boldsymbol{u} \in \mathbb{F}_2^{N_\mathrm{c}}$ of size $N_\mathrm{c} = 2^{n_\mathrm{c}}$ where $n_\mathrm{c} \in \mathbb{Z}^+$ is a positive integer. The generator matrix $\boldsymbol{G}$ is the $n_\mathrm{c}$th Kronecker product of the polar kernel $\boldsymbol{F}$ [Arı09, STG+16] that is defined recursively

$$\boldsymbol{G} = \boldsymbol{F}^{\otimes n_\mathrm{c}} = \begin{pmatrix} \boldsymbol{F}^{\otimes(n_\mathrm{c}-1)} & \boldsymbol{0} \\ \boldsymbol{F}^{\otimes(n_\mathrm{c}-1)} & \boldsymbol{F}^{\otimes(n_\mathrm{c}-1)} \end{pmatrix} \quad \text{with} \quad \boldsymbol{F}^{\otimes 1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{3.2}$$

Arıkan further describes a bit reversal matrix $\boldsymbol{B}$ to ensure input and output positions correspond to each other [Arı09]. This is not strictly necessary and is sometimes omitted, e.g. in [ETS20a]. Without a bit reversal matrix, the codeword is in bit reversed order otherwise it is in natural order [SGV+16]. One needs to be careful to correctly distinguish between those two options in order to insert frozen bits correctly. Other than that, these options are equivalent and the preferred order is implementation defined [GSL+18]. We obtain the codeword for transmission in natural order by

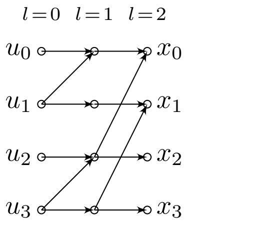
$$\boldsymbol{x} = \boldsymbol{u} \cdot \boldsymbol{B} \cdot \boldsymbol{G} \quad \text{with} \quad \boldsymbol{G} = \boldsymbol{F}^{\otimes n_\mathrm{c}} \tag{3.3}$$

with the bit reversal matrix $\boldsymbol{B}$. $\boldsymbol{B}$ is defined by permuting the rows of an identity matrix $\boldsymbol{I}_{N_\mathrm{c}} \in \mathbb{F}_2^{N_\mathrm{c} \times N_\mathrm{c}}$. First, each row is indexed $0 \ldots N_\mathrm{c} - 1$ and represented with $n_\mathrm{c}$ binary digits. Then, we reverse the binary digits of this representation and obtain the target row index with that result. Other numbering conventions, i.e. non-zero based, are used as well, e.g. in[Arı09], but would only make the interpretation more convoluted [Dyk82].

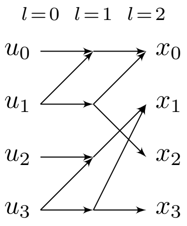
#### Low complexity encoder

The encoder, as discussed so far, computes a matrix multiplication with $\mathcal{O}\left(N_\mathrm{c}^2\right)$ complexity. In Fig. 3.4 we present graphs that illustrate polar encoders with low complexity $\mathcal{O}\left(N_\mathrm{c} \log N_\mathrm{c}\right)$. These graphs resemble Fast

Fourier Transform (FFT) butterfly structures [FJ05]. We consider a bit vector $\boldsymbol{u} \in \mathbb{F}_2^{N_\mathrm{c}}$ with indices $k$. The low complexity encoder consists of $n_\mathrm{c}$ layers with indices $l$. At each node in the graphs where two arrows end, we perform addition over GF(2).



**(a)** Reversed encoder $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{G}$



**(b)** Natural encoder $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{B}\boldsymbol{G}$

**Figure 3.4:** Low complexity butterfly representation of polar encoders

### Systematic polar codes

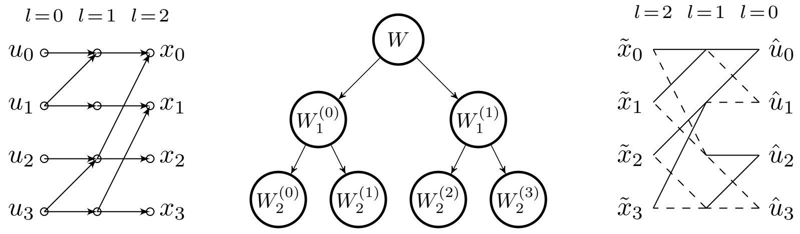
Originally polar codes were presented in their non-systematic form [Arı09]. A systematic code contains information bits as a part of its codeword [RU08]. Systematic polar codes were proposed in [Arı11] and further optimized in [VHV16, STG+16]. When compared to non-systematic polar codes, the structure of the code vector $\boldsymbol{x}$ differs slightly such that $\boldsymbol{x}_{\mathcal{A}_\mathrm{I}} = \boldsymbol{u}_{\mathcal{A}_\mathrm{I}}$ while there are parity bits $\boldsymbol{x}_{\mathcal{A}_\mathrm{Fr}}$ at the other indices. Systematic polar codes exhibit the same complexity as their non-systematic counterparts, albeit they usually require an additional encoder pass both in the encoder and decoder [STG+16]. While systematic polar codes show lower Bit-Error-Rate (BER), their FER performance is unaffected [Arı11]. Still, all approaches to boost performance and throughput, or lower latency, for non-systematic polar codes apply to their systematic counterparts as well.

## 3.2.2 Channel construction

The name polar code originates from the channel polarization effect that is exploited for polar codes. Channel construction is an offline task that needs knowledge about the encoder and decoder structures in Fig. 3.5a, 3.5c to obtain reliability information of virtual bit channels $W_l^{(i)}$.

We consider symmetric binary input $\mathbb{F}_2 = \{0, 1\}$ over a DMC, such as a Binary Erasure Channel (BEC), Binary Symmetric Channel (BSC) or AWGN channel $W$ [RU08]. Further, we consider a polar encoder, cf. Fig. 3.5a, and a SC polar decoder, cf. Fig. 3.5c, to obtain virtual bit channels $W_{n_\mathrm{c}}^{(i)}$ for transmission of the bit $u_i$ bit to $\hat{u}_i$. While all code bits $x_i$ are transmitted over the same channel $W$, virtual bit channels $W_l^{(i)}$ are derived from $W$ for each $u_i$.

Starting with $W$, we combine this channel recursively in $n_\mathrm{c}$ layers shown in Fig. 3.5b to obtain an individual virtual bit channel $W_{n_\mathrm{c}}^{(i)}$ for every bit. From layer $l$ to layer $l+1$, we perform a channel downgrade to the left and a channel upgrade to the right, which corresponds to the lower and higher index positions respectively. The exact computation method for upgrades and downgrades depends on the chosen algorithm [VVH15, HBL$^+$17]. The capacity of a virtual bit channel $W_l^{(i)}$ either tends to 0 or 1 due to the polarization effect [Arı09]. Since channel construction is an offline task, research efforts focus on error correction performance improvements [TV13, VVH15]. Still, accurate results are often computationally heavy and exact computation is often intractable. We want to summarize common approaches for channel construction and note that an in-depth analysis is out-of-scope for this work.



**(a)** Encoder **(b)** Channel construction **(c)** Decoder

**Figure 3.5:** Polar code components

Different strategies exist that yield the required virtual bit channel reliability. Originally, the Bhattacharyya Bound (BB) method is proposed for BEC [Arı09]. The BB method is computationally tractable and exact for BEC and it is proposed to use it as an approximation for other channels as well [VVH15]. Alternatively, for all channels Monte-Carlo simulations may be performed to obtain the desired information [Arı09]. In [TV13], the authors propose the Density Evolution (DE) method to calculate reliability information. Further, Gaussian Approximation (GA) has been studied to determine reliability information [Tri12, WLS14, DNS$^+$17]. All these approaches perform channel construction for a specifically defined channel $W$, e.g AWGN with $\sigma_\mathrm{n}^2 = 0.5$. With either method it is possible to define a reliability table, i.e. a sorted list of indices in ascending or descending reliability order, that can be used to parameterize a wide range of polar codes [ETS20a, CLT+21].

Recently, $\beta$-Expansion (BE) was introduced as a more efficient option to compute channel construction [HBL+17]. This channel construction option relies on Universal Partial Order (UPO) to order virtual bit channels according to their reliability. UPO is a polar code property that indicates the relation between many, but not all, virtual bit channels. The authors in [HBL+17] point out that the order of virtual bit channels up to $N_\mathrm{c} = 16$ is unique. Thus, for $N_\mathrm{c} = 16$ only 17 valid parametrizations for $\mathcal{A}_\mathrm{Fr}$ exist including a coderate $R = 0$ and $R = 1$ polar code.

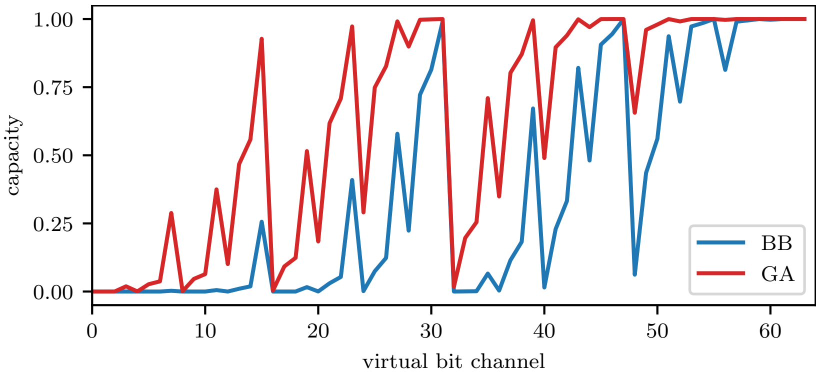


**Figure 3.6:** Virtual bit channel capacities for $N_\mathrm{c} = 64$ and dSNR $= 0\,\mathrm{dB}$

Fig. 3.6 shows an example for polar code virtual bit channel capacities with $N_\mathrm{c} = 64$ for BB and GA [Arı09, DNS+17]. Here, we use capacity as a measure of bit channel reliability. We find the frozen bit position set $\mathcal{A}_\mathrm{Fr}$ by selecting the $N_\mathrm{c} - K$ indices that correspond to the lowest virtual bit channel capacity. Correspondingly, the information bit position set $\mathcal{A}_\mathrm{I}$ is obtained by selecting the $K$ indices that correspond to the highest virtual bit channel capacity.

Polar codes are designed for a specific SNR but often the channel SNR is unknown during code design. Fully SNR dependent polar code design is infeasible and thus the authors in [VVH15] proposed to use one Design SNR (dSNR) that yields a set of frozen positions $\mathcal{A}_\mathrm{Fr}$ with good FERs over a wide range of SNRs. Polar codes are then defined as $(N_\mathrm{c}, K, \mathcal{A}_\mathrm{Fr})$ codes.

### 3.2.3 Decoder

Decoders are the most complex part of a channel coding processing chain. Originally, Arıkan proposed a SC decoder to prove that polar codes achieve capacity [Arı09]. The combination of a low complexity encoder and SC decoder together are assumed to be present in the channel construction process. With growing codeword size, e.g. $N_\mathrm{c} = 2^{17}$, this decoder strategy yields very good error correction performance due to the channel polarization effect. For medium or short codes, we need additional measures to achieve competitive performance.

Several approaches to improve performance are available. SCAN decoders employ Belief Propagation (BP) and are able to produce soft output [FB14]. SCFlip decoders focus on finding the first incorrectly decoded bits to improve on the original SC decoder [CSD18]. Successive Cancellation Stack (SCS) and SCL decoders maintain a candidate list to defer bit decisions [TV15, NC21]. Further, CA-SCL decoders extend SCL decoders by searching the candidate list for a candidate to pass a checksum test [TV15]. We focus on SC, SCL, and CA-SCL polar decoders because they are found to achieve competitive performance, low-latency, and high throughput in software implementations [SGV+16].

#### Successive cancellation

Here, we present the Successive Cancellation (SC) decoder strategy that Arıkan proposed to prove that polar codes achieve capacity. First, a 2 bit decoder is introduced in Fig. 3.7 and then further extended to larger codewords [Arı09]. We will discuss more optimizations starting in Sec. 3.2.3.

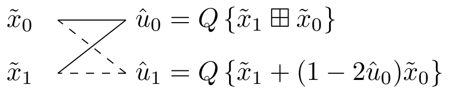

$\tilde{x}_0$
$\hat{u}_0 = Q\left\{\tilde{x}_1 \boxplus \tilde{x}_0\right\}$
$\tilde{x}_1$
$\hat{u}_1 = Q\left\{\tilde{x}_1 + (1 - 2\hat{u}_0)\tilde{x}_0\right\}$

**Figure 3.7:** 2 bit SC polar decoder with the decision function $Q\left\{\ldots\right\}$ in (3.6)

We use the LLRs of a codeword $\tilde{\boldsymbol{x}} \in \mathbb{R}^2$ for decoding that we discuss in Sec. 4.1. Here, we want to reiterate the LLR definition

$$\tilde{x} = \ln \frac{p\left(x = 0|\tilde{d}\right)}{p\left(x = 1|\tilde{d}\right)} \tag{3.4}$$

for a single transmit bit $x$ and a single noisy BPSK receive symbol $\tilde{d}$. Here,

$p\left(x|\tilde{d}\right)$ is a conditional probability density function. Now, we compute

$$\begin{aligned}\tilde{u}_0 = f\left(\tilde{x}_0, \tilde{x}_1\right) = \tilde{x}_0 \boxplus \tilde{x}_1 = & \log \frac{1 + e^{\tilde{x}_0} e^{\tilde{x}_1}}{e^{\tilde{x}_0} + e^{\tilde{x}_1}} \\ \approx & \operatorname{sgn}(\tilde{x}_0)\operatorname{sgn}(\tilde{x}_1)\min(|\tilde{x}_0|, |\tilde{x}_1|)\end{aligned} \tag{3.5}$$

where sgn(...) is the sign function. Throughout this investigation we use the approximation in (3.5) because it yields good results and is significantly more lightweight. With $\tilde{u}_0$, we reach a decoder endpoint and are able to detect the corresponding bit $\hat{u}_0$ via the decision function

$$\hat{u}_i = Q\left\{\tilde{u}_i\right\} = \begin{cases} 0 & \tilde{u}_i > 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.6}$$

where we perform hard decision on a LLR in accordance with (3.4). The first decoded bit must be a frozen bit in case of $R < 1$ [HBL+17]. This is a direct result from channel polarization as discussed in Sec. 3.2.2 where the first bit corresponds to the virtual bit channel with lowest capacity. Without loss of generality, it is common in literature to assume all frozen bits are 0 [STG+16, ETS20a] and thus, we set $\hat{u}_i = 0$; $\forall i \in \mathcal{A}_{\mathrm{Fr}}$ regardless of (3.6).

Successively, we can compute

$$\tilde{u}_1 = g\left(\tilde{x}_0, \tilde{x}_1, \hat{u}_0\right) = \tilde{x}_0 + (1 - 2\hat{u}_0)\tilde{x}_1 = \tilde{x}_0 + (-1)^{\hat{u}_0}\tilde{x}_1 \tag{3.7}$$

where $\hat{u}_0 \in \mathbb{F}_2$ and finally decide $\hat{u}_1 = Q\left\{\tilde{u}_1\right\}$. At this point we finish the 2 bit polar decoder process. Now, we extend this decoder recursively to $N_\mathrm{c}$ bit [Arı09, Gia16].

The extension to an 8 bit decoder, shown in Fig. 3.8 is straight forward. First, we extend our notation to LLRs at layer $l$ and element $i$ as $\tilde{x}_{l,i}$. The layer $l$ is indicated above the graph in Fig. 3.8 where we start at layer $l = n_\mathrm{c}$ and finish the decoding process at layer $l = 0$.

We start to compute the elements

$$\tilde{x}_{l,i} = f\left(\tilde{x}_{l+1,i}, \tilde{x}_{l+1,i+2^l}\right) \tag{3.8}$$

with the function (3.5). We need to stick to a successive decoding strategy, thus we only compute values necessary to decide for $\hat{u}_0$ which is indicated with solid lines in Fig. 3.8. Afterwards, we obtain $\hat{u}_1 = Q\left\{g\left(\tilde{x}_{1,0}, \tilde{x}_{1,1}, \hat{u}_0\right)\right\}$ with (3.7). We progress successively through all $\hat{u}_i$ as shown in Fig. 3.8. For nodes where two dashed lines end from a higher layer $l > 0$, we compute

$$\tilde{x}_{l,i} = g\left(\tilde{x}_{l+1,i+2^l}, \tilde{x}_{l+1,i}, \hat{u}_{l,i+2^l}\right) \tag{3.9}$$

together with the previously decided and partially re-encoded bits $\hat{u}_{l,i+2^l}$. Generally, whenever two solid lines end in a node, we use (3.5) and dashed lines require (3.7) to be computed.

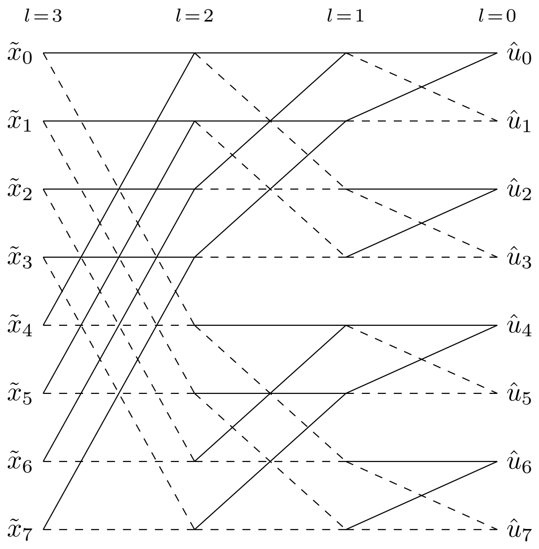


**Figure 3.8:** 8 bit SC polar decoder

### How to implement a decoder in C++

The SC decoder structure as discussed so far, exhibits the general low complexity structure. We want to discuss the implementation basics in C++ briefly to supply readers with an idea how to implement a polar decoder from theory to a Single-Instruction-Multiple-Data (SIMD) optimized implementation [ISO17, cpp21b]. This discussion may be complemented by [Gia16]. However, it is crucial to notice that these examples are only useful in a context where the overall software structure permits a high-throughput and low latency implementation. The interested reader may refer to [DL22] to immerse oneself in the C++ implementation structure.

We start with a straight forward implementation of (3.5) in Listing 3.1.

**Listing 3.1:** Straightforward C++ implementation of $f(\tilde{x}_0, \tilde{x}_1)$ (3.5)

```
inline float calculate_f(const float llr0, const float llr1)
{
  return std::copysign(1.0, llr0) *
         std::copysign(1.0, llr1) *
         std::min(std::abs(llr0), std::abs(llr1));
}
```

The function `std::copysign(1.0, llr)` takes the sign of `llr` and applies it to the first argument `1.0`. Besides, this function should be self-explanatory.

In Fig. 3.8, we observe that (3.5) needs to be computed in parallel on several nodes at every layer. This may be optimized with SIMD intrinsics for Streaming SIMD Extensions (SSE) or Advanced Vector Extensions (AVX) on x86 platforms [Int21]. In Listing 3.2 the input parameters `llrs` are 256 bit wide registers that hold eight 32 bit floating point values. Thus, we perform the function from Listing 3.1 on eight values in parallel, i.e. we vectorize this function.

**Listing 3.2:** AVX vectorized C++ implementation for eight values of $f\,(\bullet, \bullet)$.

```
 1  #include <immintrin.h>
 2  inline __m256 calculate_f_vector(const __m256 llrs0,
 3                                   const __m256 llrs1)
 4  {
 5    const __m256 sign_mask = _mm256_set1_ps(-0.0f);
 6
 7    __m256 sgnV = _mm256_and_ps(sign_mask,
 8                                _mm256_xor_ps(llrs0, llrs1));
 9
10    __m256 abs0 = _mm256_andnot_ps(sign_mask, llrs0);
11    __m256 abs1 = _mm256_andnot_ps(sign_mask, llrs1);
12    __m256 minV = _mm256_min_ps(abs0, abs1);
13    return _mm256_or_ps(sgnV, minV);
14  }
```

Since the code for this function is not as self-explanatory as Listing 3.1, we want to point out several key properties. Starting with line 1, we include the appropriate headers to make AVX intrinsics, e.g. `_mm256_min_ps`, available [Int21]. We obtain signs in line 7 via bit-wise operations combined with a bit mask. The Most Significant Bit (MSB) in a floating point (fp-32) value carries the sign which we obtain by zeroing all other bits [IEE19]. Bit manipulation of fp-32 values should be approached with caution because it is prone to errors. Afterwards, in line 10 to 12, we compute the absolute minimum by setting the sign bit to zero and obtaining the minimum. Note how each of those eight values in `llrs0` and `llrs1` is computed in parallel such that we obtain the minimum of the first elements in `llrs0` and `llrs1`, and so on. Finally, we compute the final result in line 13 with a bit-wise `OR` operation to combine the sign bit with the minimum absolute value.

**Listing 3.3:** AVX vectorized C++ implementation for eight values of $g\,(\bullet, \bullet, \bullet)$.

```
1  #include <immintrin.h>
2  inline __m256 calculate_g_vector(const __m256 llrs0,
3                                   const __m256 llrs1,
4                                   const __m256 bits)
```

```
{
  const __m256 sum = _mm256_add_ps(llrs1, llrs0);
  const __m256 dif = _mm256_sub_ps(llrs1, llrs0);
  return _mm256_blendv_ps(sum, dif, bits);
}
```

Similarly to (3.5), we can vectorize (3.7) with AVX intrinsics in Listing 3.3. We compute the result for both possibilities $\hat{u} = \{0, 1\}$ with element-wise addition with `_mm256_add_ps` and subtraction with `_mm256_sub_ps` and choose the appropriate result with `_mm256_blendv_ps` depending on the values in `bits`. Here, we assume that the sign bit of a float value carries the bit information in `bits`. The corresponding function for hard decision is shown in Listing 3.4 where positive or negative 0 is returned depending on the input. It should be noted that some compilers may remove the sign if very aggressive optimization options are active.

**Listing 3.4:** C++ implementation of $Q\{\ldots\}$.

```
inline float decide_hard(const float llr)
{
  return (llr < 0) ? -0.0f : 0.0f;
}
```

### Simplified successive cancellation

So far we discussed the originally proposed SC decoder that can be modified to the Simplified Successive Cancellation (SSC) decoder for lower latency and higher throughput [AYK11, Gia16, Sar16]. Particularly, we consider constituent codes of a particular polar code in Fig. 3.9. We transform the graph representation from Fig. 3.8 into a tree. At layer 0 in Fig. 3.9 we see all elements of a 16 bit polar code with frozen bit positions in white and information bit positions in black. With every layer, we can recursively combine nodes into one parent node and identify three patterns. These specializations do not impact error correction performance.

**Rate** $R$ nodes in gray are generic. These nodes are computed as described in Sec. 3.2.3.

**Rate** 0 nodes in white constitute sub-trees with only frozen bit positions. Thus, all values are known and we do not need to compute them any further.

**Rate** 1 nodes constitute sub-trees with information bits only. It is sufficient to decide all constituent bits at that layer and compute the encoder operation

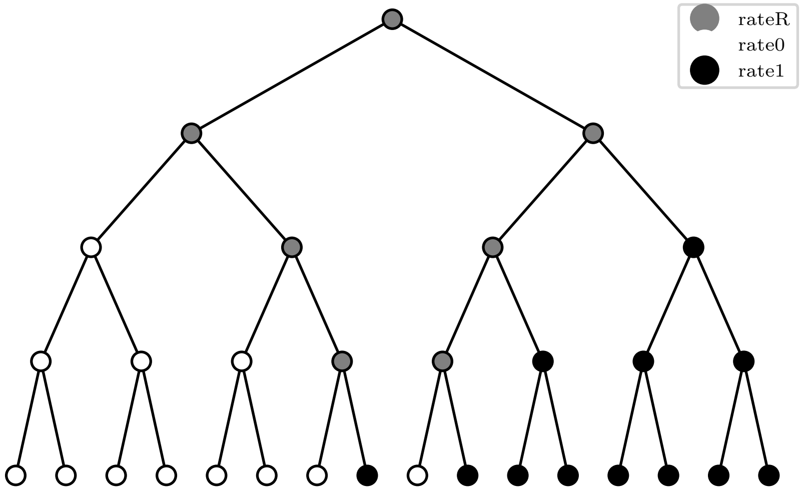


**Figure 3.9:** 16 bit SSC polar decoder tree

(3.3) on that constituent code and mind that $\boldsymbol{u} = \boldsymbol{uGG}$. This is a more lightweight operation than LLR computations.

## Fast simplified successive cancellation

The SSC decoder may be further optimized to the Fast-SSC decoder in order to improve throughput and reduce latency with more specialized constituent codes [Gia16, SGV+16, STG+16].

Again, we consider a code tree illustrated in Fig. 3.10.

**Repetition (REP)** nodes in green indicate a constituent code that is really a repetition code, i.e. all bits in a constituent code of size $N_{\mathrm{cc}} = 2^l$ are frozen bits except for the last one. Thus, it is sufficient to compute

$$\tilde{x}_{N_{\mathrm{cc}}-1} = \sum_{0}^{N_{\mathrm{cc}}-1} \tilde{x}_{l,i} \tag{3.10}$$

and decide $\hat{u}_{N_{\mathrm{cc}}-1} = Q\left\{\tilde{x}_{N_{\mathrm{cc}}-1}\right\}$. It is important to note that this summation is only valid under the assumption that frozen bits are always zero.

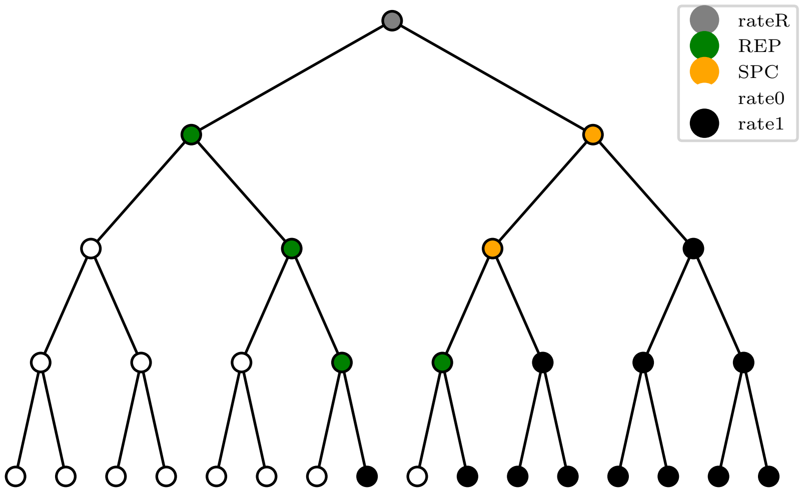


**Figure 3.10:** 16 bit Fast-SSC polar decoder tree

**Single-Parity-Check (SPC)** nodes in orange are processed such that we perform hard decision for all constituent LLRs $\tilde{x}_{l,i}$. Then, we compute

$$\text{parity} = \oplus_i^{N_{\text{cc}}} \tilde{x}_{l,i} \tag{3.11}$$

over this constituent code and, in case the parity constraint is not fulfilled, flip the bit that corresponds to the least reliable LLR.

**More specialized nodes** exist in literature [CBL18, HA17]. We use several of these, such as TypeI Double Repetition (DREP), TypeII Triple Repetition (TREP), TypeIII Double SPC (DSPC) and TypeV (TypeV) constituent codes. Further, for $N_{\text{c}} = 8$ constituent codes we use TypeIV Repetion One (REP-One) as well. In Appendix A we present a more complex Single Parity Check (SPC) node C++ implementation example.

### SC-List

While SC decoding yields excellent error-correction performance with large codewords, it falls short for moderate and short code lengths [Sar16]. Although several approaches to improve short code performance were proposed such as SCFlip [CSD18] and SCAN [FB14] decoders, we focus on Successive Cancellation List (SCL) decoding because it was recognized to yield the best error-correction performance [PCB21, SGV$^+$16, CSD16].

The authors in [TV15] originally introduced SCL decoding where they propose to follow both options whenever a bit decision is required under the SC strategy. Thus, whenever we would decode an information bit $\hat{u}_i$, instead we branch the decoder and follow both options, i.e. continue with the SC decoder process for $\hat{u}_i = 0$ and $\hat{u}_i = 1$. Without restricting the number of branches to follow, this yields a Maximum Likelihood (ML) decoder [TV15]. Although, unrestricted branching yields optimal performance, it does also yield high complexity. Therefore, we maintain a list of branches of size $L$ with the lowest metrics

$$\gamma_i = \begin{cases} \gamma_{i-1} + |\tilde{u}_i| & \text{if} \quad \operatorname{sgn}(\tilde{u}_i) \neq (-1)^{\hat{u}_i} \\ \gamma_{i-1} & \text{otherwise} \end{cases} \tag{3.12}$$

where $\gamma_0 = 0$, and prune all other branches [TV15, BSPB15]. This approach penalizes every branch that yields a decoded bit $\hat{u}_i$ that differs from the decoding decision a SC decoder would make. This penalty equals the absolute value of the LLR $|\tilde{u}_i|$ at the current position $i$. Importantly, this metric is updated for frozen as well as information bit positions to penalize probable incorrect prior decisions [BSPB15]. The branch list of size $L$ is maintained such that at every position where the SCL decoder branches into $2L$ branches, it will prune the $L$ branches with the highest metrics. Thus, the SCL decoder continues to decode $L$ branches. The list size $L$ steers the decoding complexity of the SCL decoder [TV15]. Finally, we select the surviving path with the lowest metric as the final result [TV15].

### CRC-Aided SC-List

The authors in [TV15] observed that the SCL decoder yields a final candidate list that often contains the correct information word. Still, the decoder may yield an incorrect information word because this word might have a lower metric than the correct information word. To overcome this issue the authors in [TV15] introduce a geany, mostly a short CRC, to identify the correct codeword in the candidate list. This leads to the CA-SCL decoder with state of the art error-correction performance. Since modern communication systems use a CRC for error detection [ETS20a], this strategy does not introduce additional processing steps. Suitable CRC polynomials for CA-SCL are known in literature [KC04].

However, a communication system that uses an error detection mechanism, e.g. a CRC, to detect errors may need to be adopted in case a CA-SCL decoder is employed. Effectively, $\log_2 L$ bit of an error detection checksum are used to select the correct information word from the candidate list

[KEXMH20]. In turn, this may increase the error detector False Alarm Rate (FAR) because the CRC is used for two different purposes [KEXMH20].

### 3.2.4 Puncturing

Polar code codeword sizes are inherently restricted to a power of 2 but we may circumvent this restriction with puncturing, shortening or repetition [CSD17, BGL17, ETS20a, KEXMH20]. Generally, puncturing is favorable for low code rates while shortening is superior for high code rates [BCL21]. In case the desired code length $N$ is just above a power of 2, repetition is a suitable choice to maintain error correction performance.

In the realm of URLLC, we are mostly interested in low rate codes and thus, we focus on puncturing strategies. We need to carefully choose a puncturing pattern otherwise code performance may be degraded dramatically [ZZW$^+$14]. Consider a non-systematic polar code in reversed order and puncture a code bit that is in $\mathcal{A}_{\mathrm{I}}$. This punctured bit results in a non-recoverable information bit and thus a catastrophic code.

A random puncturing strategy draws $P$ puncturing positions from $\mathcal{A}_{\mathrm{Fr}}$ at random. The authors in [NCL13] propose Quasi Uniform Puncturing (QUP) for non-systematic polar codes where puncturing positions are obtained by bit-reversing the first $P$ indices. In [ZZW$^+$14], the authors propose to choose $P$ puncturing positions that correspond to the least reliable bit positions, i.e. minimum puncturing.

In [DBD19], we investigate Frozen Quasi Uniform Puncturing (Frozen-QUP) to obtain a punctured codeword $\boldsymbol{c} \in \mathbb{F}_2^N$. First, we find a base codeword length $N_{\mathrm{c}} = 2^{\lceil \log_2 N \rceil}$ to a desired punctured codeword length $N$. Then, we puncture the base codeword $\boldsymbol{x}$ at the $P = N_{\mathrm{c}} - N$ positions and obtain a punctured codeword $\boldsymbol{c}$. Here, we choose the first $P$ bit positions from $\mathcal{A}_{\mathrm{Fr}}$ and apply bit-reversal according to the QUP strategy.

## 3.3 Integrated decoding and security

URLLC requirements focus on services with short packets and low latency requirements. Thus, it is unfeasible to concatenate packets for efficiency and error correction performance reasons. Packet overhead becomes a pronounced problem for short packets that tend to cause more overhead because metadata has a fixed size and is required for every packet. Furthermore, Factory Automation (FA) requires a high level of confidentiality to ensure production secrets do not leak to unauthorized third parties [MB17].

We filed a patent [DBD21] where we propose to merge the security and reliability domains to leverage synergies and to lower overhead. In Fig. 3.11

we illustrate the State-of-the-Art transmitter chain. It consists of two separate domains, namely security and reliability.

In the security domain, the two key targets are data confidentiality and authentication. Here, we assume every packet is first encrypted with the Advanced Encryption Standard (AES) to ensure confidentiality and requires 16 B packet size multiples and thus may add padding overhead [Dwo01]. Next, a cryptographic checksum, a MACode, and thus overhead is added to the encrypted packet to provide message authentication. We focus on CMAC for authentication [ISLP06, Dwo16]. Though, we note that other MACodes are available, e.g. HMAC would be a viable alternative [KBC97, NIS08].

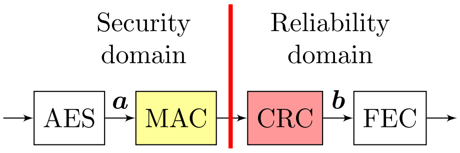
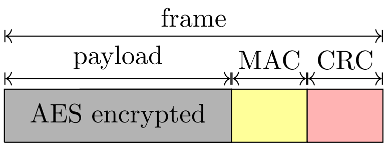


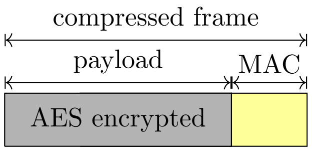
**Figure 3.11:** State-of-the-Art: separate security MAC and reliability FEC

In the reliability domain, the focus is on correct packet reception. The integrity of received packets is typically verified via CRC encoding with a high level of assurance [KC04, ETS20a]. A CRC adds a checksum, again overhead, to each packet for verification. Finally, we consider polar codes for FEC as presented in Sec. 3.2.

Now, from a processing perspective, it can be shown that both CRC and MACode add a checksum to each packet for integrity verification. The difference is that the MACode checksum supports additional functionality, namely authentication. Thus, our first idea is to drop the CRC and solely rely on the MACode as shown in Fig. 3.12b. The benefit is that we combine security and reliability and are able to reduce packet overhead.



**(a)** Traditional frame structure with payload and overhead (CRC and MACode)



**(b)** Compressed frame structure with payload and reduced overhead (CMAC only)

**Figure 3.12:** Frame structure comparison

In Sec. 3.2.3 we present a CA-SCL polar decoder that employs a CRC to identify the correct information word $\hat{\boldsymbol{a}}$ in a candidate list with $\hat{\boldsymbol{b}}_\ell$. We propose to use a MACode checksum instead to effectively merge the security and reliability domain as shown in Fig. 3.13. It is more efficient in terms of overhead because the CRC checksum is eliminated. For the small packet case with 128 bit packets and a typical CRC checksum size of 32 bit this results in a 25 % overhead reduction.

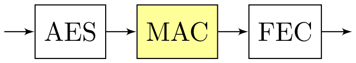


**Figure 3.13:** Key contribution: integrated security MACode and reliability FEC.

### 3.3.1 Contribution

The proposed system joins security and reliability aspects instead of treating them as separate entities in order to achieve a better wireless communication system as shown in Fig. 3.13. The resulting system is more efficient because overhead can be reduced while it maintains error correction performance as illustrated in Fig. 3.12. Specifically, we propose a joint MACode-FEC polar code for reduced overhead and a polar decoder that makes use of a MACode checksum.

## 3.4 Numerical software defined radio parameter evaluation

Naturally, FEC algorithms are rated by their error correction performance [CLT+21, CHL+19]. Though, this is not the only metric of interest. Achievable throughput and decoder latency often play a crucial role in the selection process for FEC algorithms. We started our FEC investigation in Sec. 3.1 with a comparison of State-of-the-Art codes and identified polar codes as a suitable candidate. Now, we want to present our findings for our polar code implementation [DL22] in terms of error correction performance, throughput and latency. We extend this investigation to different parametrizations to identify suitable trade-offs for URLLC.

### 3.4.1 Benchmarks

We want to gather performance metrics for our polar codes implementation that help to implement a SDR URLLC communication system. Sec. 8.4.4 details the general measurement approach and includes our definition of latency and throughput. Latency is defined as the time it takes to execute one function call. In this chapter, this is either the `encode` or `decode` function execution time. Moreover, the throughput is defined as the $\mathrm{bit\,s^{-1}}$ that are processed by the corresponding function. Coded throughput considers the $N_\mathrm{c}$ bit per function call while information throughput considers the $K$ bit per function call.

Others have published throughput and latency results for their implementations [Gia16, CHL+19], though these implementations are either not publicly available or not as fast as our implementation. The authors in [Gia16] report $398\,\mathrm{Mbit\,s^{-1}}$ to $502\,\mathrm{Mbit\,s^{-1}}$ throughput, and $2\,\mathrm{\mu s}$ to $3\,\mathrm{\mu s}$ latency for a $(2048, 1024)$ SC decoder on an Intel Core i7-4770S. However, to the best of our knowledge this implementation is not publicly available. The implementation presented in [CHL+19] was evaluated regarding throughput for an SC decoder in [LCL+19] where the authors report $215.15\,\mathrm{Mbit\,s^{-1}}$ decoder throughput for a $(2048, 1024)$ polar code. We are able to report higher throughputs for similar codes with $R = 0.5$ in Fig. 3.17. The focus of our work is on the impact of different parametrizations on error correction performance, throughput, and decoder latency for short packets.

It is possible to adopt inter-frame and intra-frame decoder strategies to boost throughput [GLJ15, Gra19]. We restrict our investigation to intra-frame decoder strategies because we focus on URLLC where the additional inter-frame latency is prohibitive. Further, we consider single-thread performance exclusively because a full SDR system needs to perform more tasks than coding which require compute resources as well.

### 3.4.2 SCL list size

In our discussion in Sec. 3.1, we showed that there exists a trade-off between error correction performance and complexity. In terms of complexity, we consider a SC polar decoder with complexity $\mathcal{O}\left(N_\mathrm{c} \log N_\mathrm{c}\right)$ and SCL decoders with $\mathcal{O}\left(L N_\mathrm{c} \log N_\mathrm{c}\right)$. Thus, keep in mind that complexity scales linearly with list size $L$.

#### Error correction performance

All our investigations are performed with our implementation [DL22]. We start our investigation with a comparison of different list sizes $L$ and channel construction strategies in Fig. 3.14 and Fig. 3.15 as discussed in Sec. 3.2.2 and 3.2.3. We use a 8 bit CRC in each case to enable CA-SCL where applicable. We observe an error correction performance boost for larger list sizes at the expense of increased complexity. While the error correction performance receives a significant boost for $L = 8$ compared to $L = 1$, further error

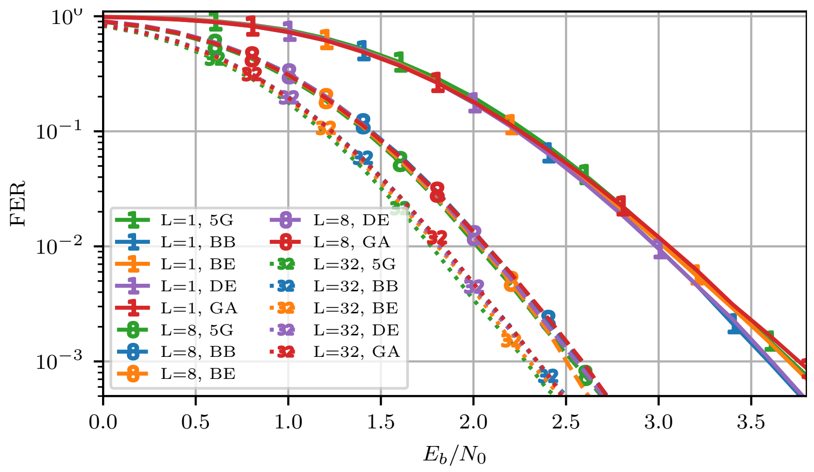


**Figure 3.14:** $(512, 256)$ Polar codes with varying list size $L$ and channel constructions (5G, BB, BE, DE, GA).

correction performance boosts are relatively small while complexity escalates. In accordance with 5G NR we will use $L = 8$ as an error performance benchmark and consider it a suitable trade-off between throughput and error correction performance [BCL21].

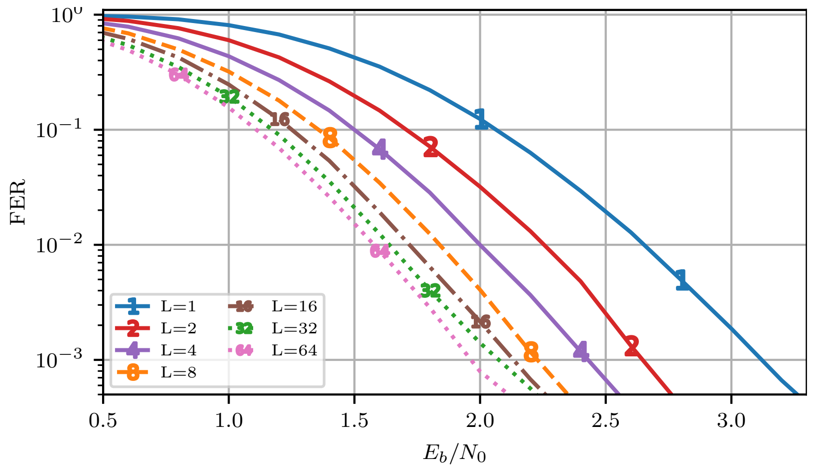


**Figure 3.15:** $(1024, 512)$ Polar codes with varying list sizes $L$.

We compare throughput for different list sizes $L$ in Fig. 3.16 on an AMD

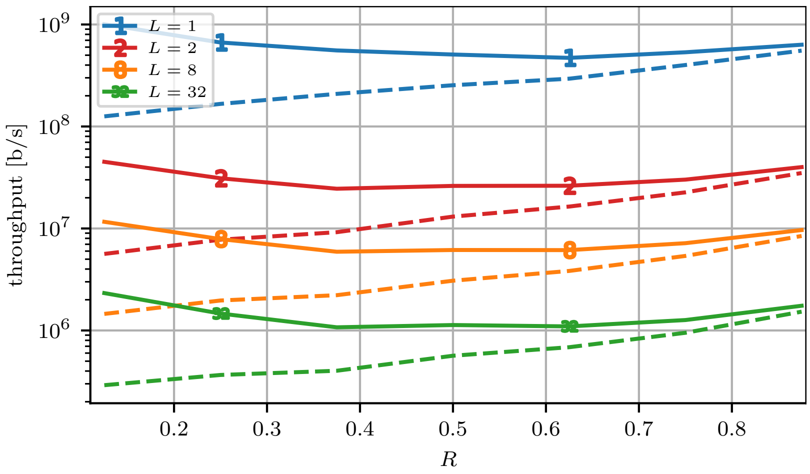


**Figure 3.16:** Throughput for $L \in \{1, 2, 8, 32\}$ polar codes with $N_{\mathrm{c}} = 512$, BB, dSNR $= 1.0$, without CRC, systematic code, and a fp-32 decoder implementation. (solid: coded, dashed: information)

Ryzen Threadripper 3970X (TRX3970X) machine as discussed in Sec. 8.6. Here, we consider no CRC, a block length $N_{\mathrm{c}} = 512$, channel construction with Bhattacharyya Bound (BB) and dSNR $= 1.0$ with a systematic floating point decoder implementation. We present results for both, coded bits throughput and information bits throughput because their relation varies for different coderates $R$. Note that throughput is plotted logarithmically to ensure all graphs are sensibly visible. A list size of $L = 2$ reduces throughput by a factor of 65. A further increase in list size from $L = 2$ to $L = 8$ reduces throughput by factor of four 4. Thus, a list size $L = 8$ reduces throughput by a factor of $> 200$. At first, we would expect throughput to decrease to an eighth of the SC decoder, corresponding to $L = 1$, due to the corresponding complexity increase with $\mathcal{O}\left(L N_{\mathrm{c}} \log N_{\mathrm{c}}\right)$ [TV15]. Instead, we observe a throughput decrease to two hundredth. We attribute this decrease to complexities with memory management, list sorting and metric calculation. Further investigation and optimization would be key to improve performance here. The step from $L = 8$ to $L = 32$ decreases throughput by a factor of four which corresponds to the complexity increase. We conclude that further investigation into inherent SCL decoder optimizations seems appealing. Also, we implement a mixed decoder as well. First, we employ the SC decoder and only use a CA-SCL decoder if the first CRC check fails. For practical points of operation, we expect most codewords to pass with the fast SC decoder and only very few codewords require a CA-SCL. We want to point out that the worst case decoding latency is considerably higher in this case, while throughput is significantly improved.

Further, we see lower throughput for codes around $R = 0.5$ while low and high rate codes deliver higher coded throughput. A SC decoder for a low rate code is able to decode more than $1\,\mathrm{Gbit\,s^{-1}}$ on a single core. Information throughput increases with code rate which is to be expected.

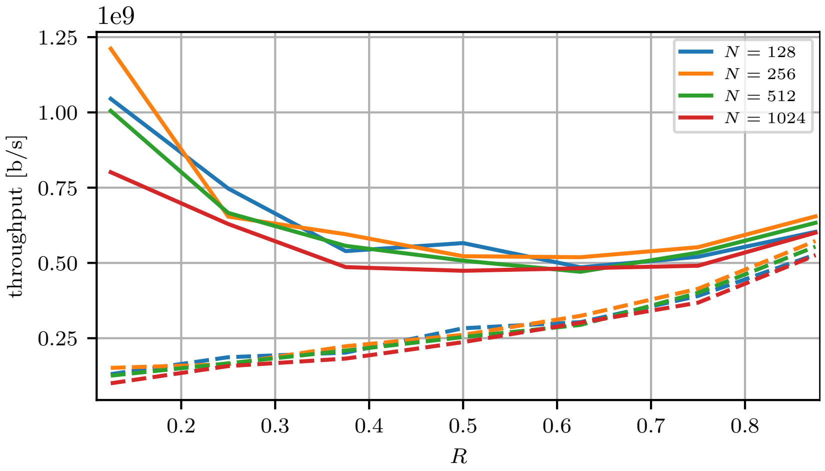
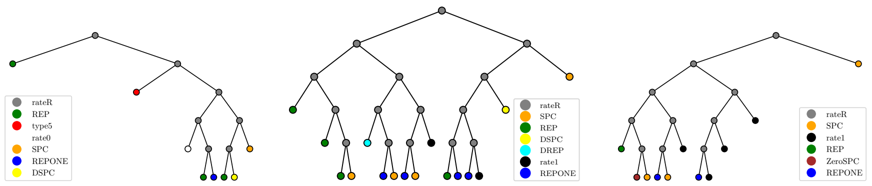


**Figure 3.17:** Throughput for $N_c \in \{128, 256, 512, 1024\}$ polar codes with $L = 1$. (solid: coded, dashed: information)

Now, we want to further investigate throughput for different polar codes in Fig. 3.17 with an SC decoder. Note that throughput is plotted on a linear scale. The information bit throughput increases with higher code rate. However, the coded throughput shows a minimum around $R \approx 0.6$. We reckon this dip is caused by smaller constituent codes. We corroborate our findings by considering pruned decoder trees in Fig. 3.18 and observe that high and low rate codes tend to show larger constituent codes that are better suited for optimization.

Previously, we noted that polar codes exhibit $\mathcal{O}\left(N_c \log N_c\right)$ complexity. Consequentially, we expect higher coded throughput for smaller block sizes $N_c$. On the contrary, $N_c = 256$ codes yield higher throughput than smaller $N_c = 128$ codes which in turn fall behind $N_c = 512$ codes at higher code rates. Particularly, we observe a peak coded throughput for $N_c = 256$ of $1.25\,\mathrm{Gbit\,s^{-1}}$ where the decoder hits a sweet-spot for vectorized optimizations.



**(a)** $R = 1/8$ **(b)** $R = 5/8$ **(c)** $R = 7/8$

**Figure 3.18:** Polar decoder trees for $N_c = 256$

## 3.4.3 Channel construction algorithm

Sec. 3.2.2 introduced multiple channel construction algorithms. Fig. 3.19 and Fig. 3.20 show the corresponding error correction performance evaluations.

### Error correction performance

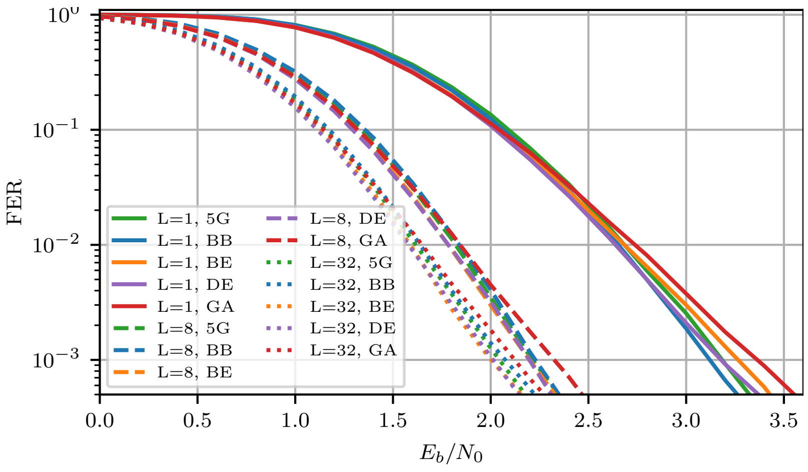


**Figure 3.19:** (1024, 512) Polar codes with varying channel construction algorithms with dSNR = 1 dB where applicable.

For $L = 1$ BB exhibits the best performance while larger $L$ favors 5G and $\beta$-Expansion (BE) in Fig. 3.19. Gaussian Approximation (GA) shows the worst performance among all algorithms. Though, we note that this channel construction algorithm only yields a minor performance degradation. Finally, Density Evolution (DE) performance is in between the other algorithms. DE tends to yield better performance in comparison for larger list sizes $L$.

While it might be suitable for small packets $N_\mathrm{c} \leq 1024$ to use 5G channel construction, we generally consider BB and BE due to their generality, good performance and simple construction algorithms.

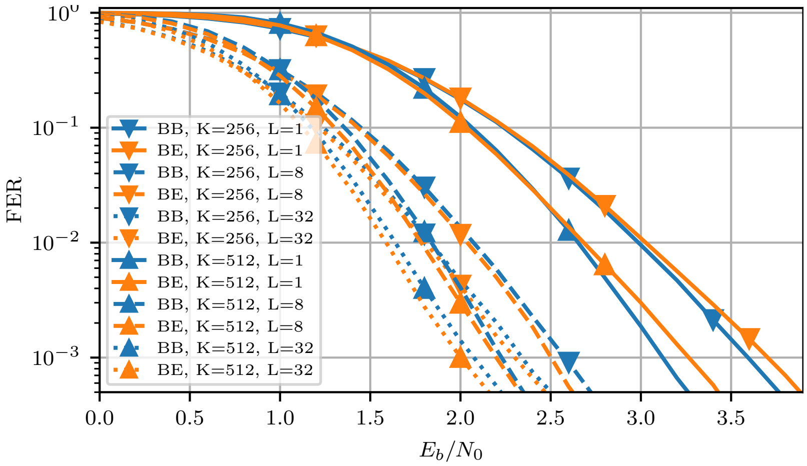


**Figure 3.20:** $R = 0.5$ polar codes with $N_\mathrm{c} \in \{1024, 512\}$

We want to corroborate our findings with the results in Fig. 3.20 for polar codes with code rate $R = 0.5$. We only consider BB and BE in this case for different list sizes. First, we observe that BB yields better performance for $L = 1$ while BE yields better performance for larger list sizes. A larger codeword size $N$ results in better error correction performance for all configurations as expected.

## Channel construction parameters

Different Design SNRs (dSNRs) yield varying polar code structures and in turn different error correction performance. Our encoder latency investigation reveals that latency increases linearly with $K$, while code structure has a negligible effect on latency. This might very well be a dominating factor for decoder latency. Thus, in Fig. 3.21 we see the corresponding decoder results. A first conclusion here would be to acknowledge that a lower dSNR yields lower latency. Second, we see a maximum latency for a code with rate $R = \frac{5}{8}$ which is an indication that specialized nodes, mostly Repetition (REP) and SPC, have a positive impact on latency. Considering polar encoder and decoder, it becomes obvious that channel construction for lower dSNR is preferable.

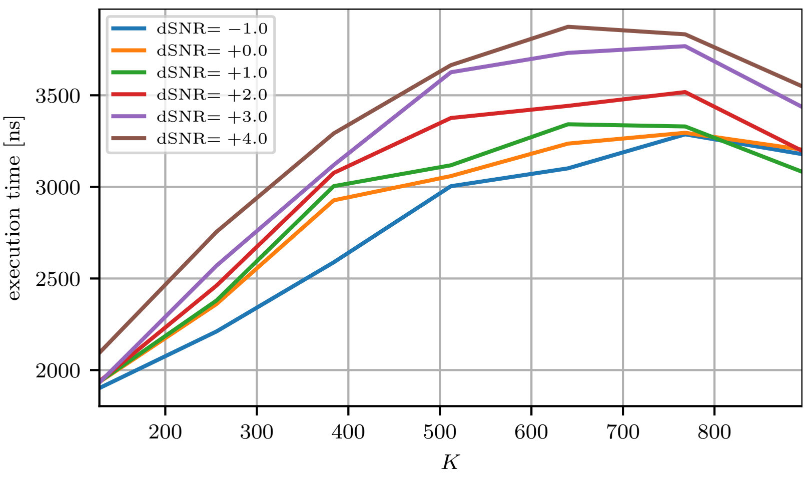


**Figure 3.21:** Polar decoder latency with varying dSNR and $N = 1024$, CRC32, $L = 1$, fp-32, non-systematic.

## 3.4.4 CRC aided polar codes and security

The CA-SCL decoders in Sec. 3.2.3 use a CRC to identify the correct codeword in a candidate list. Our proposal in Sec. 3.3 replaces the CRC with a CMAC to aid correct codeword identification. Here, we explore the benefits and drawbacks of these solutions. The encoder benchmarks highlight the impact of the proposed checksum algorithms on latency. Simulation results reveal the exact same error correction performance for CRC as well as CMAC checksums, and thus we focus on latency investigations.

In Fig. 3.22 we investigate the impact of different truncated CMAC lengths on latency. We use `openssl` to compute CMACs [Ope21]. We observe that the polar encoder latency is unaffected by the CMAC length. Since a truncated CMAC is used in each case that is derived from a 128 bit CMAC, this result is to be expected. Further, a longer input sequence does not affect latency which is probably due to constant-time execution requirements for security reasons in the `openssl` library.

Next, we compare 32 bit CRC and 32 bit CMAC based polar encoders. Here, we observe a constant offset of approximately 350 ns between both implementations. We conclude that a CRC implementation offers slightly lower latencies in case security features are not required.

Further, we consider different CRC lengths in Fig. 3.22. A length 0 CRC implies that we do not add a checksum at all. Thus, this implementation offers the lowest latency. For length 8, we use an optimized Look-Up-Table

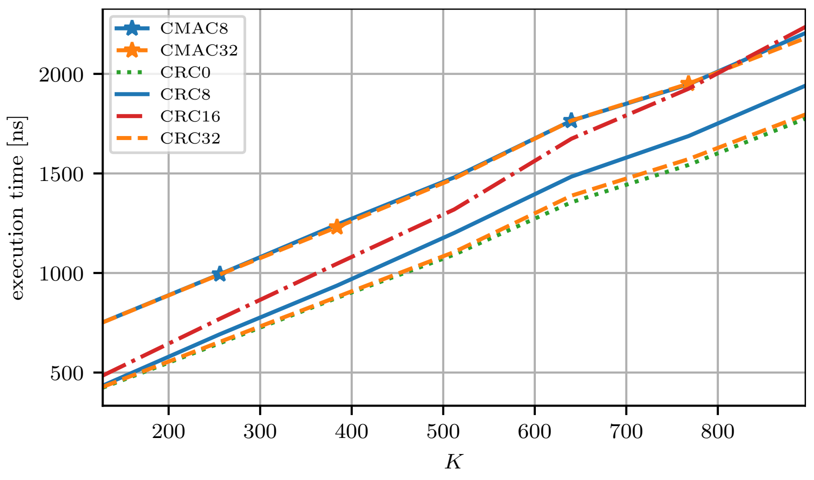


**Figure 3.22:** Polar encoder with CRC and CMAC checksum sizes $0, 8, 16, 32$, $N = 1024$, BB dSNR $= 1.0$, non-systematic.

(LUT) based implementation that increases latency slightly. The length 16 CRC is made available through the CRC++ library [Bah21]. In this case latency increases faster with larger $K$, probably due to the added flexibility offered by the chosen library. Finally, the length 32 CRC implementation is optimized to leverage SIMD instructions [Int21]. We observe that longer input increases latency only slightly in contrast to the plain case without CRC. Therefore, we can conclude that specialized SIMD instructions are useful to improve latency. Though, this optimization is restricted to one specific CRC polynomial.

## 3.4.5 Systematic versus non-systematic codes

Systematic polar codes exhibit better BER performance while their FER performance is unchanged [STG+16]. Here, we want to explore benchmark results for these codes.

A systematic polar encoder mostly requires a second encoder pass to yield a codeword. In Fig. 3.23 we observe that a systematic encoder requires approximately 180 ns more latency. This result indicates that encoder latency is dominated by information bit insertion, while the recursive encoder structure is well suited for low latency applications.

On the contrary, the polar decoder exhibits lower latency for systematic codes as shown in Fig. 3.24. This should come as a surprise but is well

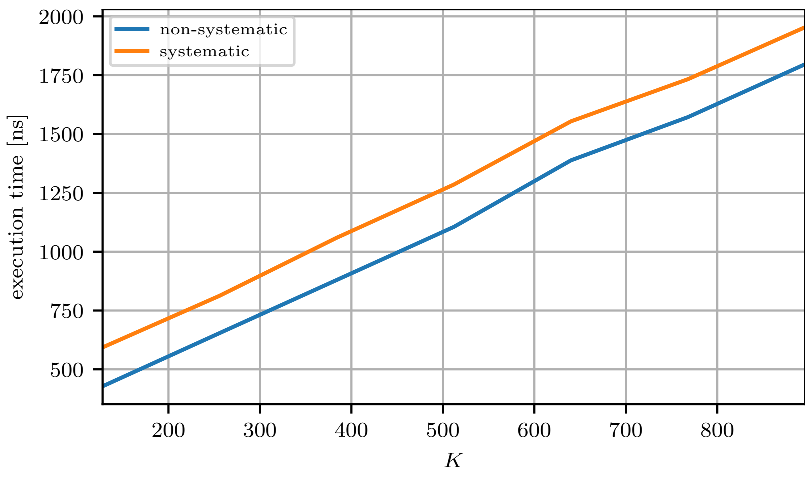


**Figure 3.23:** Polar encoder latency: systematic vs non-systematic, $N = 1024$, CRC32, BB dSNR = 1.0.

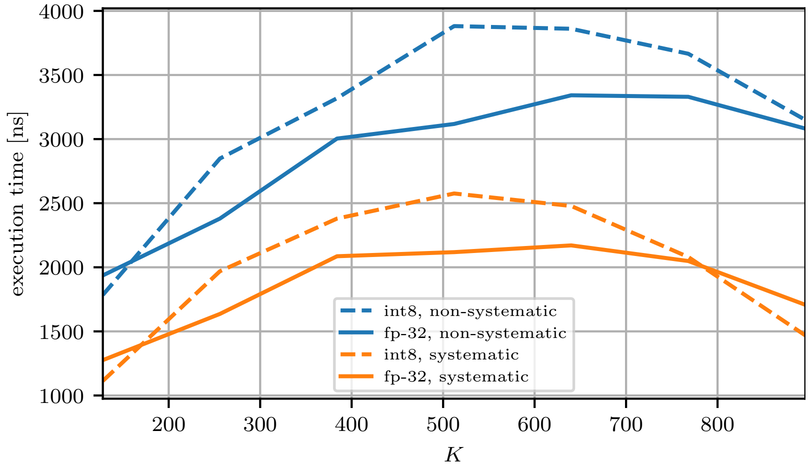


**Figure 3.24:** SC polar decoder latency: systematic vs non-systematic, and int8 vs fp-32, $N = 1024$, CRC32, BB dSNR = 1.0, and $L = 1$.

justified because the decoder implementation needs to compute most encoder operations anyways and thus yields a systematic codeword first that needs to go through another encoder pass to yield a non-systematic information word. The systematic 8bit integer (int8) fixed point decoder implementation yields a peak latency at 2575 ns while the fp-32 decoder shows a peak at 2118 ns. Overall, the polar decoders show a higher impact on latency then their encoder counterparts and thus, the configuration should be chosen such that decoder latency is minimized.

### 3.4.6 Punctured codes

Polar codeword sizes are inherently restricted to powers of two. In order to gain more flexibility, we investigate puncturing schemes for low rate codes as discussed in Sec. 3.2.4. The focus of this investigation is on error correction performance of punctured polar codes. Since we consider puncturing outside the scope of the actual encoder and decoder, we do not present benchmark results because these operations are readily integrable into adjacent operations, e.g. interleaving.

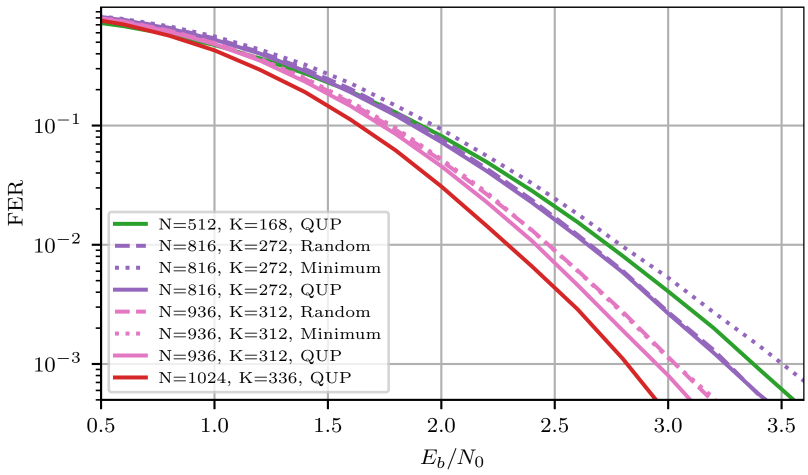


**Figure 3.25:** Puncturing scheme comparisons with $R \approx \frac{1}{3}$

Fig. 3.25 compares polar codes with $R \approx \frac{1}{3}$ with different codeword sizes $512 \leq N \leq 1024$. First off, the polar codes $(512, 168)$ and $(1024, 336)$ exhibit slightly lower coderate and do not require puncturing but serve as a reference. The $(936, 312)$ codes are all located within these bounds and reveal that Frozen-QUP delivers the highest error correction performance among the considered puncturing schemes. Puncturing out the positions with minimal reliability reveals the poorest performance in every configuration. With more punctured out bits, the Frozen-QUP performance moves closer to the random puncturing strategy. Still, we conclude that Frozen-QUP yields the lowest FER and thus should be favored over the other puncturing schemes.

### 3.4.7 Contribution

We investigate the error correction performance, latency and throughput of short polar codes with our latency-optimized software implementation [DL22]. These investigations reveal that low and high code rates are beneficial to throughput while a code rate $\approx 0.6$ yields the lowest throughput. Further, we investigate the trade-off between error correction performance and latency for CA-SCL decoders. The results show that a moderate list size of $L = 8$ is sufficient to collect most performance gains while still maintaining a reasonable latency. Finally, we conduct a thorough analysis of polar codes with differing parametrizations and discuss trade-offs.

## 3.5 Summary

We started this chapter with a review of modern coding algorithms for short codes. Subsequent sections introduce polar codes together with their most prominent advanced decoder option, the CA-SCL polar decoder. These discussions include algorithmic as well as software implementation specific considerations. Afterwards, we proposed a system that joins the security and reliability realms of a communication system in a joint MACode-FEC polar code for reduced overhead. Finally, an extensive set of experiments reveals the impact of different polar code parametrizations in terms of error correction as well as latency and throughput.

### 3.5.1 Contribution

In this chapter we verify the findings in [SWJ$^+$16] that polar codes, especially in conjunction with advanced decoders, are an attractive choice for short packet communication systems. Since most advanced polar decoders make use of a CRC, we propose to fuse the security and coding domains to leverage synergies that help to reduce frame overhead and thus boost efficiency especially for small packet communication. Specifically, we propose a joint MACode-FEC polar code for reduced overhead and a polar decoder that makes use of a MACode checksum. Moreover, we present our open source polar coding implementation [DL22]. Extensive simulation and benchmark measurement for short polar codes reveal trade-offs that may aid in designing communication systems with short packets, especially URLLC.

# Chapter 4

# Symbol mapping

In this chapter we discuss symbol mapping with Bit-Interleaved Coded Modulation (BICM) and their impact on latency. First, a discussion on all components and their theoretical function is conducted. This includes a discussion on specializations, approximations, and resulting optimizations to reduce the latency impact of these components. Previous works on soft demappers, [TB02, ALF04, MAXC16], focused on numerical performance, approximation quality, and complexity analysis. In this chapter, we contribute a thorough latency analysis of our open-source implementation [Dem22a] to determine their impact and readiness for future Cloud RAN systems.

We focus on symbol mappings that are standardized in current wireless communications systems such as LTE, 5G NR, and Wi-Fi. First, the process to map bits to complex symbols, i.e. symbol mapping, is discussed. It is accompanied by LLR calculation for reception as well as optimized approximations for standardized symbol constellations. These measures boost low latency but show a negligible impact on error rate performance. The theoretical part is completed with a discussion on BICM that ensures desirable statistical properties for FEC. Afterwards, we present our latency benchmark results for our open-source symbol mapping implementation [Dem22a] that we integrate into the GNU Radio API. The presented implementation can be used in simulations and SDR field tests. Field tests and simulations that share a common code base may drastically accelerate technology verification through synergies. Additionally, future Cloud RANs will benefit from a software implementation [BRW$^+$15] that will enable more efficient use of available hardware. Benchmarks reveal how susceptible the implemented functionality is to different parameter sets. Finally, these latency benchmarks provide reliable figures for low latency broadband SDR system design for I4.0 URLLC applications. In summary, the investigated software implementation for Cloud RANs exhibits suitable low latency for I4.0 URLLC applications Cloud RAN.

## 4.1 Fundamentals

Depending on the current channel state and reliability constraints, we may dynamically choose different mapping schemes to boost efficiency or reliability [CTB98]. Our considerations are restricted to QAM mappings because these are the only relevant options in multicarrier contexts [ETS18a, IEE12].

We present a flowgraph in Fig. 4.1 that focuses on mapping details. Correspondingly, in Fig. 2.2 we present a system overview. A codeword $\boldsymbol{c}$ with $N$ elements, as discussed in Chapter 3, is first interleaved into an interleaved codeword $\boldsymbol{c}_{\mathrm{IL}} \in \mathbb{F}_2^N$. Bit interleaving is employed to ensure bit-layers and code bits are sufficiently independent and thus boost FEC decoder performance [CTB98]. Interleaving and rate matching, discussed in Sec. 3.2.4, both require selecting specific elements and moving them to new positions, which may be combined for efficiency reasons. Here, we require that $N = MN_{\mathrm{d}}$ holds after puncturing. The interleaved codeword $\boldsymbol{c}_{\mathrm{IL}}$ is partitioned into groups of $M$ elements and mapped to a complex symbol from the current QAM constellation $\mathcal{A}_{\mathrm{C}}$ with $|\mathcal{A}_{\mathrm{C}}| = 2^M$ that we discuss in Sec. 4.2 with an example in Fig. 4.2. Thus, $M$ represents the mapping or constellation order, while $2^M$ represents the mapping or constellation size. Furthermore, the interleaved codeword $\boldsymbol{c}_{\mathrm{IL}}$ of size $N$ is mapped onto the complex symbol vector $\boldsymbol{d} \in {\mathcal{A}_{\mathrm{C}}}^{N_{\mathrm{d}}}$ with $N_{\mathrm{d}}$ elements.

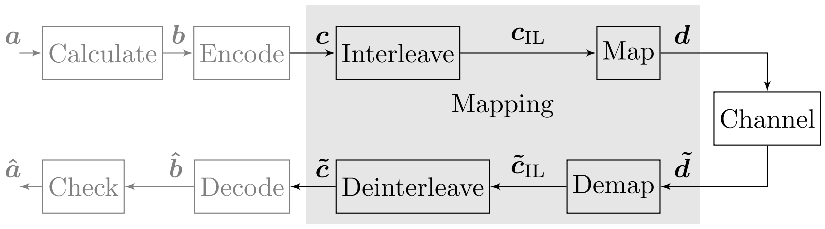


**Figure 4.1:** Mapping flowgraph with $\boldsymbol{c} \in \mathbb{F}_2^N$, $\boldsymbol{c}_{\mathrm{IL}} \in \mathbb{F}_2^N$, $\boldsymbol{d} \in {\mathcal{A}_{\mathrm{C}}}^{N_{\mathrm{d}}}$, and $N = MN_{\mathrm{d}}$.

At the receiver, the demapper yields LLRs $\tilde{\boldsymbol{c}}_{\mathrm{IL}}$ from the received complex symbols $\tilde{\boldsymbol{d}}$. These LLRs are then deinterleaved before they are fed into a FEC decoder.

## 4.2 Quadrature amplitude modulation

We consider multiple constellations with their corresponding alphabets $\mathcal{A}_\mathrm{C}$ where $|\mathcal{A}_\mathrm{C}| = 2^M$ is the number of constellation points. Many communication standards, e.g. Wi-Fi, LTE, 5G NR, share a common set of constellations and corresponding bit-to-symbol mapping [IEE12, ETS19b, ETS19a]. A vector $\boldsymbol{c} \in \mathbb{F}_2^{MN_\mathrm{d}}$ is rearranged into $N_\mathrm{d}$ groups

$$\boldsymbol{c} = [c_0, c_1, \ldots, c_{Mi+l}, \ldots, c_{MN_\mathrm{d}-2}, c_{MN_\mathrm{d}-1}] \tag{4.1}$$

where each group consists of $M$ bits. Each group is mapped to a constellation point

$$d_i = \mathcal{M}\left([c_{Mi+0}, \ldots, c_{Mi+l}, \ldots, c_{Mi+M-1}]\right) \tag{4.2}$$

where $l = \{0, \ldots, M-1\}$ indicates the position within a group and $i = \{0, \ldots, N_\mathrm{d}-1\}$ indicates the index of the corresponding complex symbol. We assume Gray labeling for all constellations, i.e. neighboring constellation point labels only differ by one bit [CTB98, Boc12]. Generally, we assume that the mean complex symbol power $\sigma_d^2 = 1$.

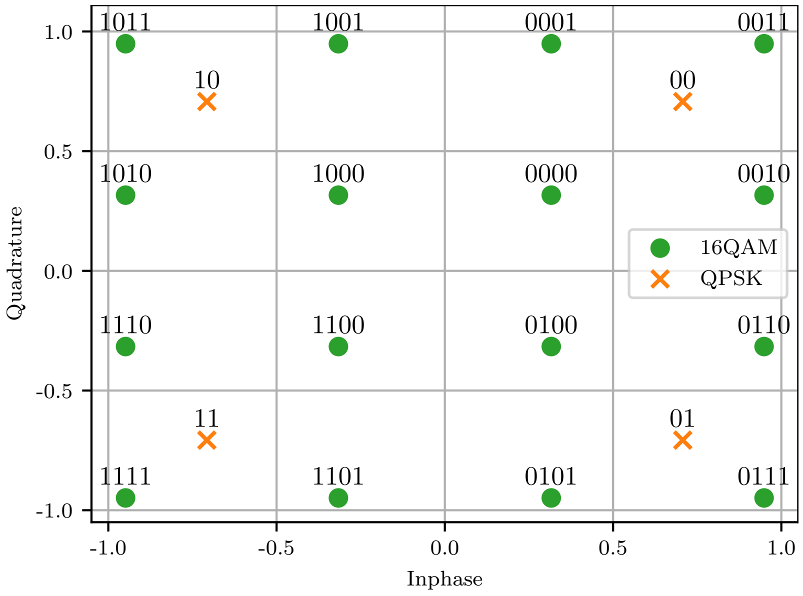


**Figure 4.2:** LTE/5G NR/Wi-Fi Quadrature Phase Shift Keying (QPSK) and 16QAM constellations with corresponding bit to symbol mappings

Fig. 4.2 shows two exemplary constellations, namely QPSK and 16QAM, together with their bit labels. In our work, we mostly consider lower order constellations, e.g. Binary Phase Shift Keying (BPSK), QPSK or 16QAM, but note that higher order constellations, e.g. 64QAM and 256QAM, are commonly used as well and available in our implementation [ETS18a].

## 4.3 Soft demapping

While mapping with (4.2) boils down to a simple Look-Up-Table (LUT), demapping is more complex. A soft demapper yields soft bit values, or LLRs, $\tilde{\boldsymbol{c}}$ that are fed into a FEC decoder as discussed in Sec. 3.2.3. Soft demapping yields a significant performance gain compared to the hard decision approach and is thus the preferred method in most cases [Pro95]. We want to leverage corresponding error correction performance gains of soft demapping over hard demapping with hard decision values $\hat{\boldsymbol{c}}$. We assume that the sign bit indicates the equivalent hard decision bit $\hat{c}$ and the absolute value is a measure of reliability, i.e. we assume $\tilde{c} > 0$ if $\hat{c} = 0$ and $\tilde{c} \leq 0$ if $\hat{c} = 1$ [TB02, ALF04, MAXC16]. In general we obtain hard decisions

$$\hat{d} = \mathcal{Q}_{\mathcal{A}_\mathrm{C}}\left(\tilde{d}\right) = \underset{d \in \mathcal{A}_\mathrm{C}}{\operatorname{argmin}} \left|d - \tilde{d}\right|^2 \tag{4.3}$$

from received symbols $\tilde{d}$ by finding the transmit symbol $d \in \mathcal{A}_\mathrm{C}$ with the smallest Euclidean distance.

We recall our LLR definition from (3.4) for BPSK or NRZ mapped bits

$$\tilde{c} = \ln \frac{p\left(c = 0|\tilde{d}\right)}{p\left(c = 1|\tilde{d}\right)} = \ln \frac{p\left(\tilde{d}|c = 0\right)}{p\left(\tilde{d}|c = 1\right)} \tag{4.4}$$

with the Bayes' theorem and under the assumption of equiprobable bits. Now, we extend this LLR definition for mappings with higher mapping order $M$. Here, we distinguish two distinct sets ${\mathcal{A}_\mathrm{C}}_l^{(0)}$ and ${\mathcal{A}_\mathrm{C}}_l^{(1)}$. Specifically, the set ${\mathcal{A}_\mathrm{C}}_l^{(0)}$ contains all constellation points $d$ where $c_l = 0$ holds for the bit at position $l$, while ${\mathcal{A}_\mathrm{C}}_l^{(1)}$ holds the corresponding constellation points $d$ where $c_l = 1$ holds. For $M > 1$, we sum up all likelihoods for ${\mathcal{A}_\mathrm{C}}_l^{(0)}$ and ${\mathcal{A}_\mathrm{C}}_l^{(1)}$ separately and obtain

$$\tilde{c}_l = \ln \frac{\sum\limits_{\forall d \in {\mathcal{A}_\mathrm{C}}_l^{(0)}} p\left(\tilde{d}|d\right)}{\sum\limits_{\forall d \in {\mathcal{A}_\mathrm{C}}_l^{(1)}} p\left(\tilde{d}|d\right)} \tag{4.5}$$

where we compute the ratio between both sums. Finally, we obtain an efficient approximate equation

$$\tilde{c}_l \approx \frac{1}{\sigma_{\mathrm{n}}^2} \left( \min_{d \in \mathcal{A}_{\mathrm{C}_l}^{(0)}} |\tilde{d} - d|^2 - \min_{d \in \mathcal{A}_{\mathrm{C}_l}^{(1)}} |\tilde{d} - d|^2 \right) \tag{4.6}$$

under the AWGN channel assumption. All further details are discussed in Appendix C along with specialized and optimized implementations for specific constellations [TB02, ALF04, MAXC16].

## 4.4 Bit-interleaved coded modulation

A mapped bit $c_l$ at bit position $l$ may experience a different reliability then a bit at another bit position on the same complex symbol. Furthermore, bits that are mapped to symbols on neighboring subcarriers in a multicarrier system may experience correlated channels. In contrast we assume i.i.d. bits for FEC decoding, hence we need to ensure that this assumption is fulfilled to the best extend possible. The authors in [CTB98] propose Bit-Interleaved Coded Modulation (BICM) to ensure this assumption holds sufficiently well with the help of an interleaver before bits are mapped to symbols.

An interleaver is defined by a permutation pattern $\sigma(i)$, known to both transmitter and receiver, that re-arranges $\boldsymbol{c}$ into

$$\boldsymbol{c}_{\mathrm{IL}} = \left[c_{\sigma(0)}, c_{\sigma(1)}, \ldots, c_{\sigma(i)}, \ldots, c_{\sigma(N-1)}\right]. \tag{4.7}$$

The value at position $i$ in $\boldsymbol{c}_{\mathrm{IL}}$ equals the value at position $\sigma(i)$ in $\boldsymbol{c}$. In practice this permutation pattern is represented by a list of integers. In most cases, we employ a standard random interleaver [CTB98]. Other interleaver designs are available as well such as 5G NR interleavers [ETS20b]. Conceptually these interleavers insert values row-wise into a matrix and gather them column-wise or vice versa. We refer to these interleavers as block interleavers because they operate on blocks of fixed size. Often, columns are permuted before the interleaver gathers values from this matrix.

## 4.5 Mutual information

Before we consider practical and achievable performance for standardized constellations in simulations, we investigate its theoretical properties. The mutual information $I_{\mathrm{BICM}}$ for BICM is a measure of channel quality and required for Link Abstraction (LA) in Sec. 7.2 as well. We would like to remind the reader of the set of assumptions that are commonly made, specifically, we assume BICM, Gray labeling, and i.i.d. real and imaginary parts. The goal is to compute bit capacity for each layer depending on SNR. The mutual information $I_{\text{BICM}}$ calculation is split into $M$ different bit layers $l$ and these bit layers differ in reliability [Boc12]. For each bit layer, we compute

$$I_{\text{BICM},l} = 1 - E\left\{\log_2 \frac{\sum\limits_{d_2 \in \mathcal{A}_{\text{C}}} p_{\tilde{\text{d}}|\text{d}}\left(\tilde{d}|d_2\right)}{\sum\limits_{d_3 \in \mathcal{A}_{\text{C}}{}_l^b} p_{\tilde{\text{d}}|\text{d}}\left(\tilde{d}|d_3\right)}\right\} \tag{4.8}$$

and then calculate $I_{\text{BICM}} = \sum_{l=0}^{M-1} I_{\text{BICM},l}$ for the desired SNR. The interested reader may refer to Appendix B for further details how to solve this equation numerically via $N_{\text{GHQ}}$ point Gauss-Hermite-Quadrature [Jäc05, Boc12].

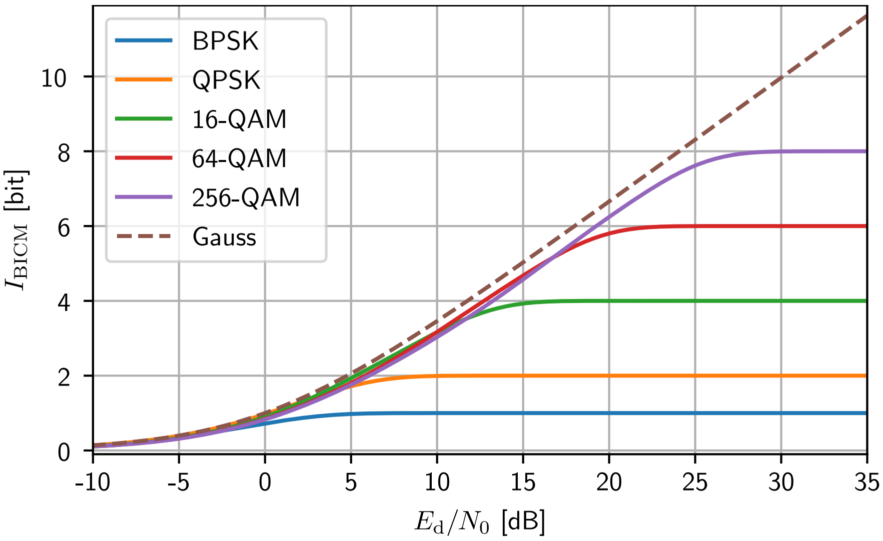


**Figure 4.3:** Mutual information for multiple constellations

Results for different mappings are shown in Fig. 4.3 together with Gaussian capacity. Computing these values is potentially a resource intensive task, thus we will use cached values in conjunction with linear interpolation. Further, Fig. 4.3 illustrates the need to dynamically choose different constellations depending on channel conditions. A 16QAM constellation should be favored over a 64QAM constellation for situation where SNR is below 10 dB but the same choice would be wasteful at 15 dB where 64QAM would be more efficient.

## 4.6 Numerical software defined radio parameter evaluation

In this section we benchmark the latency and throughput impact of BICM, i.e. interleaver, mapping, and demapping, on a SDR system. We consider a AMD Ryzen Threadripper 3970X (TRX3970X) Linux host as discussed in Sec. 8.6 and 8.4.1 with GNU Compiler Collection (GCC) 9.3 and optimization level `-O3`. Our latency benchmarks are performed according to the procedure discussed in Sec. 8.4.4.

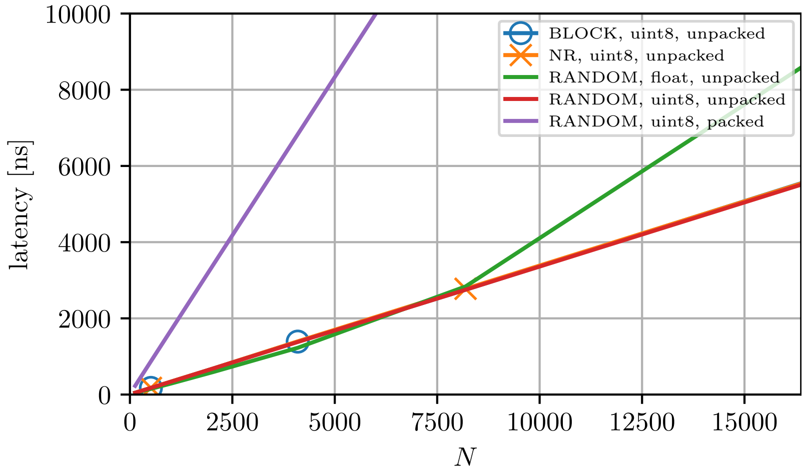


**Figure 4.4:** Interleaver latency benchmarks

The reported interleaver latencies in Fig. 4.4 indicate that latency increases linearly with block size $N$. We observe that all interleaver permutation patterns, namely RANDOM, NR, and BLOCK, yield the same latency results for unpacked bits of data type `uint8`. For $N > 8192$ the `float` interleaver latencies rise faster than the latencies for the `uint8` interleavers. Since `float` values require 4 B per value and $4\,\mathrm{B} \cdot 8192 = 32\,768\,\mathrm{B}$, larger $N$ values require memory in excess to the 32 768 B level 1 cache of the TRX3970X. This result exemplifies that it is important to consider CPU memory and caches on the host system during software development [HP14].

If we exceed the available memory at a given cache level, we must expect a steeper ascend in latency.

There is a trade-off in an implementation between *packed* and *unpacked* bits. If all 8 bit in a byte carry information, we consider the corresponding array to be *packed*. In contrast if only the Least Significant Bit (LSB) carries information the corresponding array is *unpacked*. A *packed* implementation incurs extra overhead if individual bits need to be extracted and inserted while it carries the advantage that it requires less memory. In Fig. 4.4 we observe that the *packed* interleaver introduces higher latencies because it needs to unpack, interleave, and pack every codeword. The advantage of *packed* interleavers, less memory usage, is outweighed by its disadvantage, more complexity and slower execution. We observed the same issue for packed bits and mappers and thus decided to stick to an unpacked design at this stage.

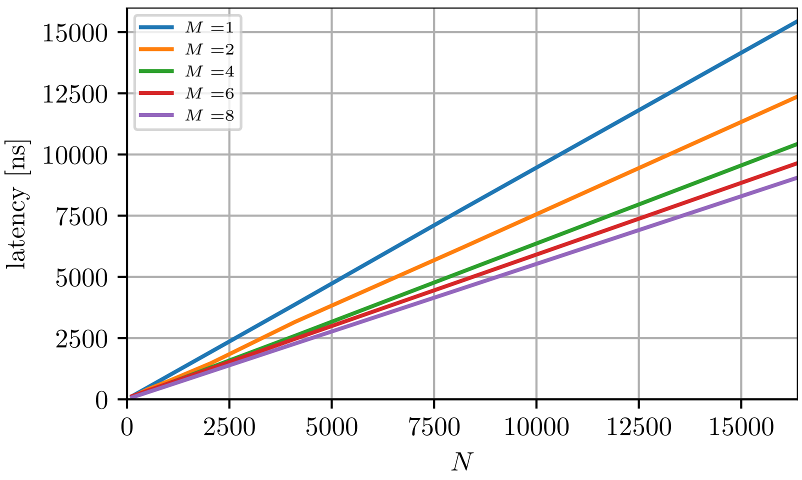


**Figure 4.5:** Mapping latency benchmarks with $N = MN_\mathrm{d}$ bits for various mapping orders $M$.

The mapping benchmarks in Fig. 4.5 reveal that higher mapping orders $M$ introduce lower latency for the same number of bits $N$. The mapping function needs to execute more instructions per produced complex symbol in case $M$ is larger, thus one would expect higher latencies for higher values of $M$. To the contrary, a higher mapping order yields lower latencies for the same number of bits $N$. Again, this result highlights the influence of CPU memory access because a higher modulation order $M$ results in a smaller $N_\mathrm{d}$ and thus, fewer values to write.

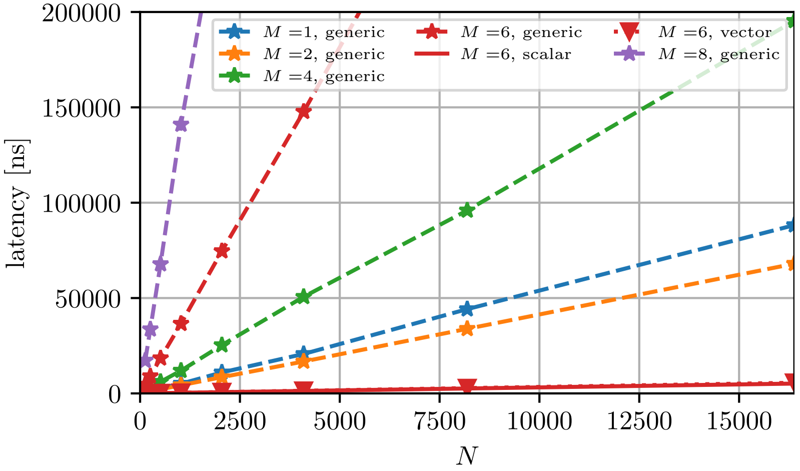


**Figure 4.6:** Generic demapper latency benchmarks. The highest latency specialized (`scalar`, `vector`) $M = 6$ demappers serve as a reference to Fig. 4.7.

We evaluate demapper latencies with the generic equation (4.6) in Fig. 4.6. First, we conclude that a higher mapping order $M$ results in exponentially higher latencies because the complexity scales with $\mathcal{O}\left(2^M\right)$ [MAXC16]. Every received symbol must be compared to every possible transmit symbol. Further, we observe that $M = 1$ exhibits higher latency than $M = 2$ which we attribute to CPU cache access again.

At the very bottom, we observe the latency results for $M = 6$ with optimized `scalar` and `vector` implementations. The $M = 6$ `scalar` and `vector` implementations received latency optimizations for this specific configuration while the `generic` implementation can be used for any mapping order $N_\mathrm{d}$ configuration. In case of AWGN channels it is sufficient to use one scalar SNR value to demap all symbols and hence, we call it a `scalar` implementation. In order to obtain accurate LLRs in multipath Rayleigh fading channels, especially in multicarrier systems where each subcarrier may be subject to a different CNR, we use a CNR vector to scale LLRs for every received symbol individually. Thus, we call this a `vector` implementation. Both versions, `scalar` and `vector`, use a specialized implementation that is optimized for $M = 6$.

Fig. 4.7 presents the results for specialized demappers as detailed in Appendix C and, for comparison, the `generic` $M = 2$ latency results. Even the fastest `generic` implementation yields higher latencies than all

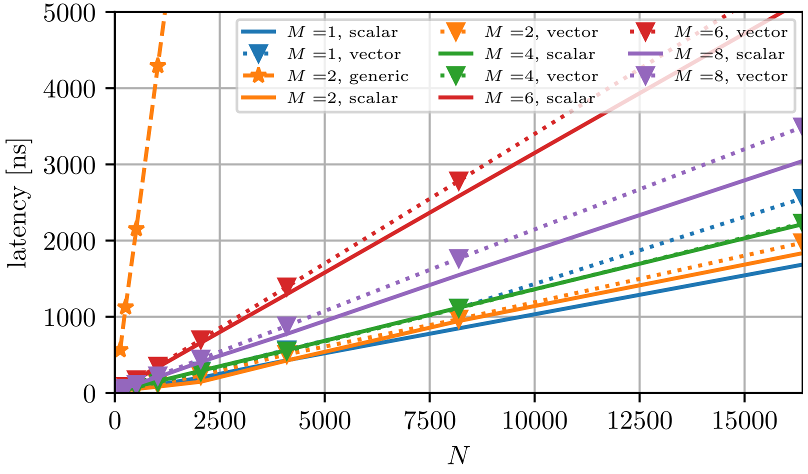


**Figure 4.7:** Specialized demapper latency benchmarks. The lowest latency `generic` demapper with mapping order $M = 2$ serves as a reference to Fig. 4.6.

specialized implementations. For $N < 2500$, and thus for all $N$ that we expect in a URLLC scenario, we observe a latency of less than 1 µs, even for $M = 6$. Intuitively, we would expect that a higher mapping order $M$ yields higher latencies. However, we notice that $M = 8$ yields lower latencies than $M = 6$ but we do not investigate this outlier any further. The `vector` implementations all exhibit slightly higher latencies than their `scalar` counterparts that we attribute to caching.

## 4.7 Summary

In this chapter, we introduced BICM which consists of interleaving, deinterleaving, symbol mapping, and symbol demapping. The discussion started with the theory behind these signal processing steps, and focused on standardized symbol mappings with specially optimized implementations afterwards. Further, known optimizations and approximations were discussed that lower latency drastically while their FER performance impact is known to be small and negligible [TB02, MAXC16]. Finally, latency benchmarks reveal the influence of different optimizations and parametrizations on the system latency.

### 4.7.1 Contribution

We presented our open-source symbol mapping and interleaving implementation that is integrated into a GNU Radio Out-Of-Tree (OOT) [Dem22a]. Besides a theoretical description and optimizations, the in-depth software implementation analysis reveals the latency impact that these signal processing steps exhibit on an overall SDR system. Since the actual processing steps for both, transmitter and receiver, are lightweight, our results indicate how further CPU features, such as caches, influence latency and throughput as discussed in Fig. 4.5. Previous works on soft demappers, [TB02, ALF04, MAXC16], focused on numerical performance, approximation quality, and complexity analysis. Here, we present measurements that reveal the latency impact of these optimizations and approximations to enable future Cloud RAN implementations.

# Chapter 5

# Multicarrier modulation

Most modern communication standards use multicarrier modulation, e.g. Long Term Evolution (LTE), 5th Generation New Radio (5G NR), and IEEE 802.11 (Wi-Fi). While most standards use Orthogonal Frequency Division Multiplexing (OFDM), we consider Generalized Frequency Division Multiplexing (GFDM) as well. In this chapter, we investigate how suited our multicarrier modulation software implementation for GFDM [DRKK22], as well as OFDM, is for Software-Defined Radio (SDR).

The chapter commences with a discussion on GFDM and OFDM multicarrier modulation with its specifics to arrive at optimized implementations. Next, an extensive set of benchmarks of our open-source, modular, and portable GFDM implementation in GNU Radio reveal how susceptible various of its components are to specific parameters and their impact on latency. We compliment these investigations with performance simulations for GFDM and OFDM to enable educated decisions on trade-offs between performance and latency. Finally, we show that low latency broadband SDR systems for Industry 4.0 (I4.0) Ultra Reliable Low Latency Communication (URLLC) applications are feasible.

The main contribution of this chapter is an investigation of latencies and performance implications introduced in the multicarrier modulation Digital Signal Processing (DSP) chain of our GFDM SDR implementation [DRKK22]. To this end, the chapter incorporates an extension to our prior works [DBD17a, DBD$^+$17b]. An extensive set of benchmarks and simulations demonstrates the practicability of our solution in the context of URLLC and I4.0 low-latency applications. In contrast, other works such as [DMG$^+$15] focus on hardware implementations without a focus on latency measurements. We identified the multicarrier modulation and demodulation steps to be critical while all other DSP operations have a minor latency impact. Prior works [FJ05] focus on $2^{\times}$ Fast Fourier Transform (FFT) transforms, while we analyze the impact of more primes on latency. Furthermore, this analysis shows that the number of timeslots in a GFDM system should be kept to a power of two contrary to the number of subcarriers where the impact on latency is lighter. Moreover, GFDM in conjunction with Forward Error Correction (FEC) does not benefit from more than two Interference-Cancellation (IC) iterations at high effective rates $R_{\text{eff}}$. At a low effective rate, a GFDM system does not benefit from IC at all.

**Multicarrier requirements** Current 4G LTE, Wi-Fi, Digital Audio Broadcasting (DAB), Digital Video Broadcasting (DVB) as well as 5G NR systems rely on OFDM which is a simple and effective multicarrier modulation scheme. However, several shortcomings with regards to OFDM were identified [CSW14]. These shortcomings include high Out-Of-Band (OOB) emission properties, strict synchronization requirements and low spectral efficiency due to Cyclic Prefix (CP) requirements. Furthermore, low spectral efficiency implies more overhead that contributes to higher latency, e.g. with more guard intervals.

A multicarrier communication system for I4.0 URLLC is expected to better suit these requirements than SotA systems. This includes operation in harsh, densely populated environments such as production floors with multiple communication systems in close spatial as well as spectral proximity [OMM16]. Thus, I4.0 systems must coexist with each other and with legacy systems. Further, these multicarrier systems are required to provide more flexibility to adapt to fading and counter deep fades in industrial radio channels [DHC+19].

**Multicarrier options** To overcome OFDM shortcomings, several multicarrier modulation scheme candidates for I4.0 exist which offer different approaches [SGA14]. Filter-Bank Multi-Carrier (FBMC) minimizes OOB emissions by filtering each subcarrier but introduces large filter delays [CSW14]. Universal Filterbank Multi-Carrier (UFMC) groups multiple subcarriers and then filters each group jointly in order to decrease filter delays, though it still only considers timeslots individually [VWS+13].

GFDM goes beyond symbol-based modulation by modulating entire frames [MMG+14]. Generally, GFDM is a highly flexible non-orthogonal waveform that promises to overcome the OFDM shortcomings, e.g. with low OOB emissions. OFDM can be considered a special parametrization of GFDM. Circular filters retain the option to use a CP and a Cyclic Suffix (CS) for whole frames. Furthermore, these filters avoid large delays and are able to minimize OOB emissions. For I4.0 low-latency communication systems, short filter delays are an important advantage because latency reduction is a critical design criteria which GFDM can deliver.

**Prior works on DSP latency** In addition to over-the-air latencies, signal processing adds significant delays to a communication system. The introduced latencies need to be known in order to be able to design a reliable low-latency I4.0 system.

In [DMG$^+$15], the authors present a GFDM implementation that specifically targets a hardware implementation in the Field Programmable Gate Array (FPGA) on an Ettus USRP X310 (X310). Though, the processing latencies of this FPGA implementation are unknown to the best of our knowledge. Moreover, we focus on SDR, and thus more flexible software implementations, in contrast to FPGA targets as discussed in Sec. 8.3.

We conducted latency investigations in two prior publications and want to extend on them [DBD17a, DBD$^+$17b]. This includes new benchmarks on newer hardware, a larger variable space, and performance versus latency trade-off discussions. At the time of this writing, there are no other software latency measurements of GFDM systems known to the authors.

**Software defined radio principles** Field tests and simulations which share a common code base drastically improve technology verification. A SDR implementation offers the advantage to combine these features and it can reveal latency figures in a GFDM system for I4.0. Additionally, future Cloud Radio Access Networks (Cloud RANs) will benefit from a software implementation [BRW$^+$15] which will enable more efficient use of available Commercial-Of-The-Shelf (COTS) hardware.

## 5.1 Fundamentals

We want to develop a common notation for both, GFDM and OFDM, before we focus on specifics. The flowgraph in Fig. 5.1 expects complex vectors $\boldsymbol{d} \in \mathbb{C}^{N_\mathrm{d}}$ from a symbol mapper as discussed in Sec. 4.1. Multicarrier modulation starts with resource mapping where elements from $\boldsymbol{d}$ are assigned to positions on a resource grid, or time-frequency plane, that is discussed in Sec. 5.2. We continue with multicarrier modulation in Sec. 5.3.

Eventually a CP is added, and discussed in Sec. 5.4, to ensure favorable channel properties. A CP serves as a guard interval to prevent Inter-Symbol Interference (ISI). Further, it transforms multipath fading channel distortions into cyclic convolution which we exploit during channel equalization to

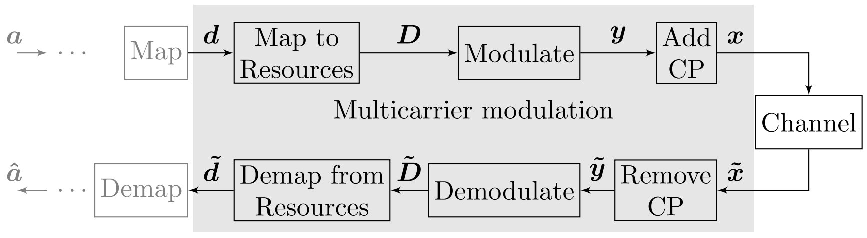


**Figure 5.1:** Multicarrier modulation flowgraph

use simple one-tap equalizers. The corresponding receiver operations are discussed in conjunction with their transmit counter-parts. However, we expect some receiver-specific operations, namely channel estimation and equalization.

## 5.2 Resource grid

Multicarrier systems divide the available bandwidth $B_s$ into $K_s$ equally-spaced subcarriers with subcarrier spacing $B_s/K_s$ as shown in Fig. 5.2 [SGA14, MMG$^+$14]. Further, resources are divided into $K_t$ timeslots over time to comprise frames of size $N_F = K_t \cdot K_s$. In an OFDM system one timeslot corresponds to one OFDM symbol.

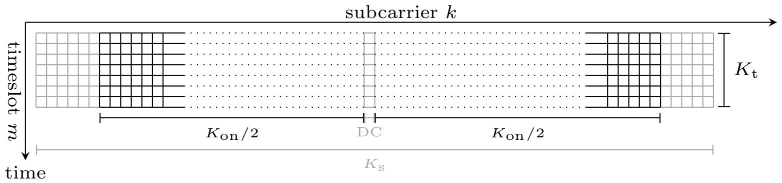


**Figure 5.2:** Multicarrier resource grid with $K_s$ subcarriers and $K_{on}$ active subcarriers symmetrically placed around a DC subcarrier. $K_t$ timeslots divide the resource grid in time. Individual elements, or complex symbols, $d_{m,k}$ are located in the $m$th timeslot on the $k$th subcarrier.

Now, each element $d$ from $\boldsymbol{d} \in \mathbb{C}^{N_d}$ is mapped to a unique point on the resource grid, or time-frequency plane, or lattice, represented by $\boldsymbol{D} \in \mathbb{C}^{K_t \times K_s}$ where an element $d_{m,k}$ corresponds to a symbol in the $m$th timeslot on the $k$th subcarrier [MMG$^+$14, SGA14, Gol05, KD18]. We denote a frame on the resource grid $\boldsymbol{D}$ with $N_F = K_t \cdot K_s$ elements. With $\boldsymbol{d}_k \in \mathcal{A}_C{}^{K_t}$ we denote all elements on the $k$th subcarrier over all timeslots $K_\mathrm{t}$ and with $\boldsymbol{d}_m \in \mathcal{A}_\mathrm{C}{}^{K_\mathrm{s}}$ we denote all elements in the $m$th timeslot over all subcarriers $K_\mathrm{s}$.

Moreover, only $K_\mathrm{on}$ active subcarriers are occupied with symbols and $K_\mathrm{s} - K_\mathrm{on}$ are unused to realize a guard band, shown left and right in Fig. 5.2. Often, the carrier frequency $f_c$ and the Local Oscillator (LO) frequency coincide, i.e. we abstain from offset tuning. In this case, it is desirable to leave the corresponding baseband subcarrier, also known as Direct Current (DC) subcarrier, unoccupied.

Together with the number of timeslots $K_\mathrm{t}$ we can denote a resource grid for one frame that conveys $N_\mathrm{d} = K_\mathrm{t} \cdot K_\mathrm{on}$ symbols and, together with a guard band and DC subcarrier, occupies $K_\mathrm{t} \cdot K_\mathrm{s}$ resources. An OFDM system modulates all $K_\mathrm{s}$ complex symbols in one timeslot $m$ into one OFDM symbol and adds a CP to each OFDM symbol. While OFDM modulates individual timeslots in a frame, GFDM modulates an entire frame jointly. Moreover, OFDM is a special case of GFDM with $K_\mathrm{t} = 1$ and a rectangular filter. However, it is possible to use GFDM and divide a frame into parts that are modulated individually with GFDM.

Communications engineers must choose these parameters carefully to match expected channel conditions. For example, if the number of timeslots $K_\mathrm{t}$ results in a frame duration that is longer than the expected channel coherence time, the system design assumptions are violated. While the system design assumptions are discussed in Sec. 2.2.2, we want to re-iterate those assumptions here in accordance with [Pro95, Gol05, KD18]. First, individual subcarriers are approximately frequency flat. A CP effectively prevents ISI if $N_\mathrm{CP} \geq N_\mathrm{h}$, i.e. the assumption that the CP duration $T_\mathrm{s} N_\mathrm{CP}$ is greater or equal to the maximum relevant channel delay $\tau_\mathrm{max}$ holds. Finally, we assume a block fading channel as discussed in Sec. 2.2.2. If this assumptions are violated a system might not perform as expected or may even be rendered completely dysfunctional.

## 5.3 Multicarrier modulation

The core signal processing step is multicarrier modulation where all the minute differences surface. While OFDM modulates single timeslots with a fixed filter, GFDM modulates frames, or multiple timeslots, with a flexible filter [Pro95, MZS+16]. The GFDM approach introduces more flexibility and thus, GFDM can be better matched to different individual use-cases, e.g. better Time-Frequency-Localization (TFL) and thus smaller guard bands or minimize latencies by designing a system accordingly. According to the Balian-Low theorem, we must choose between TFL, orthogonality and efficient resource usage [SGA14]. In case of OFDM we choose orthogonality and efficient resource usage but the signal is not localized in frequency. Generally, in case of GFDM we choose TFL and efficient resource usage at the expense of a non-orthogonal system.

### 5.3.1 Orthogonal frequency division multiplexing

Orthogonal Frequency Division Multiplexing (OFDM) is a widely used multicarrier modulation [Pro95, Gol05, KD18]. It is efficiently implementable via FFTs, simple one-tap per subcarrier Frequency-Domain Equalization (FDE), and it integrates well with other technologies, such as Multiple Input Multiple Output (MIMO).

We obtain OFDM symbols by modulating each timeslot with

$$\boldsymbol{y}_m = \mathcal{F}_{K_\mathrm{s}}^{-1} \boldsymbol{d}_m \tag{5.1}$$

from $\boldsymbol{D}$ with a $K_\mathrm{s}$ point inverse Discrete Fourier Transform (DFT) $\mathcal{F}_{K_\mathrm{s}}^{-1}$, i.e. we transform each vector into the time domain [Pro95]. Similarly to (5.1), we compute the received OFDM symbol vector

$$\tilde{\boldsymbol{d}}_m = \boldsymbol{G} \cdot \mathcal{F}_{K_\mathrm{s}} \tilde{\boldsymbol{y}}_m \tag{5.2}$$

at the receiver. The modulated receive vector $\tilde{\boldsymbol{y}}_m$ from timeslot $m$ is Fourier transformed $\mathcal{F}_{K_\mathrm{s}}$ to the frequency domain. Afterwards, we perform equalization with a FDE matrix $\boldsymbol{G} \in \mathbb{C}^{K_\mathrm{s} \times K_\mathrm{s}}$ that we discuss in Sec. 5.3.2.

### 5.3.2 Non-iterative equalization strategies

The receiver needs to mitigate channel distortions. With the channel assumptions, discussed in Sec. 2.2.2, we can narrow down the choice of non-iterative equalizers to a simple one-tap Matched-Filter (MF), Zero-Forcing (ZF), or Minimum Mean Square Error (MMSE) equalizer per subcarrier, i.e. we use a Frequency-Domain Equalization (FDE) [Pro95]. Most importantly, we assume frequency flat subcarrier channels, with block fading, i.e. the channel does not change for the duration of a frame. It is a part of appropriate communication system design to find a suitable parameter set that fulfills these requirements.

With this assumption, the equalization matrix $\boldsymbol{G}$ in (5.2) is a square matrix and we assume that all off-diagonal elements are zero. Thus, we identify each diagonal element $g_{kk}$ in our equalization matrix $\boldsymbol{G}$ and simplify our notation to $g_k$ for each diagonal element. While OFDM operates with an equalizer matrix $\boldsymbol{G} \in \mathbb{C}^{K_\mathrm{s} \times K_\mathrm{s}}$ per timeslot, GFDM FDE employs the same technique with an equalizer matrix $\boldsymbol{G} \in \mathbb{C}^{N_\mathrm{F} \times N_\mathrm{F}}$ per frame.

**Matched-Filter (MF)** equalization is suited for Phase Shift Keying (PSK) constellations which solely use phase to carry information. We compute MF equalizer taps with

$$g_k = \breve{h}_k^* \tag{5.3}$$

where $^*$ denotes complex conjugation and $\breve{h}$ denotes a channel tap in the frequency domain as discussed in (2.17). The MF approach is ideal in the sense that it maximizes Signal-to-Noise-Ratio (SNR) for PSK modulation. However, this approach introduces a bias, i.e. for constellations which use amplitude to carry information, the MF approach adds additional amplitude distortion [KD18]. Thus, we may normalize the equalizer taps

$$g_k = \frac{\breve{h}_k^*}{|\breve{h}_k|^2} \tag{5.4}$$

such that we compensate for the introduced bias [Ver98, Gal08]. Now, the MF approach is suitable for Amplitude Shift Keying (ASK), and thus Quadrature Amplitude Modulation (QAM), constellation equalization.

**Zero-Forcing (ZF)** equalization removes any fading distortion at the expense of possible noise enhancement. We compute ZF equalizer taps with

$$g_k = \breve{h}_k^{-1} \tag{5.5}$$

and thus just the inverse of the subcarrier channel tap $\breve{h}_k$.

**Minimum Mean Square Error (MMSE)** equalization tries to find an equilibrium between SNR maximization and fading distortion removal [Gol05, KD18]. This approach leads to the computation

$$g_k = \frac{\breve{h}_k^*}{\breve{h}_k^* \breve{h}_k + \sigma_\mathrm{n}^2} = \frac{\breve{h}_k^*}{\left|\breve{h}_k\right|^2 + \sigma_\mathrm{n}^2} . \tag{5.6}$$

The MMSE approach requires additional information about $\sigma_\mathrm{n}^2$ that the receiver needs to obtain.

## 5.3.3 Generalized frequency division multiplexing

We present Generalized Frequency Division Multiplexing (GFDM) theory in this section and note that the results are partially published in two prior publications [DBD17a, DBD$^+$17b]. While OFDM modulates individual

OFDM symbols, or timeslots, GFDM modulates frames that consist of multiple timeslots. Thus, it is only necessary to consider one CP per frame which in turn enables shorter frames and potentially lower latency compared to OFDM. Further, we may exploit more flexibility with filter design and ensure TFL to potentially decrease required guard bands between co-existing systems [SGA14]. However, these measures may increase receiver complexity because we need to mitigate self-interference caused by non-orthogonal filter design.

We consider a multicarrier resource grid $\boldsymbol{D}^{K_\mathrm{t} \times K_\mathrm{s}}$ and stack its columns $\boldsymbol{d}_k \in \mathbb{C}^{K_\mathrm{t}}$ into a stacked vector

$$\boldsymbol{d}_\mathrm{SRG} = \begin{bmatrix} \boldsymbol{d}_0^T & \boldsymbol{d}_1^T & \dots & \boldsymbol{d}_k^T & \dots & \boldsymbol{d}_{K_\mathrm{s}-2}^T & \boldsymbol{d}_{K_\mathrm{s}-1}^T \end{bmatrix}^T \in \mathbb{C}^{K_\mathrm{t} K_\mathrm{s}} \tag{5.7}$$

with $N_\mathrm{F} = K_\mathrm{t} K_\mathrm{s}$ elements [MMG+14]. The GFDM modulator computes a linear operation

$$\boldsymbol{y} = \boldsymbol{A}_\mathrm{mod} \boldsymbol{d}_\mathrm{SRG} \tag{5.8}$$

to modulate whole frames with the modulation matrix $\boldsymbol{A}_\mathrm{mod}$ [MMG+14]. This modulation matrix $\boldsymbol{A}_\mathrm{mod}$ contains the filter coefficients for one symbol in each column and can be written as

$$\boldsymbol{A}_\mathrm{mod} = \begin{bmatrix} \boldsymbol{\omega}_{0,0} & \boldsymbol{\omega}_{0,1} & \dots & \boldsymbol{\omega}_{1,0} & \dots & \boldsymbol{\omega}_{K_\mathrm{s}-1, K_\mathrm{t}-1} \end{bmatrix} \in \mathbb{C}^{N_\mathrm{F} \times N_\mathrm{F}} \tag{5.9}$$

where the columns represent filters derived from a prototype filter $\boldsymbol{\omega} \in \mathbb{C}^{N_\mathrm{F} \times 1}$ [MMG+14]. The $i$-th element of the derived filter for the $k$-th subcarrier in the $m$-th timeslot is obtained by mixing and circularly shifting the prototype filter

$$\omega_{k,m}[i] = \omega[(i - mK_\mathrm{s}) \mod N_\mathrm{F}] \cdot e^{j2\pi i \frac{k}{K_\mathrm{s}}}. \tag{5.10}$$

The cyclic shift with $mK_\mathrm{s} \mod N_\mathrm{F}$ retains the cyclic frame property that is similar to the OFDM case where we use a FFT. Furthermore, with $K_\mathrm{t} = 1$ and a rectangular prototype filter $\boldsymbol{\omega}$ we obtain an OFDM system as a special case [MMG+14].

From (5.8) it can be observed that GFDM is a linear, frame-based multicarrier modulation scheme. However, the potentially large matrix multiplication is a very expensive operation and we discuss efficient modulation in Sec. 5.3.3. The desired spectral and interference properties can be designed by choosing an appropriate prototype filter $\boldsymbol{\omega}$, e.g., Root-Raised-Cosine (RRC), or Gaussian filters. OFDM is a special case of GFDM with $K_\mathrm{t} = 1$ timeslot and a rectangular time domain filter. The chosen prototype filter may constitute orthogonal or non-orthogonal modulation and thus controls if and how much self-interference is present in a specific GFDM system. Appropriate prototype filter design may be performed with the aid of ambiguity functions [MWBD15, Du08]. In general, non-orthogonal modulation must be assumed and self-interference must be considered.

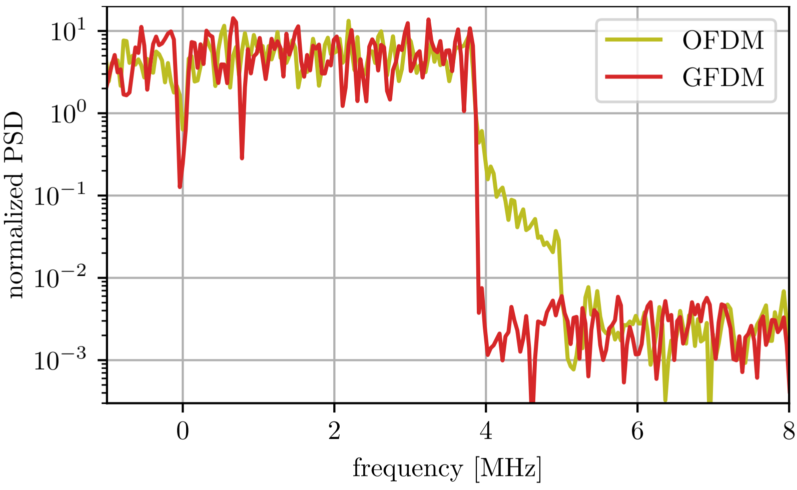


**Figure 5.3:** OFDM and GFDM periodograms with $K_\mathrm{s} = 64$, $K_\mathrm{on} = 48$, $N_\mathrm{CP} = 8$, $N_\mathrm{W} = 4$, 40 dB SNR, and 10 MHz bandwidth. $N_\mathrm{CP}$ and $N_\mathrm{W}$ are discussed in Sec. 5.4.

The periodograms, or Power Spectral Densitys (PSDs), in Fig. 5.3 exemplify a major GFDM advantage, namely frequency localization. The normalized periodograms are computed via oversampling and with Welch's method [Wel67]. While we can observe strong OOB emissions for OFDM, GFDM is well contained within the desired bandwidth from −3.75 MHz to 3.75 MHz. With $K_\mathrm{s} = 64$ subcarriers over 10 MHz bandwidth, a signal would use frequencies form −5 MHz to 5 MHz. Since the signal in Fig. 5.3 is designed with these properties before upsampling, we observe a drop in the PSD at 5 MHz. However, the system is configured with $K_\mathrm{on} = 48$ active subcarriers and thus, the active, or desired, bandwidth is 7.5 MHz and a signal would use frequencies from −3.75 MHz to 3.75 MHz. Thus, it is possible to reduce guard bands of GFDM systems to neighboring systems in comparison to OFDM systems and thus better utilize scarce spectrum. We discuss further GFDM advantages, e.g., frame length, and latency, in Fig. 5.4.

### Efficient modulation

Matrix multiplication is an expensive operation, especially if $N_\mathrm{F}$ tends to be large. We perform efficient frequency domain GFDM modulation and demodulation to drastically reduce complexity [GMN+13] and achieve low latency processing [DBD17a, DBD+17b]. The main advantages of the presented approach are efficient use of the FFT algorithm, element-wise multiplications and additions instead of large matrix multiplications, and complexity reduction that is steered by the overlap factor $N_\mathrm{OV}$. The implementation in [DRKK22] uses the presented approach to enable low latency processing.

In accordance with [GMN+13], we rewrite (5.8) to

$$\boldsymbol{y} = \boldsymbol{A}_\mathrm{mod}\boldsymbol{d}_\mathrm{SRG} = \mathcal{F}_{N_\mathrm{F}}^{-1}\left\{\sum_{k=0}^{K_\mathrm{s}-1} \boldsymbol{P}^{(k)}_{N_\mathrm{F}\times N_\mathrm{t,OV}} \boldsymbol{\Omega}_{N_\mathrm{t,OV}\times N_\mathrm{t,OV}} \boldsymbol{R}_{N_\mathrm{t,OV}\times K_\mathrm{t}} \mathcal{F}_{K_\mathrm{t}} \boldsymbol{d}_k\right\} \tag{5.11}$$

where $\boldsymbol{d}_k \in \mathbb{C}^{K_\mathrm{t}}$ denotes the complex symbols modulated onto one subcarrier. First, $\boldsymbol{d}_k$ is transformed per subcarrier with a $K_\mathrm{t}$-point DFT $\mathcal{F}_{K_\mathrm{t}}$ into the frequency domain.

Next, a repetition matrix $\boldsymbol{R}_{N_\mathrm{t,OV}\times K_\mathrm{t}} \in \mathbb{R}^{N_\mathrm{t,OV}\times K_\mathrm{t}}$ is used for upsampling with overlap factor $N_\mathrm{OV} \leq K_\mathrm{s}$ and the short hand $N_\mathrm{t,OV} = K_\mathrm{t}N_\mathrm{OV}$. The repetition matrix consists of $N_\mathrm{OV}$ repetitions of a $\boldsymbol{I}_{K_\mathrm{t}}$ identity matrix. The multiplication with a repetition matrix can be implemented via a lightweigth copy operation in software. For $K_\mathrm{s} = N_\mathrm{OV}$, (5.11) is an alternative representation of (5.8). If the prototype filter $\boldsymbol{\omega}$ is chosen such that its OOB leakage decays outside its subcarrier bandwidth, $N_\mathrm{OV}$ can become smaller than $K_\mathrm{s}$ [GMN+13]. In most cases $N_\mathrm{OV} = 2$ is considered to be sufficient and it becomes clear that $N_\mathrm{OV}$ controls a modulators computational complexity.

The diagonal filter matrix $\boldsymbol{\Omega}_{N_\mathrm{t,OV}\times N_\mathrm{t,OV}}$ holds $\boldsymbol{\omega} \in \mathbb{C}^{N_\mathrm{t,OV}}$ filter taps on its diagonal while all off-diagonal elements are zero. The multiplication with the diagonal filter matrix is efficiently implemented via element-wise multiplications in [DRKK22]. Under the assumption that $\boldsymbol{\omega}$ decays outside the subcarrier bandwidth, it holds only values with very low or zero amplitude outside its subcarrier bandwidth and thus, truncating this filter is further justified. In this case, only neighboring subcarriers overlap and selecting a smaller $N_\mathrm{OV}$ solely steers complexity. Our considerations signify the observation that GFDM is a non-orthogonal multicarrier modulation scheme and we must expect self-interference.

$\boldsymbol{P}^{(k)}_{N_\mathrm{F}\times N_\mathrm{t,OV}} \in \mathbb{R}^{N_\mathrm{F}\times N_\mathrm{t,OV}}$ performs subcarrier mixing in the frequency domain by shifting the samples into a vector of size $N_\mathrm{F}$ at the corresponding position of the $k$th subcarrier. The subcarrier mixing operation is implemented via pointer offset computations and addition to the input vector to the inverse FFT $\mathcal{F}_{N_\mathrm{F}}^{-1}$. Furthermore, the filter operation and mixing operations can be combined into a multiply accumulate operation that may further reduce latency. Finally, all subcarrier modulations are summed up and transformed back to the time domain with $\mathcal{F}_{N_\mathrm{F}}^{-1}$.

Considering (5.11), it becomes apparent that $K_\mathrm{t}$ and $K_\mathrm{s}$ should both be a power of two in order to exploit the efficient Cooley-Tukey FFT algorithm [FJ05]. However, GFDM systems exhibit bad performance if both, $K_\mathrm{t}$ and $K_\mathrm{s}$, are even numbers [MMF14]. Thus, we mostly consider systems where either $K_\mathrm{t}$ or $K_\mathrm{s}$ is odd.

### Interference-cancellation

Since GFDM is a non-orthogonal modulation scheme in general, Interference-Cancellation (IC) might be performed in order to remove, or reduce, self-interference [GMN+13]. Unlike OFDM, GFDM enables to control self-interference by means of filter design, e.g. by choosing an appropriate roll-off factor for RRC filters. The argument that the overlap factor $N_\mathrm{OV} = 2$ suffices is tantamount to only adjacent subcarriers need to be considered for IC. In case of RRC filters $\boldsymbol{\omega}$, we only expect Inter-Carrier Interference (ICI), i.e. adjacent subcarrier interference, but timeslots do not interfere.

A GFDM receiver expects a received frame

$$\tilde{\boldsymbol{y}} = \breve{\boldsymbol{H}} \cdot \boldsymbol{y} + \boldsymbol{n} \tag{5.12}$$

in accordance with Sec. 2.2.2 where $\breve{\boldsymbol{H}}$ is a diagonal cyclic block fading channel matrix and a noise vector $\boldsymbol{n}$. Here we discuss the receiver operations including equalization and IC that a GFDM receiver needs to perform based on the modulation (5.11). The demodulator first obtains an equalized frequency domain vector

$$\tilde{\boldsymbol{y}}_\mathrm{EQ} = \boldsymbol{G}\mathcal{F}_{N_\mathrm{F}}\,\tilde{\boldsymbol{y}} \tag{5.13}$$

by transforming the received vector $\tilde{\boldsymbol{y}}$ to the frequency domain and performing one-tap FDE with $\boldsymbol{G}$ in accordance with (5.3), (5.5), and (5.6). We obtain a per subcarrier $k$ demodulated frequency domain vector

$$\tilde{\boldsymbol{y}}_{\mathrm{F},k}^{0} = \left(\boldsymbol{R}_{N_\mathrm{t,OV}\times K_\mathrm{t}}\right)^T \boldsymbol{\Omega}_{\mathrm{RX},N_\mathrm{t,OV}\times N_\mathrm{t,OV}} \left(\boldsymbol{P}^{(k)}_{N_\mathrm{F}\times N_\mathrm{t,OV}}\right)^T \tilde{\boldsymbol{y}}_\mathrm{EQ} \tag{5.14}$$

by first shifting it back to the zero subcarrier with the real-valued matrix $\left(\boldsymbol{P}^{(k)}_{N_\mathrm{F}\times N_\mathrm{t,OV}}\right)^T$, then we apply a truncated receiver filter $\boldsymbol{\Omega}_{\mathrm{RX},N_\mathrm{t,OV}\times N_\mathrm{t,OV}}$, and finally sum up all subcarrier parts with the real-valued matrix $\left(\boldsymbol{R}_{N_\mathrm{t,OV}\times K_\mathrm{t}}\right)^T$. In order to maximize SNR, we assume a MF design for

$\boldsymbol{\Omega}_{\mathrm{RX},K_\mathrm{t}N_\mathrm{OV}\times K_\mathrm{t}N_\mathrm{OV}} = \boldsymbol{\Omega}_{K_\mathrm{t}N_\mathrm{OV}\times K_\mathrm{t}N_\mathrm{OV}}$ for real-valued filter taps. Thus, $\tilde{\boldsymbol{y}}^0_{\mathrm{F},k}$ only suffers from interference from adjacent subcarriers [MMF14]. A ZF receive filter design would remove all self-interference, introduced by the non-orthogonal waveform, at the expense of noise-enhancement. We may obtain demodulated symbols per subcarrier

$$\tilde{\boldsymbol{d}}_k = \mathcal{F}^{-1}_{K_\mathrm{t}} \tilde{\boldsymbol{y}}^0_{\mathrm{F},k} \tag{5.15}$$

to produce $\tilde{\boldsymbol{D}}$.

We introduce Interference-Cancellation (IC) with $J$ iterations to combat interference in $\tilde{\boldsymbol{d}}_k$ from adjacent subcarriers $\tilde{\boldsymbol{d}}_{k-1}$ and $\tilde{\boldsymbol{d}}_{k+1}$ [GMN+13]. The algorithm is parallelizable per iteration but each iteration runs successively. In each iteration $j$, we compute hard decisions with (4.3)

$$\hat{\boldsymbol{d}}^j_k = \mathcal{Q}_{\mathcal{A}_\mathrm{C}}\left(\tilde{\boldsymbol{d}}^j_k\right) \tag{5.16}$$

from the soft values we obtain with (5.15). Unoccupied subcarriers produce a zero vector $\hat{\boldsymbol{d}}_k = \boldsymbol{0}$, otherwise these subcarriers would introduce additional interference. Then, the interference from adjacent subcarriers

$$\tilde{\boldsymbol{y}}^j_{\mathrm{I},k} = \boldsymbol{\Omega}_\mathrm{I} \mathcal{F}_{K_\mathrm{t}} \left(\hat{\boldsymbol{d}}^j_{k+1 \mod K_\mathrm{s}} + \hat{\boldsymbol{d}}^j_{k-1 \mod K_\mathrm{s}}\right) \tag{5.17}$$

is calculated where $\boldsymbol{\Omega}_\mathrm{I}$ accounts for the weighted interference derived from the used filters [GMN+13]. Eventually, the next IC iteration $j+1$ is performed with updated receive vectors

$$\tilde{\boldsymbol{y}}^{j+1}_{\mathrm{F},k} = \tilde{\boldsymbol{y}}^j_{\mathrm{F},k} - \tilde{\boldsymbol{y}}^j_{\mathrm{I},k} \,. \tag{5.18}$$

This process from (5.15) onward is repeated $J$-times with updated values $\tilde{\boldsymbol{y}}^{j+1}_{\mathrm{F},k}$ in order to minimize self-interference. After $J$ iterations the demodulator returns interference-reduced soft values $\tilde{\boldsymbol{D}}$ that are passed on to resource demapping. With IC, the receiver needs to perform more $K_\mathrm{t}$ point DFTs $\mathcal{F}_{K_\mathrm{t}}$ and its inverse $\mathcal{F}^{-1}_{K_\mathrm{t}}$ because these operations are part of every IC iteration. Thus, the focus for latency benchmarks is further pushed towards the latency of the operations (5.15) – (5.18) that are computed in every IC iteration in contrast to (5.13) and (5.14) that are only computed once.

## 5.4 Cyclic prefix

Most multicarrier systems consider a Cyclic Prefix (CP) with $N_\mathrm{CP}$ elements [Pro95]. A CP consists of the last $N_\mathrm{CP}$ elements from $\boldsymbol{y}$ that are prepended to constitute a guard interval. The length is chosen such that it exceeds, or at least equals, the expected maximum channel delay to form a guard interval against ISI. As further discussed in Sec. 2.2.2, a CP helps to transform the convolution of the baseband signal $\boldsymbol{s}$ and the frequency-selective fading channel taps $\boldsymbol{h}$ into a cyclic convolution under the assumption that $N_{\mathrm{CP}} \geq N_{\mathrm{h}}$ holds. Moreover, this leads to a cyclically convolved modulated receive vector $\tilde{\boldsymbol{y}}$ which in turn leads to a simple one-tap FDE [Pro95, Gol05, KD18]. In case of OFDM, a DFT produces a cyclic timeslot that is prepended by a CP to protect against ISI. Due to the cyclic filter shift in (5.10), a GFDM frame is cyclic as well. Thus, a CP can be employed to obtain a cyclically convolved signal at the receiver and again a one-tap FDE is feasible. In contrast to OFDM, only one CP per frame is necessary for GFDM which can be utilized to shorten frames and thus in turn reduce latency and improve resource efficiency.

For our following considerations, we focus on GFDM frames, though the argument applies to OFDM timeslots alike. A CP is prepended to a frame by taking the last $N_{\mathrm{CP}}$ elements from $\boldsymbol{y} = [y_0, \dots, y_{N_{\mathrm{F}}-1}]$ to use it as a prefix. Similarly, it is possible to append a Cyclic Suffix (CS) by taking the first $N_{\mathrm{CS}}$ elements from $\boldsymbol{y}$ and append them to use them to smoothen transition in conjunction with windowing. We obtain

$$\boldsymbol{x}_{\mathrm{NW}} = [y_{N_{\mathrm{F}}-N_{\mathrm{CP}}}, \dots, y_{N_{\mathrm{F}}-1}, y_0, \dots, y_{N_{\mathrm{F}}-1}, y_0, \dots, y_{N_{\mathrm{CS}}-1}] \tag{5.19}$$

by prepending a CP and appending a CS.

Apart from CP, and CS, insertion, frame windowing is performed to reduce OOB emissions [MMG+14]. In accordance with [IEE12], a Raised-Cosine (RC) filter is applied where $N_{\mathrm{W}}$ elements at the beginning and end of $\boldsymbol{x}_{\mathrm{NW}}$ are considered. A window with $2N_{\mathrm{W}}$ RC taps is applied element-wise to the first $N_{\mathrm{W}}$ and last $N_{\mathrm{W}}$ elements of $\boldsymbol{x}_{\mathrm{NW}}$ to obtain $\boldsymbol{x}$. At the receiver we obtain a received modulated frame

$$\tilde{\boldsymbol{y}} = [\tilde{x}_{N_{\mathrm{CP}}}, \dots, \tilde{x}_{N_{\mathrm{F}}+N_{\mathrm{CP}}-1}] \tag{5.20}$$

by simply removing all CP and CS elements.

In case of OFDM, we apply windowing per timeslot and use the CS to smoothen discontinuities between adjacent timeslots. Thus, the last $N_{\mathrm{CS}}$ elements and the first $N_{\mathrm{CS}}$ from the next timeslot are added up element-wise [IEE12]. We introduce a CS to aide windowing, thus we expect $N_{\mathrm{CS}} = N_{\mathrm{W}}$ in most cases for both OFDM and GFDM. In order to reduce latency, we investigate efficiency for OFDM and GFDM in Fig. 5.4. For the sake of simplicity, we assume $K_{\mathrm{s}} = K_{\mathrm{on}}$, i.e. all subcarriers are occupied. Further, we assume that the CP is $N_{\mathrm{CP}} = K_{\mathrm{s}}/2$ and the CS length is $N_{\mathrm{CS}} = K_{\mathrm{s}}/4$. The large CP length is the same for both, GFDM and OFDM, for clarity.

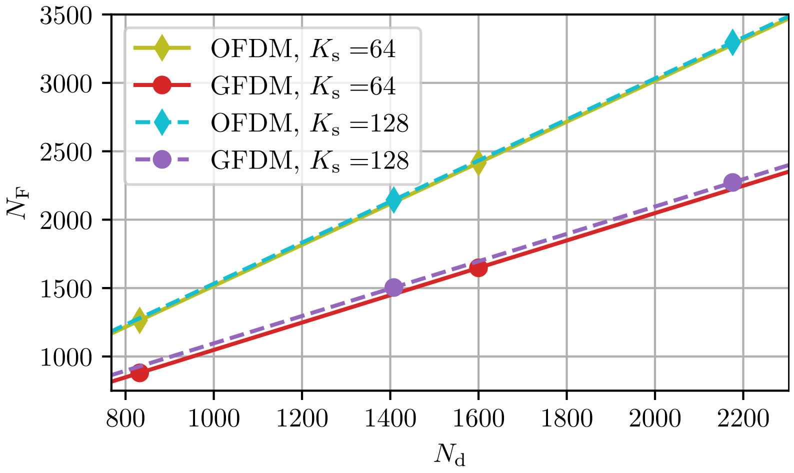


**Figure 5.4:** OFDM and GFDM efficiency, $K_\mathrm{s} = K_\mathrm{on}$, $N_\mathrm{CP} = K_\mathrm{s}/2$, $N_\mathrm{CS} = K_\mathrm{s}/4$.

With an increasing number of timeslots $K_\mathrm{t}$, and thus growing $N_\mathrm{F}$, we can observe that GFDM is advantageous because it only requires one CP per frame instead of one CP per timeslot. We want to stress the point that this GFDM property translates directly to lower latency because individual GFDM frames exhibit a shorter Over-the-Air (OTA) duration $T_\mathrm{s}N_\mathrm{F}$ than their OFDM counterparts when those systems convey the same number of complex symbols $N_\mathrm{d}$. Furthermore, this increases efficiency because we spend fewer resources on CPs.

## 5.5 Channel estimation

Unless we transmit over an Additive White Gaussian Noise (AWGN) channel, a communication system must equalize received frames and thus requires a channel estimate for various processing steps. A multicarrier demodulator requires a channel estimate for equalization and a symbol demapper requires SNR or better Carrier-to-Noise-Ratio (CNR) information for accurate Log-Likelihood Ratio (LLR) computation. We rely on a Schmidl&Cox preamble $\boldsymbol{y}_\mathrm{P}$ based approach with two identical parts to perform channel estimation as discussed in Sec. 8.5.2 [ZM09, SC97, AKE08, GMMF14]. We expect the same CP, CS, and windowing procedure as with every data frame to obtain a preamble $\boldsymbol{x}_\mathrm{P}$, only its transmitted data is known to the receiver.

Some algorithms we discuss in our work require SNR knowledge, e.g., LLR calculation, and MMSE equalization. We implement SNR, and CNR, estimation with the algorithm in [ZM09] based on a received preamble $\tilde{\boldsymbol{y}}_{\mathrm{P}}$. We refer the interested reader to [ZM09] for further information.

For channel tap estimation, we use the received preamble $\tilde{\boldsymbol{y}}_{\mathrm{P}}$ without a corresponding CP and CS for channel estimation in the frequency domain. We compute subcarrier channel coefficients

$$\breve{h}_k = \frac{1}{2}\left(\frac{\tilde{y}_{\mathrm{P}0,k}}{y_{\mathrm{P}0,k}} + \frac{\tilde{y}_{\mathrm{P}1,k}}{y_{\mathrm{P}1,k}}\right) \tag{5.21}$$

by averaging over both timeslots per subcarrier. In case of GFDM, we employ linear interpolation to obtain a $N_{\mathrm{F}}$ element estimate from our $K_{\mathrm{s}}$ element original estimate. Since we focus on I4.0 URLLC, we expect small packets and thus short frames that are subject to block fading as discussed in Sec. 2.2 and thus further justify a preamble-based approach.

## 5.6 Numerical software defined radio parameter evaluation

In this section we explore how our system performs regarding latency as well as error correction performance depending on varying parameters to identify suitable setups for URLLC. This is an extension to our previous works [DBD17a, DBD+17b]. In this work we investigate a broader set of parametrizations and incorporate investigations into FFT sizes. The primary target is to measure latencies for the different stages in the signal processing chain outlined in Sec. 5.1. Latency benchmarks run on a AMD Ryzen Threadripper 3970X (TRX3970X) Linux host as discussed in Sec. 8.6 and 8.4.1. Sec. 8.5.3 discusses our software project *gr-gfdm* in further detail [DRKK22]. The latency benchmarks on *gr-gfdm* are carried out as described in Sec. 8.4.4. The benchmarks present a minimum latency cost for the investigated multicarrier system. Additional latencies due to multi-threading, peripheral access and such are not considered. The simulation benchmarks help identify suitable parameter sets for the task at hand. Also, they help at identifying possible targets for future optimizations.

### 5.6.1 Resource mapping

Before modulation, the resource mapper assigns symbols to multicarrier resources. Fig. 5.5 and Fig. 5.6 show latencies for resource mapper and demapper with $K_{\mathrm{s}} = 256$. $K_{\mathrm{s}} = 256$ is a representative while other values yield similar results. First, we observe that more active subcarriers $K_{\mathrm{on}}$ contributes to a higher latency for both, resource mapper and demapper.

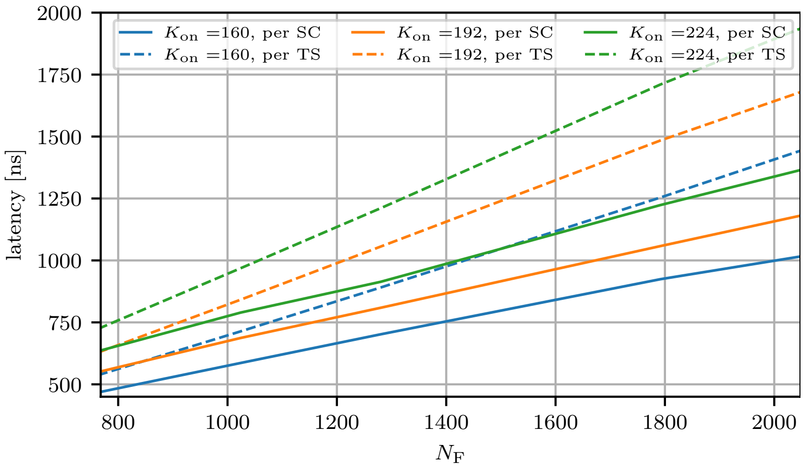


**Figure 5.5:** Resource mapping latency measurements per $K_{\mathrm{on}} \in \{160, 192, 224\}$ with $K_{\mathrm{s}} = 256$, per timeslot (per TS) or per subcarrier (per SC).

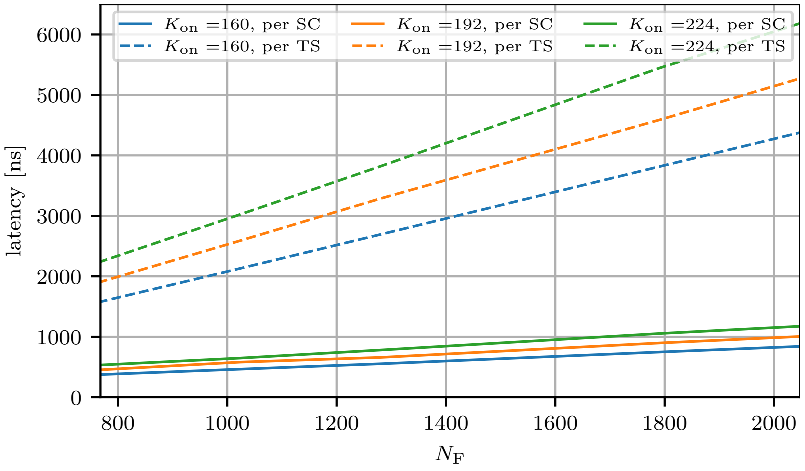


**Figure 5.6:** Resource demapping latency measurements per $K_{\mathrm{on}} \in \{160, 192, 224\}$ with $K_{\mathrm{s}} = 256$, per timeslot (per TS) or per subcarrier (per SC).

Typically, an OFDM system requires per subcarrier resource mapping such that values for one OFDM timeslot are in consecutive memory while GFDM requires per timeslot resource mapping in accordance with (5.11) where a FFT $\mathcal{F}_{K_{\mathrm{t}}}$ is applied to $K_{\mathrm{t}}$ symbols that are modulated onto one subcarrier. We note a significant difference if we map resources per timeslot (per TS) or per subcarrier (per SC). In Fig. 5.6 it can be observed that this difference is more pronounced at the resource demapper.

We conclude that resource mapper and demapper contribute a minor increase in latency compared to multicarrier modulation in Fig. 5.7 and Fig. 5.11. Parametrization for GFDM introduces slightly higher latencies than their OFDM parameter counterpart.

## 5.6.2 Fast Fourier transform

We discuss Fastest Fourier Transform in The West (FFTW) benchmarks because we use it to compute all FFTs [FJ05]. Having more insight here contributes to a better benchmark analysis and deeper insights.

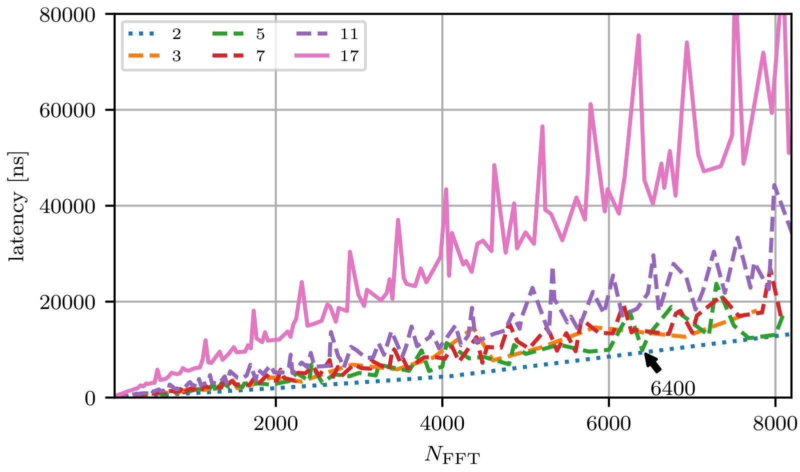


**Figure 5.7:** FFTW latency measurements with different largest primes.

OFDM systems are canonically designed with $N_{\mathrm{FFT}} = 2^{\times}$ being a power of two. This convention implies that the largest prime in $N_{\mathrm{FFT}}$ is $2^{\bullet}$ and can be computed most efficiently with the Cooley-Tukey FFT algorithm [FJ05]. We observe corresponding latencies for power of two $2^{\times}$ FFTs in Fig. 5.7 as the baseline. In case $K_{\mathrm{t}} > 1$ for OFDM, we can simply multiply this latency by $K_{\mathrm{t}}$.

With prime factorization, we compute all prime factors of an integer value, e.g. $N_{\mathrm{FFT}}$ =245 = $5 \cdot 7^2$. Thus, the largest prime in 245 would be $7^{\bullet}$. The second largest prime would be 7 as well, while the third largest would be 5.

We increase the largest prime in $N_{\mathrm{FFT}}$ from $2^{\bullet}$ to $3^{\bullet}$, $5^{\bullet}$, $7^{\bullet}$, $11^{\bullet}$, and $17^{\bullet}$. Small prime values $3^{\bullet}$ and $5^{\bullet}$ show a minor latency increase where $5^{\bullet}$

appears to be the most efficient. We want to point out $N_{\mathrm{FFT}} = 2^8 \cdot 5^2 = 6400$ as an example that is annotated in Fig. 5.7 and produces results very close to our $2^\bullet$ baseline. Other examples for this observation may be found at $N_{\mathrm{FFT}} \in \{400, 640, 1280\}$. $7^\bullet$ already contributes a larger latency increase. We can observe that latency increases significantly with $17^\bullet$ while $11^\bullet$ contributes to a moderate latency increase. We omit other, especially larger, prime values but note that increasing the largest prime increases latency further.

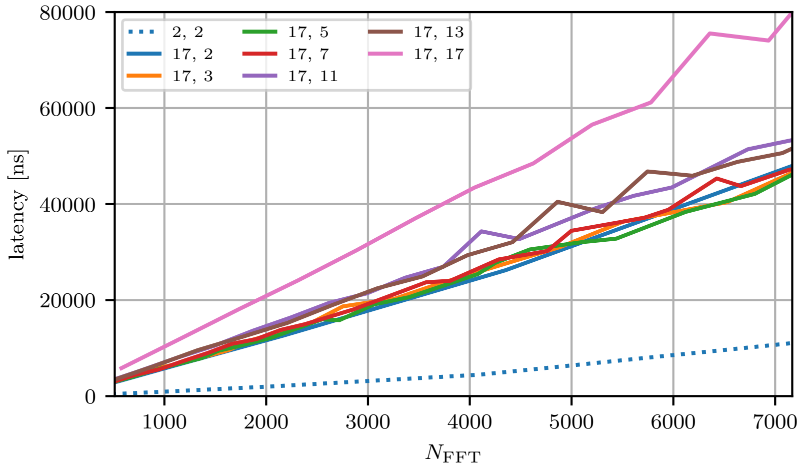


**Figure 5.8:** FFTW latency measurements with largest prime $17^\bullet$ and differing second largest primes. $2^\bullet$ serves as a baseline.

The graph for $17^\bullet$ exhibits significant jumps in latency because further, smaller prime values $\{13, 11, 7, 5, 3, 2\}$ may be present. Fig. 5.8 yields further insight into behavior with $2^\bullet$ as a baseline. The largest prime $17^\bullet$ is the same for all curves, but the second largest differs as indicated. Instead of significant jumps, we observe smoother graphs compared to Fig. 5.7 that lead to the conclusion that the number of large primes is a significant factor in latency increase as well. It becomes obvious that $N_{\mathrm{FFT}}$ must be chosen carefully because large prime values dominate latency.

The FFT algorithm is prone to higher latencies if $N_{\mathrm{FFT}}$ includes large primes. Hence, it is important to find a configuration that requires FFT calculations with a small largest prime value. Further, it is important to pay attention to smaller prime values as well, especially if the largest prime grows.

### 5.6.3 GFDM parameter evaluation on latency

At the heart of the transmitter chain is the Generalized Frequency Division Multiplexing (GFDM) modulator. Fig. 5.9 presents a latency comparison for $K_\mathrm{s} = 64$ with increasing overlap factor $N_\mathrm{OV}$. The modulator causes most

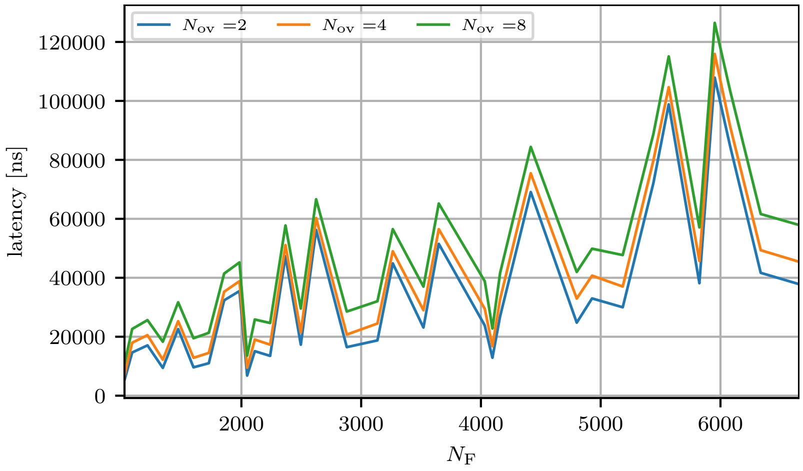


**Figure 5.9:** GFDM modulator latency measurements with varying overlap factor $N_\mathrm{OV}$, $K_\mathrm{s} = 64$.

of the processing latency spent at the transmitter in comparison to symbol mapping and encoder discussed in Sec. 4.6 and Sec. 3.4. In contrast to the other processing steps, it does not exhibit a linear behavior with regards to $N_\mathrm{F}$ due to the employed Fourier transforms that are not powers of $2^\times$. We discuss this observation and reasons in Sec. 5.6.2 extensively.

We compare latency results at $N_\mathrm{F} = 2880 = 5 \cdot 3^2 \cdot 2^6$ with the largest prime $5^\bullet$ and $N_\mathrm{F} = 3264 = 17 \cdot 3 \cdot 2^6$ with the largest prime $17^\bullet$. Measured latency at $N_\mathrm{F} = 2880$ is 16.3 µs for $N_\mathrm{OV} = 2$ and increases by 4.6 µs for $N_\mathrm{OV} = 4$ and by 14 µs for $N_\mathrm{OV} = 8$. While at $N_\mathrm{F} = 3264$ measured latency is 44.7 µs for $N_\mathrm{OV} = 2$ and increases by 4.6 µs for $N_\mathrm{OV} = 4$ and by 14.8 µs for $N_\mathrm{OV} = 8$. While the frame size $N_\mathrm{F}$ increases by roughly 13 %, the latency increases by 174 %. This result signifies the importance to focus on low prime values in $N_\mathrm{F}$. In contrast to the impact of non-power-of-two primes in $N_\mathrm{F}$, the overlap factor $N_\mathrm{OV}$ has a lighter impact on latency increase. Still, the impact of the overlap factor $N_\mathrm{OV}$ on latency is higher than other processing steps in the transmitter processing chain, e.g. resource mapping in Sec. 5.6.1. In conjunction with the assumption that employed filters decay rapidly outside their subcarrier bandwidth, this result indicates that it is advisable to stick to $N_{\mathrm{OV}} = 2$.

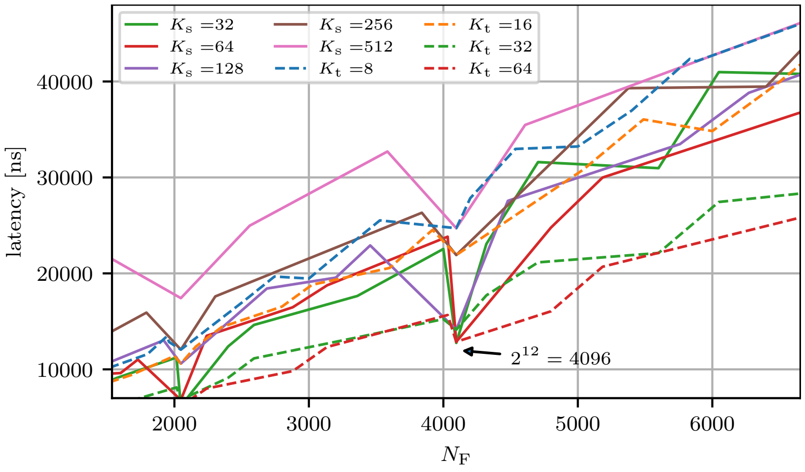


**Figure 5.10:** GFDM modulator latencies for either fixed $K_s$ or $K_t$, $N_{\mathrm{OV}} = 2$.

Equation (5.11) shows that every GFDM modulator needs to compute multiple $K_t$-point FFTs and one $N_F = K_t K_s$-point FFT. Fig. 5.10 explores different options with either $K_t = 2^\times$ or $K_s = 2^\times$ while the respective other value varies. A fixed power of two $2^\times$ value for $K_t$ tends to yield lower latencies. We recommend to increase the number of timeslots $K_t$ to the next power of two if possible if $N_F$ needs to be larger because it is advantageous to decrease latency.

It is worth discussing a special case at $N_F = 4096$. In this case both, $K_t$ and $K_s$, are a power of two. We can observe that larger $K_t$, but $\leq 64$, i.e. the FFT that needs to be computed multiple times, has a larger impact on latency then one large $N_F$-point FFT. Increasing $K_t$ from 64 to 128, however, does not expose the same effect on latency. With $K_t \geq 64$ latency stays roughly the same and it is sufficient to increase the number of subcarriers.

Correspondingly, we investigate latencies for GFDM demodulators in Fig. 5.11. The graph for a modulator with $N_{\mathrm{OV}} = 2$ is included for reference. We conclude that modulator and demodulator incur the same latencies for the same configuration. Again, latency increases if $N_{\mathrm{OV}}$ increases. A GFDM demodulator may include equalization, and thus, we include these results in Fig. 5.11 with dashed lines. We conclude that Frequency-Domain Equalization (FDE) has a minor impact on latency. We note that channel estimation is not a part of equalization but we conduct corresponding channel estimation experiments.

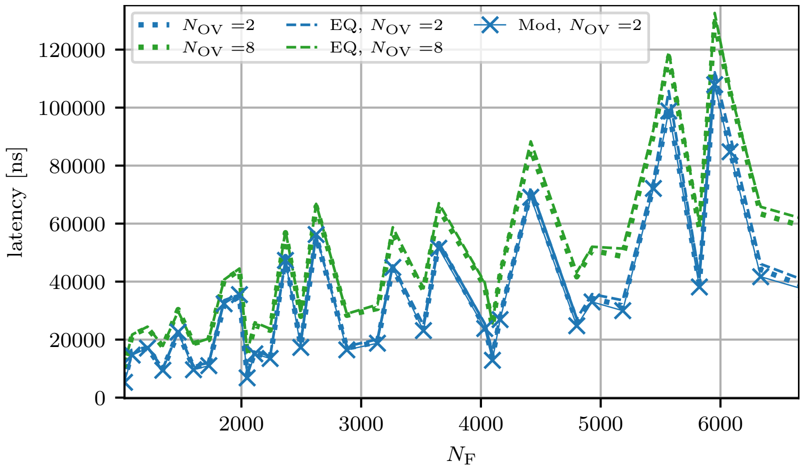


**Figure 5.11:** GFDM demodulator latencies, varying overlap factor, with and without equalization, $K_s = 64$.

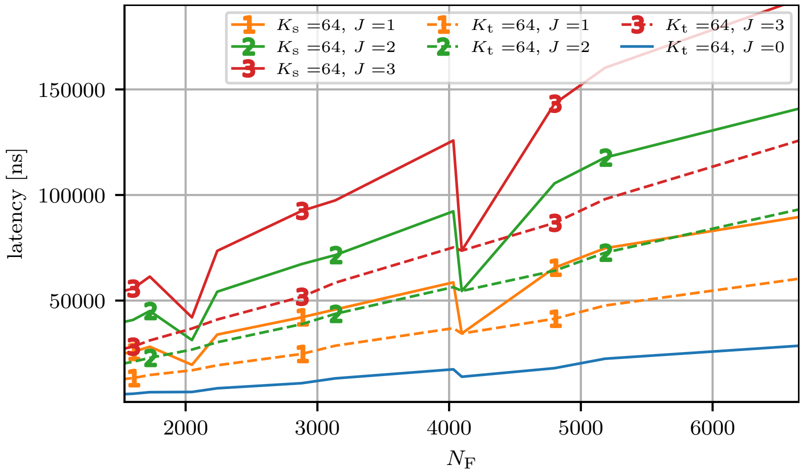


**Figure 5.12:** GFDM demodulator latencies with varying number of IC iterations $J$ for either $K_s = 64$ (solid) or $K_t = 64$ (dashed).

GFDM is not an orthogonal modulation scheme and thus introduces self-interference. To combat this self-interference, GFDM demodulation may incorporate Interference-Cancellation (IC) that we investigate in Fig. 5.12 where more iterations generally improve the Symbol-Error-Rate (SER) performance [GMN+13]. We include a simple demodulator with $J = 0$ as a baseline. While discussing Fig. 5.10 we established that either $K_\mathrm{t} = 64$ or $K_\mathrm{s} = 64$ is a favorable choice to reduce latency. With Fig. 5.12, we conclude that it is favorable to restrict the number of timeslots $K_\mathrm{t} = 2^\times$ to powers of two because this variable has the highest impact on latency. Specifically, $K_\mathrm{t}$ has a higher impact on latency than $K_\mathrm{s}$. This finding is emphasized if we compare the results for $N_\mathrm{F} = 4096$ where we observe a dip in latency for $K_\mathrm{s} = 64$ while $K_\mathrm{t} = 64$ only incurs a minor latency decrease. Since IC is a major contributor to increased latency, it is advisable to restrict $J$ to a minimum.

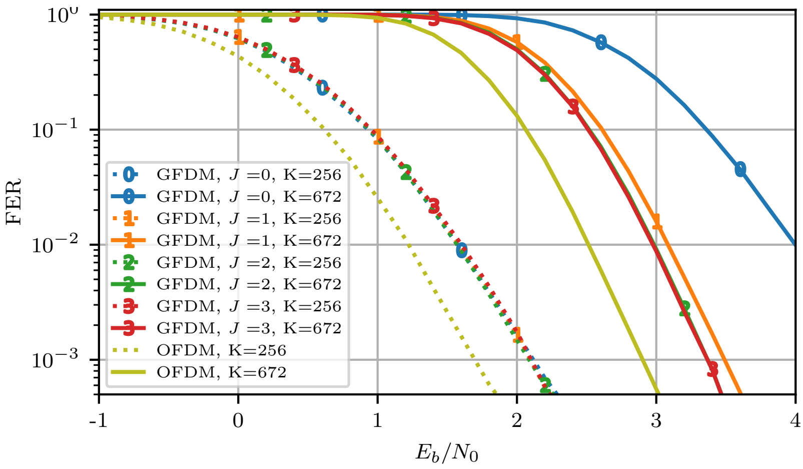


**Figure 5.13:** Multicarrier Frame-Error-Rate (FER) performance with $K_\mathrm{on} = 56$, $K_\mathrm{s} = 64$, $M = 2$, $N = 1008$, $L = 8$ for high rate ($K = 672$) and low rate ($K = 256$) polar codes with varying IC iterations $J$ for GFDM.

We want to explore the performance implications of IC in Fig. 5.13 and compare the FER performance of OFDM and GFDM. The simulations are carried out with a polar code with $N = 1008$ and $L = 8$ and varying information word size $K$ as discussed in Chapter 3. OFDM yields roughly 0.5 dB better performance than GFDM. The non-orthogonal GFDM design has a negative impact on performance that can not be recovered with IC under perfect channel conditions, i.e. no offsets or time or frequency mismatch. The simulations for low $R \approx \frac{1}{4}$ and high $R \approx \frac{2}{3}$ coderates reveal that GFDM only benefits from IC at high coderates. Furthermore, these simulations reveal that $J = 2$ suffices to improve performance. More IC iterations do not improve performance.

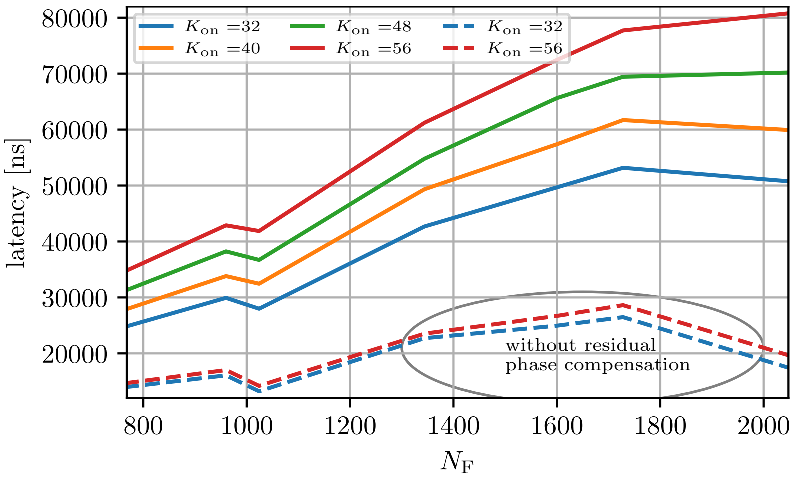


**Figure 5.14:** GFDM demodulator latencies with varying number of active subcarriers, with and without residual phase compensation, $J = 1, K_{\text{s}} = 64$.

GFDM demodulation may incorporate multiple processing steps. Besides demodulation, equalization, and IC, it is possible to integrate residual phase compensation as part of the IC process. In Fig. 5.14, we present benchmark results with and without residual phase compensation for $J = 1$ with $K_{\text{s}} = 64$. Since residual phase compensation requires detected receive symbols, we require $J \geq 1$. Given the latency impact of residual phase compensation, we conclude that it should only be an option in cases where it is impossible to improve equalization.

Further, latency increases with an increasing number of active subcarriers $K_{\text{on}}$ because IC is computed for active subcarriers only. However, the impact of an increased number of active subcarriers is small, especially without residual phase compensation.

### 5.6.4 Cyclic prefix

After modulation, a Cyclic Prefix (CP), a CS, and windowing is applied. The corresponding benchmark results in Fig. 5.15 include only one CP, CS and windowing operation per frame of size $N_{\text{F}}$. This implies that increasing the frame size $N_{\text{F}}$, while the other parameters are unchanged, only results in more elements that need to be copied. In case of OFDM, this operation must be computed per timeslot and thus might have a larger impact on

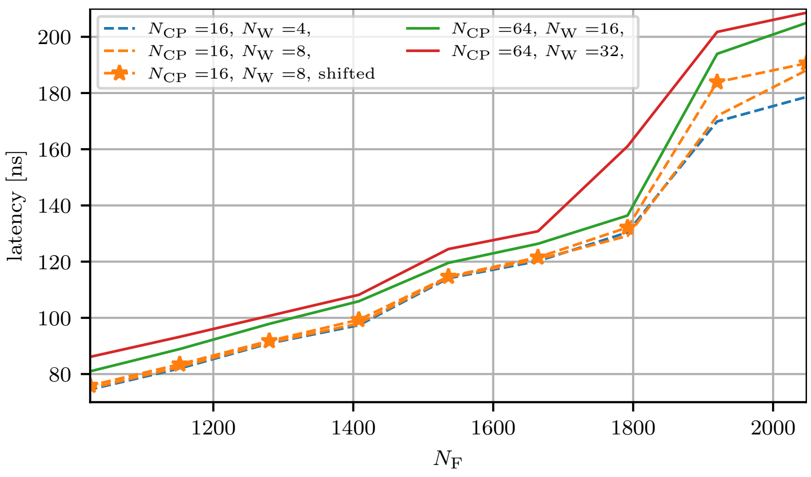


**Figure 5.15:** CP insertion and windowing, $K_{\mathrm{s}} = 128$.

latency. The benchmark reveals two major sources for latencies. First, the copy operation dominates latency. Second, a larger transition window at the beginning and the end of a block increases latency. A cyclic shift, indicated with `shifted`, leaves latency mostly unaffected. The details on cyclic shifts are discussed in Sec. 6.1.1 where we discuss Cyclic Delay Diversity (CDD) for Joint Transmission (JT).

### 5.6.5 Channel estimation

Aside from AWGN channel simulations, a receiver requires equalization. In order to perform equalization, channel estimation needs to provide proper channel estimates. In our work, we focus on preamble-based channel estimation.

Channel tap estimation is a two step process. First, the preamble is used to estimate per subcarrier channel taps together with filtering to reduce noise. Second, we need to interpolate to obtain $N_{\mathrm{F}}$ taps from a $2K_{\mathrm{s}}$ preamble. The latency benchmarks in Fig. 5.16 indicate that we experience a fixed offset due to the input size that is fixed to $2K_{\mathrm{s}}$. Again, we observe higher latency if the largest prime in $K_{\mathrm{s}}$ is greater two. In comparison to modulation and demodulation, we conclude that channel tap estimation has a minor impact on latency.

Accurate LLR calculation requires SNR values and per subcarrier CNR values. Fig. 5.17 reveals that SNR estimation latency is input driven. Again, the largest impact is due to the number of subcarriers $K_{\mathrm{s}}$ but the number

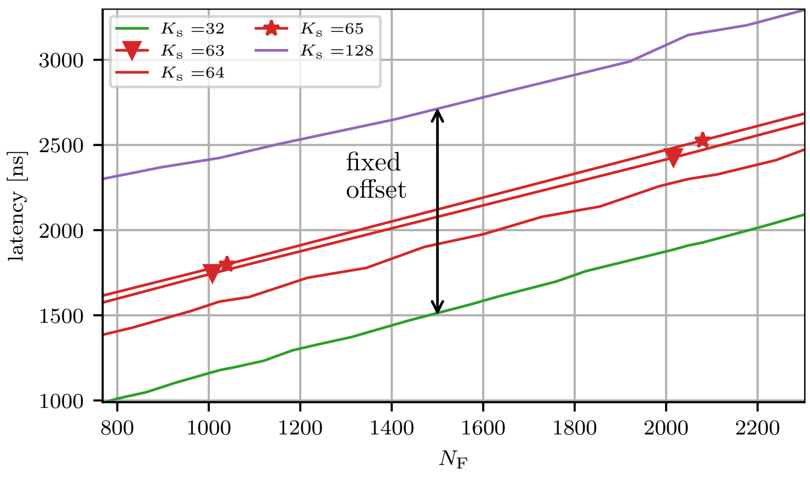


**Figure 5.16:** Preamble-based channel tap estimation.

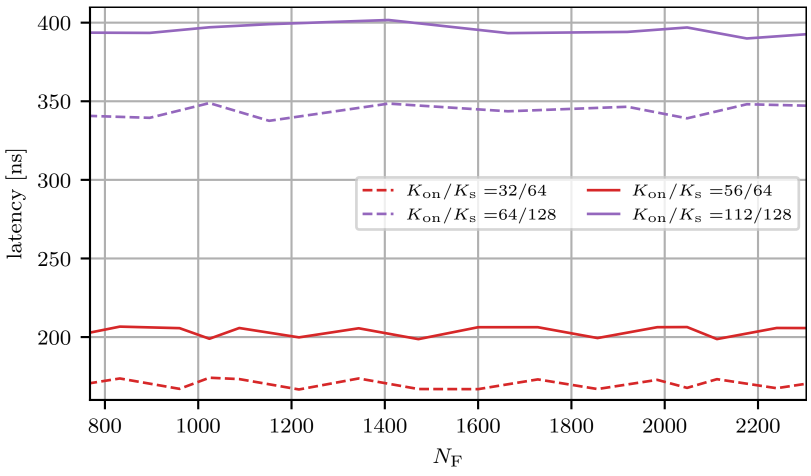


**Figure 5.17:** Preamble-based SNR and CNR estimation.

of active subcarriers $K_{on}$, i.e. the necessary number of values we need to compute, impacts latency as well. To summarize our SNR estimation results, we note that it has a minor impact on latency compared to channel estimation. Compared to demodulation, SNR estimation latency is negligible.

### 5.6.6 System evaluation

We consider several examples for multicarrier systems and their overall DSP latency. In Fig. 5.18, several graphs with either differing $K_\mathrm{s}$ or varying $K_\mathrm{t}$ are shown. In accordance with our conclusions in Sec. 5.6.2, we limit the largest prime to $7^\bullet$ because larger prime values show a significant latency increase. Moreover, we keep the overlap factor $N_\mathrm{OV} = 2$ because the conclusions in Fig. 5.9 indicate that the overlap factor $N_\mathrm{OV} = 2$ is sufficient. Finally, the results in Fig. 5.13 show that $J = 2$ IC iterations are sufficient to improve error correction performance even in conjunction with high rate codes. First, we compare the *FFT* results, that correspond to OFDM

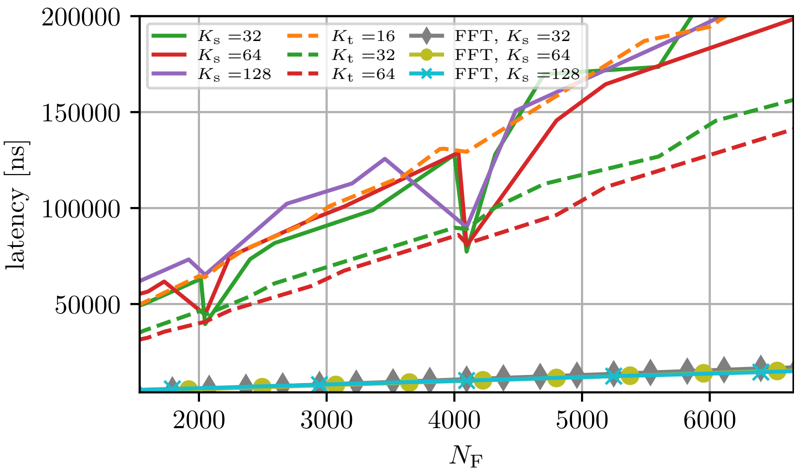


**Figure 5.18:** Full transceiver with $K_\mathrm{on} = \frac{7K_\mathrm{s}}{8}$, $N_\mathrm{CP} = K_\mathrm{s}/2$, $N_\mathrm{CS} = K_\mathrm{s}/4$, FFT size $N_\mathrm{FFT} = K_\mathrm{s}$ for OFDM, and either $K_\mathrm{t}$ or $K_\mathrm{s}$ are a power of two for GFDM.

systems, with the other results for GFDM where $K_\mathrm{t} = \{16, 32, 64\}$ and $K_\mathrm{s} = \{32, 64, 128\}$. By comparing the latency results at $N_\mathrm{F} = 3000$, we conclude that OFDM systems with multiple timeslots are computationally lightweight with latencies below 10 000 ns in comparison to GFDM systems with latencies above 60 000 ns. Second, we compare GFDM configurations where either $K_\mathrm{t} = 2^\times$ with $\{16, 32, 64\}$ or $K_\mathrm{s} = 2^\times$ with $\{32, 64, 128\}$. Here, we conclude that a GFDM system where the number of timeslots $K_\mathrm{t}$ is kept to a power of two $2^\times$ yields lower latency. However, the GFDM configuration with $K_\mathrm{t} = 16$ exhibits higher latencies than most configurations with $K_\mathrm{s} = 2^\times$. Here, we conclude that it is advantageous to keep all FFT sizes $N_\mathrm{FFT} \geq 32$ in a GFDM system.

Finally, we conclude that a low-latency GFDM system is possible that offers advantages in terms of OOB emissions and efficiency. Further, current and future low latency applications are well supported with system latencies well below 1 ms.

## 5.7 Summary

We started with a discussion of the theory behind multicarrier modulation and continued with an in-depth analysis of their latency impact in an optimized software implementation. The presented GFDM and OFDM transceiver runs completely in software [DRKK22]. The common code-base enables simulations as well as field tests for rapid prototyping. We focused on flexible modulation for GFDM and OFDM where we discuss their specific advantages an disadvantages. The implementation is capable of providing low latency signal processing for high data rate broadband I4.0 URLLC SDR environments. The performance indicates that General Purpose Processor (GPP) hardware is a suitable option to be considered for low-latency systems.

### 5.7.1 Contribution

The main contribution of this chapter is an investigation of latencies and performance implications introduced in the multicarrier modulation DSP chain of our GFDM SDR implementation [DRKK22]. To this end, the chapter incorporates an extension to our prior works [DBD17a, DBD$^+$17b]. Our extensive set of latency benchmarks and error correction performance simulations demonstrates the practicability of our solution with system latencies well below 1 ms in the context of URLLC and I4.0 low-latency applications. In contrast, other works on GFDM such as [DMG$^+$15] focus on hardware implementations without a focus on latency measurements. We identified the multicarrier modulation and demodulation steps to be critical while all other DSP operations have a minor latency impact. Prior works [FJ05] on FFT implementation benchmarks focus on $2^\times$ FFT transforms, while we analyze the impact of more primes on latency. Furthermore, this analysis showed that the number of timeslots in a GFDM system should be kept to a power of two contrary to the number of subcarriers where the impact on latency is lighter. Moreover, GFDM in conjunction with FEC does not benefit from more than two IC iterations at high code rates $R$ as discussed in Fig. 5.13. At a low code rate, a GFDM system does not benefit from IC at all.

# Chapter 6

# Cloud radio access network with functional splits

In order to deliver small packets with ultra high reliability and low latency, it is crucial to leverage diversity. Even more so in industrial communication systems for Industry 4.0 (I4.0) applications that require Ultra Reliable Low Latency Communication (URLLC) where challenging wireless channel properties with deep fades must be expected [DHC$^+$19]. In this chapter we propose a Cloud Radio Access Network (Cloud RAN) architecture with Functional Split (FS) options to leverage channel diversity from distributed Radio Access Points (RAPs) in conjunction with centralization benefits.

First, we consider polar code based Forward Error Correction (FEC) as discussed in Chapter 3. Second, the considered symbol mappings are discussed in Chapter 4. Third, multicarrier modulation as presented in Chapter 5 forms the basis for the analysis in this chapter. Here we extend this analysis to multiple distributed RAPs in an Ultra Dense Network (UDN) that jointly provide connectivity to leverage diversity and thus boost reliability. We aim to exploit spatial diversity with multiple RAPs for pre-processing and Joint Transmission (JT) and Joint Reception (JR) on a cloud platform. At a transmitter, we consider JT via distributed RAPs with Cyclic Delay Diversity (CDD) to boost reliability, and thus Frame-Error-Rate (FER) performance [BM06]. At a receiver, we achieve spatial diversity in an UDN where distributed RAPs observe the same message through different channels. Moreover, we explore an option to partition Radio Access Network (RAN) functions via FSs to flexibly balance fronthaul data rates and JR gains to fully leverage spatial diversity. We specifically propose an intra-PHYsical layer (intra-PHY) FS to reduce fronthaul data rate requirements while leveraging JR processing gains. Full I&Q sample forwarding from a RAP to a cloud platform requires a very high fronthaul data rate that may be unavailable.

Our contribution in this chapter is a Cloud Radio Access Network (Cloud RAN) architecture with distributed Access Points (APs) and FSs where we investigate its performance. Additionally, this chapter includes an extension to our paper [DMB+20] where we use the results from [KY14] to reduce the fronthaul data rates in a Cloud RAN deployment with distributed APs. This analysis receives an extension with quantizers for high Signal-to-Noise-Ratios (SNRs) in frequency-selective Rayleigh fading channels and investigations with multiple distributed RAPs. Without quantization, floating point (fp-32) values with high resolution, and thus high rate, require potentially prohibitively high data rates for forwarding. Information Bottleneck Method (IBM) quantization and potential alternatives may provide good FER performance with low quantization resolution to minimize the fronthaul data rate. We investigate these options in this chapter. Further, it is sufficient to design $N_Q = 32$ quantizers for different SNRs over a sufficiently large SNR range to reap all diversity in a frequency-selective Rayleigh fading scenario. In contrast, our previous work [DMB+20] suggested that this may be insufficient while we show how to overcome the inherent difficulties in Rayleigh fading channels. Finally, the JT extension with Cyclic Delay Diversity (CDD) shows the potential to improve FER performance, while Joint Decoding (JD) shows promising benefits to drastically improve FER performance with $N_{\text{RAP}} = \{2, 4\}$ RAPs and IBM quantization beyond the improvements available with JT.

## 6.1 Cloud RAN with distributed radio access points

In order to fulfill URLLC requirements a Cloud RAN setup is considered to enable high reliability. In Fig. 6.1 we outline our physical architecture. We assume an industrial environment with Automated Guided Vehicles (AGVs), or otherwise mobile units, that require URLLC. The AGVs are wirelessly connected to a RAN to enable mobility. While the communication system on the AGVs is required to be simple, the RAN may deploy advanced communication technologies to provide the highest possible reliability and lowest latency.

Before we dive deeper into Cloud RAN functional components, we want to discuss our understanding of the term cloud and by extension cloud computing based on the National Institute of Standards and Technology

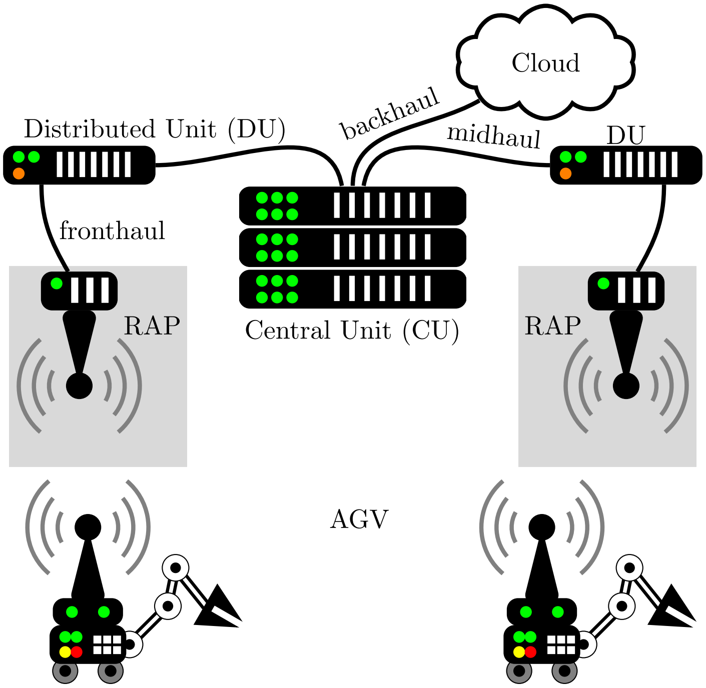


**Figure 6.1:** Physical Cloud RAN example

(NIST) definition [MG11]. Most importantly, we expect ubiquitous compute resources to deploy our Software-Defined Radio (SDR) solutions with minimal effort. The cloud computing model presented in [MG11] considers essential characteristics, service models and deployment models. We focus on the RAN, while a core network provides internet connectivity and management services and is possibly deployed on a public cloud platform via a software as a service model. The RAN may be deployed on a private cloud with a platform as a service model where accelerators for specific Digital Signal Processing (DSP) processing tasks are available. A platform, or software, provider offers the infrastructure and maintenance thereof. Thus, we need to investigate the performance of our solution and the resulting cloud requirements. This investigation includes Cloud RAN considerations for required capabilities in distributed as well as centralized setups.

According to [3GP17], the RAN contains three components, Centralized Unit (CU), Distributed Unit (DU), and Radio Unit (RU) that we consider to be part of a cloud. The cloud concept introduces the flexibility to rapidly distribute RAN functions among resources on demand via FSs [LCC19, 3GP17]. While a Cloud RAN is under heavy load, it acquires additional resources to cater all users, while it releases resources as soon as the load drops. Moreover, we distinguish between RAPs and a cloud platform. The cloud platform provides compute power for a RAN on demand, while a RAP includes dedicated radio hardware and may provide some specialized compute functions.

The Centralized Unit (CU) in Fig. 6.1 performs higher layer tasks and is connected to the network core on a public cloud platform via a backhaul. In our Cloud RAN scenario the CU always runs on a General Purpose Processor (GPP) cloud platform. Each CU is connected to one or more Distributed Units (DUs) via a midhaul. The DU performs lower layer DSP in order to reduce the required midhaul data rate requirements. Finally, the Radio Unit (RU), also known as Remote Radio Unit (RRU) or Remote Radio Head (RRH), is connected to the DU via a fronthaul and provides Radio Frequency (RF) functionality such as Analog-to-Digital Converters (ADCs), amplifiers and antennas. An example of a RU may be a Universal Software Radio Peripheral (USRP). A RU is preferably located at, or close to, an actual antenna site while a CU is preferably located such that it can be used to pool resources. We consider a DU to be co-located with either a RU in a RAP or a CU on a cloud platform. However, a DU may be a physically separate unit located in a different location [BBK22].

In order to leverage spatial diversity, with Joint Transmission (JT) and Joint Reception (JR), it is advantageous to execute most, or all, signal processing jointly on a cloud platform to maximize performance [CCY+15, PHV15, AVK19, LCC19]. Thus, a CU and DU are co-located on a cloud platform and all DSP is performed in software including the whole PHYsical layer (PHY). However, this approach puts a huge data rate requirement on a fronthaul and might be infeasible due to hardware restrictions. A FS has major implications in terms of system capabilities and requirements. We discuss possible mixtures for distributed functionality execution and centralized execution shortly.

In literature, numerous possible FS options are considered [3GP17, LCC19], though the usefulness of some options is disputed in literature [AVK19]. Fig. 6.2 presents a selection of these FSs with focus on relevant functions in our work. These FSs are enumerated and this notation is preserved but irrelevant options are omitted. A FS separates logical functions that can be flexibly located in a CU, a DU, or a RU [LCC19].

Split 1 is the highest layer FS between RAN and core network with network layer functionality. With this option, we are unable to leverage spatial diversity or flexibility gains to reduce cost [LCC19, RBD+14]. The

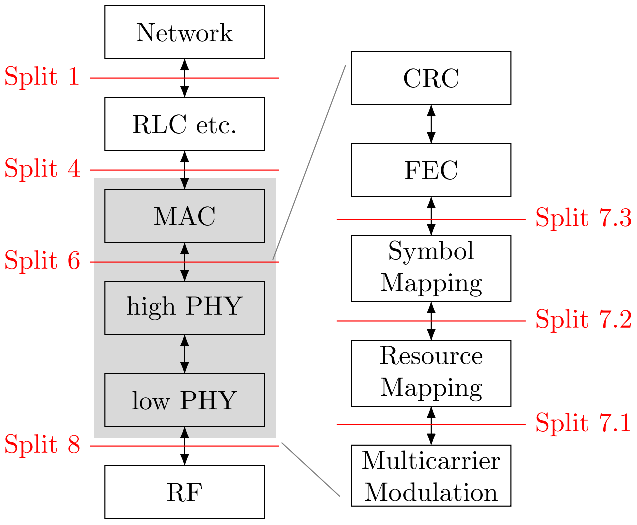


**Figure 6.2:** Logical Cloud RAN functions with selected FS options.

lowest FS is Split 8 where only a radio frontend is part of a RAP and thus all Inphase and Quadrature component (I&Q) samples are forwarded to a cloud platform. Consequently, all DSP is performed in software including the whole PHY. While this split preserves all possible options to leverage spatial diversity, it puts a huge data rate requirement on a fronthaul that is often impossible to meet, especially in case of rate-limited fronthauls. Split 4 considers all PHY and Medium-Access-Control (MAC) processing to be part of a RAP while Split 6 only locates PHY processing in RAP. Split 5 would be an intra-MAC split.

In our work we focus on intra-PHY splits. Here, three options, Split 7.1, Split 7.2, and Split 7.3, are known in literature [LCC19]. We focus on Split 7.3 in the uplink for Joint Decoding (JD) because this FS separates FEC and symbol mapping. In contrast, other literature does not consider the Split 7.3 in the uplink [3GP17]. In each RAP we perform distributed low-PHYsical layer (low-PHY) processing, including synchronization and multicarrier demodulation. Next, a symbol demapper produces Log-Likelihood Ratios (LLRs) with high resolution that we forward to our cloud platform which are then used for FEC decoding. We propose to replace the symbol demapper with an IBM quantizer in the RAPs to drastically reduce forwarded bit resolution and thus fronthaul data rates while preserving relevant information for JD. On the cloud platform we use Look-Up-Tables (LUTs) to obtain representative LLRs for joint high-PHYsical layer (high-PHY) processing, including decoding. This JD concept is evaluated with polar codes that are discussed in Chapter 3.

### 6.1.1 Joint transmission

The focus of our work is on URLLC and thus measures to boost reliability. Joint Transmission (JT) with Cyclic Delay Diversity (CDD) is a simple and effective measure to improve transmit diversity [BM06]. CDD aids to combat deep fades, and thus channel outages, in frequency-selective Rayleigh fading channels. The CDD approach promises to be a computationally lightweight solution at the transmitter and does not require additional processing at the receiver since we consider CDD for fading channels where equalization is strictly required [BM06]. In Sec. 5.6.4, we included a benchmark for a combination of cyclic shifts and Cyclic Prefix (CP) insertion that shows that CDD has a minor, if any, effect on latency.

We consider a modulated vector $\boldsymbol{y}$ for transmission at $N_{\mathrm{RAP}}$ distributed RAPs with usually one antenna each $N_{\mathrm{T}}/N_{\mathrm{RAP}} = 1$. Some channel realizations may constitute a deep fade and thus, it is preferable to send a copy of $\boldsymbol{y}$ at spatially diverse locations. However, we risk destructive interference at the receiver in case multiple signals arrive at the receiver through good channel realizations with differing phase. Therefore, we introduce CDD to circumvent this issue.

The transmit vector for the $i$th RAP

$$\boldsymbol{y}_i = \boldsymbol{P}_i \boldsymbol{y} \tag{6.1}$$

is a cyclically shifted version of $\boldsymbol{y}$ via a permutation matrix $\boldsymbol{P}_i$. $\boldsymbol{P}_i$ is an identity matrix $\boldsymbol{I}_{N_{\mathrm{y}}}$ with cyclically shifted rows by cyclic shift $N_{\mathrm{CD}i}$. Thus the first row $[1, 0, \ldots, 0]$ of $\boldsymbol{I}_{N_{\mathrm{y}}}$ is the $N_{\mathrm{CD}i}$th row in $\boldsymbol{P}$. For example if cyclic shift $N_{\mathrm{CD}i} = 1$, the corresponding permutation matrix

$$\boldsymbol{P}_i = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.2}$$

can be found by moving the last row $[0, 0, 0, 1]$ in $\boldsymbol{I}_4$ to the first row. $N_{\mathrm{CD}i}$ is expected to be a multiple $\{0, 1, 2, \ldots\}$ of $N_{\mathrm{CD}}$.

CDD impacts the effective channel even under ideal conditions. Effectively, the received signal

$$\tilde{\boldsymbol{y}} = \sum_{i=0}^{N_{\mathrm{RAP}}} \boldsymbol{y}_i \tag{6.3}$$

through an ideal channel is convolved with an artificial channel $\boldsymbol{h} = [1, 0, \ldots, 1, 0 \ldots, 0, 1]$ in time domain. These channel taps are all zero except at the $N_{\mathrm{RAP}}$ positions where it is 1. These positions are at indices $N_{\mathrm{CD}i}$ [BM06]. We expect that a receiver mitigates this effect via channel estimation and subsequent equalization.

## 6.1.2 Joint reception

The idea of Joint Reception (JR) in our Cloud RAN scenario is to perform high-PHY processing on a cloud platform. A visual presentation of the idea is shown in Fig. 6.3 with intra-PHY Split 7.3. At the cloud platform, we perform JD, Cyclic Redundancy Check (CRC) check and all upper layer tasks while low-PHY processing at $N_{\mathrm{RAP}}$ RAPs is performed distributedly to lower fronthaul data rates.

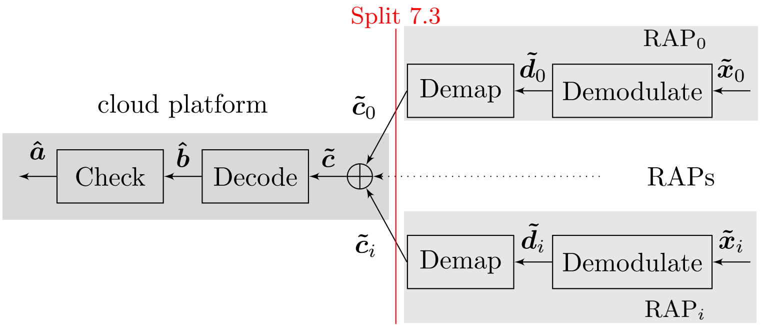

Split 7.3
RAP$_0$
cloud platform
$\tilde{\boldsymbol{c}}_0$
Demap
$\tilde{\boldsymbol{d}}_0$
Demodulate
$\tilde{\boldsymbol{x}}_0$
$\hat{\boldsymbol{a}}$
Check
$\hat{\boldsymbol{b}}$
Decode
$\tilde{\boldsymbol{c}}$
RAPs
$\tilde{\boldsymbol{c}}_i$
Demap
$\tilde{\boldsymbol{d}}_i$
Demodulate
$\tilde{\boldsymbol{x}}_i$
RAP$_i$

**Figure 6.3:** Receive flowgraph with Joint Reception (JR) on a cloud platform, forwarding over a fronthaul, and low-PHY processing on multiple distributed RAPs.

While it is possible to have multiple antennas at each RAP, we usually assume one antenna per RAP. Many systems use multiple antennas per RAP. However, we focus on large scale diversity and thus, only one antenna per RAP. To leverage spatial diversity we require independent signal paths. The most effective approach to ensure spatial diversity is to physically separate all antennas. Large scale fading considerations, e.g. spatial coherence, emphasize the importance of this approach as discussed in Sec. 2.2.

A transmitter sends a signal $\boldsymbol{x}$ that is received by distributed RAPs. These distributed RAPs observe received signals $\tilde{\boldsymbol{x}}_i$ that are received through independent and identically distributed (i.i.d.) Rayleigh channels as discussed in Sec. 2.2. After low-PHY processing, all RAPs forward their resulting received codeword $\tilde{\boldsymbol{c}}_i$ to a cloud platform for high-PHY processing. Before, we continue with our conventional signal processing, we compute a combined received codeword

$$\tilde{\boldsymbol{c}} = \sum_{i=0}^{N_{\mathrm{RAP}}-1} \tilde{\boldsymbol{c}}_i \tag{6.4}$$

from the forwarded pre-processed i.i.d. receive codewords $\tilde{\boldsymbol{c}}_i$. In other words, the combined received codeword $\tilde{\boldsymbol{c}}$ is the sum of all $\tilde{\boldsymbol{c}}_i$ from every RAP. In the log-domain, the summation of LLRs corresponds to the multiplication of their probabilities in the linear domain. Further, we assume i.i.d. values for all observations. Thus, our approach in (6.4) is optimal [RU08]. Besides this summation processing step, all DSP is carried out as discussed in previous chapters. This holds true for all RAPs, however, each RAP processes a signal in parallel as illustrated in Fig. 6.3.

## 6.2 Quantization

Sec. 6.1.2 introduces JR where we forward LLRs over the FS via a rate-limited fronthaul to reap the benefits of JD on a cloud platform. Now, the aim of our investigation is to reveal how Information Bottleneck Method (IBM) can be employed to reduce fronthaul rate requirements [KY14, HCSM21]. We add uniform quantization in Sec. 6.2.2 and fixed LLR quantization in Sec. 6.2.3 as further options, while we consider floating point LLR FER performance as our FER benchmark that we want to achieve. Fig. 6.4 illustrates the concept. At each RAP a quantizer compresses a received vector $\tilde{\boldsymbol{d}}$ to lower the required fronthaul rate.

For now, we assume Quadrature Phase Shift Keying (QPSK) symbols and replace the demapper with a quantizer that produces a vector $\tilde{\boldsymbol{q}}_i$ of quantized values. At the cloud platform, a LUT transforms these quantized values to representative LLRs $\tilde{\boldsymbol{c}}_i$. These LLRs are treated in the exact same fashion as discussed in Sec. 6.1.2 with (6.4) but are sourced from a reduced pool of values. In [MWD21], the authors use a different decoder that operates on quantized values while we integrate quantization into an available system without any modifications to the processing chain.

The quantizer is SNR dependent. Therefore, each RAP requires accurate SNR information to select a quantizer according to the current SNR value. In case of Rayleigh fading channels and multicarrier systems, each RAP

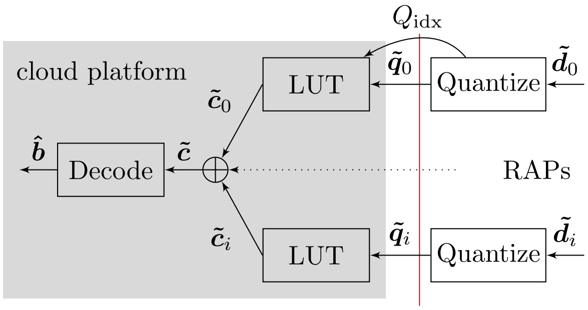

$Q_{\text{idx}}$
cloud platform
LUT
LUT
Decode
Quantize
Quantize
RAPs
$\tilde{\boldsymbol{q}}_0$
$\tilde{\boldsymbol{q}}_i$
$\tilde{\boldsymbol{d}}_0$
$\tilde{\boldsymbol{d}}_i$
$\tilde{\boldsymbol{c}}_0$
$\tilde{\boldsymbol{c}}_i$
$\tilde{\boldsymbol{c}}$
$\hat{\boldsymbol{b}}$

**Figure 6.4:** Joint Reception (JR) flowgraph on a cloud platform featuring IBM quantizers at each RAP with a LUT at the cloud platform for every connected RAP.

needs accurate Carrier-to-Noise-Ratio (CNR) information, as discussed in (2.18), to perform this selection process per subcarrier. Overall, a RAP has $N_Q$ SNR dependent options to select from and therefore needs to forward the index of the selected quantizer $Q_{\text{idx}} \in \{0, \ldots, N_Q - 1\}$ along with the quantized values. In case of frequency-selective channels, this information needs to be forwarded per subcarrier depending on the CNR. In [DMB+20], we demonstrated that 4 bit quantization is sufficient to arrive at a negligible FER performance loss. Here, we extend these findings by considering multiple RAPs and a wider range of designed quantizers.

### 6.2.1 Information bottleneck based quantization

In the field of IBM research, we focus on the special case of Deterministic Quantizers (DQs) that are known in literature to maximize mutual information [KY14]. Furthermore, an efficient algorithm computes optimal Sequential Deterministic Quantizers (SDQs) for real valued Amplitude Shift Keyings (ASKs) that maximize mutual information at a limited bit rate [HCSM21]. The focus of this work is on complex Additive White Gaussian Noise (AWGN) channels under the assumption of i.i.d. real and imaginary parts. In this work, a quantizer processes the real and imaginary parts independently. The interested reader may refer to e.g. [HWD20] for IBM quantization. The receiver flowgraph in Fig. 6.5 illustrates the IBM quantizer setup.

We restrict this discussion to binary input, e.g. $d \in \{1, -1\} = \mathcal{A}_{\text{C}}$, with equiprobable symbols $p(d) = \frac{1}{2}$ [KY14]. Considering an AWGN channel, we

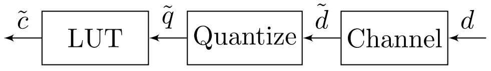


**Figure 6.5:** IBM quantizer receiver flowgraph

denote the conditional probability density function

$$p\left(\tilde{d}|d\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{|\tilde{d}-d|^2}{2\sigma^2}\right) \tag{6.5}$$

for the received symbols. The offline task at hand is to find an optimal quantizer $\tilde{q} = Q^\star\left(\tilde{d}\right)$ that maximizes mutual information $I_{\mathrm{BICM}}\left(\tilde{\mathrm{q}};\mathrm{d}\right)$ between the quantized symbols $\tilde{q} \in \mathcal{Q}$ and the transmit symbols $d$ with

$$Q^\star = \underset{Q}{\operatorname{argmax}}\, I_{\mathrm{BICM}}\left(\tilde{\mathrm{q}};\mathrm{d}\right) \text{ subject to } |\mathcal{Q}| = N_{\mathrm{q}}\,. \tag{6.6}$$

$I_{\mathrm{BICM}}\left(\bullet\right)$ is discussed in Appendix B. $\tilde{q}$ is a label for an interval that is bounded by a lower and upper threshold. In (6.11), we consider representative LLR values for each interval. We enumerate these labels $\mathcal{Q} = \{0, 1, \ldots, N_{\mathrm{q}} - 1\}$. The cardinality of $\mathcal{Q}$ determines the number of bits $\lceil\log_2|\mathcal{Q}|\rceil$ for a specific quantizer $Q^\star$. Therefore, we need to find the set $\mathcal{A}_q$ of $N_{\mathrm{q}} - 1$ thresholds that optimally partitions $\mathbb{R}$ into $N_{\mathrm{q}}$ intervals.

An algorithm to design the optimal quantizer is presented in [KY14]. This algorithm discussion, which is reiterated here, starts with a fine-grained, uniformly discretized version of $p\left(\tilde{d}|d\right)$ with $N_{\mathrm{qd}}$ elements. The presented algorithm requires this initial discretization since the algorithm is designed for discrete-output channels. We obtain $N_{\mathrm{qd}} - 1$ finite bounds with $\{-N_\sigma\sigma\min(\mathcal{A}_{\mathrm{C}}), \ldots, N_\sigma\sigma\max(\mathcal{A}_{\mathrm{C}})\}$ to define $N_{\mathrm{qd}}$ intervals. Typically, we consider the two-sigma-interval with $N_\sigma = 2$ because it covers over 95 % probability that a value falls into this interval.

**Partial mutual information** In accordance with [KY14] we first define partial mutual information

$$\iota\left(a' \to a\right) = \sum_{d\in\mathcal{A}_{\mathrm{C}}} p\left(d\right) \sum_{y=a'+1}^{a} p\left(y|d\right) \log \frac{\sum_{y'=a'+1}^{a} p\left(y'|d\right)}{\sum_{d'\in\mathcal{A}_{\mathrm{C}}} p\left(d'\right) \sum_{y'=a'+1}^{a} p\left(y'|d'\right)} \tag{6.7}$$

to compute the contribution of individual intervals to the overall mutual information. It is important to note that $a$, $a'$, $y$, and $y'$ are all indices.

Moreover, $p\left(\bullet|\bullet\right)$ denotes a conditional probability mass function and the ratios $\frac{p(d'|1)}{p(d'|-1)}$ are sorted in ascending order [KY14].

The values $\iota\left(a' \to a\right)$ are pre-computed for each combination of $a' \in \{0, 1, , \ldots, N_{\mathrm{qd}} - 1\}$ and $a \in \{a' + 1, \ldots, t\}$ with $t = \min\left(a' + 1 + N_{\mathrm{qd}} - N_{\mathrm{q}}, N_{\mathrm{qd}}\right)$. In essence, we compute the partial mutual information contribution of every possible quantization with $N_{\mathrm{q}}$ intervals that each contain a contiguous set of integers. Next, we search for the quantization that yields the maximum overall mutual information.

**Dynamic state computation** We recursively compute the state value

$$S_q\left(a\right) = \max_{a'}\left( S_{q-1}\left(a'\right) + \iota\left(a' \to a\right)\right) \tag{6.8}$$

that maximizes the sum of partial mutual information terms when we cluster $a$ elements into $q$ bins. We start with $S_0\left(0\right) = 0$ and compute state values for all $q \in \{1, \ldots, N_{\mathrm{q}}\}$ and $a \in \{q, \ldots, q + N_{\mathrm{qd}} - N_{\mathrm{q}}\}$. Further, we need to store the index

$$h_q\left(a\right) = \underset{a'}{\mathrm{argmax}}\left( S_{q-1}\left(a'\right) + \iota\left(a' \to a\right)\right) \tag{6.9}$$

that yields the just computed maximum where $a'$ is in the interval $[q-1, a-1]$. Obviously, both computations can be efficiently combined to yield the state value and its corresponding index.

**Optimal index backtrace** Finally, we need to backtrace through the just computed state values and indices to find the optimal quantizer $Q^\star$ with its optimal bound indices $a^\star$. The backtrace starts at index ${a^\star}_{N_{\mathrm{q}}}$. Then we traverse $q \in \{N_{\mathrm{q}} - 1, N_{\mathrm{q}} - 2, \ldots, 1\}$ in descending order and find the optimal quantizer bound indices

$${a^\star}_q = h_{q+1}\left({a^\star}_{q+1}\right). \tag{6.10}$$

These indices are then used to find the $N_{\mathrm{q}} - 1$ finite quantizer bounds $\mathcal{A}_q$ that we use for the optimal quantizer $Q^\star$.

We illustrate the results depending on SNR in Fig. 6.6 for the computed finite bounds. With ascending SNR values, the bounds tend towards zero. As long as we assume that all transmit symbols are equiprobable, the optimal decision bound is zero. Further, the most critical region in terms of information preservation is around zero. Thus, we expect the bounds to be symmetric around zero.

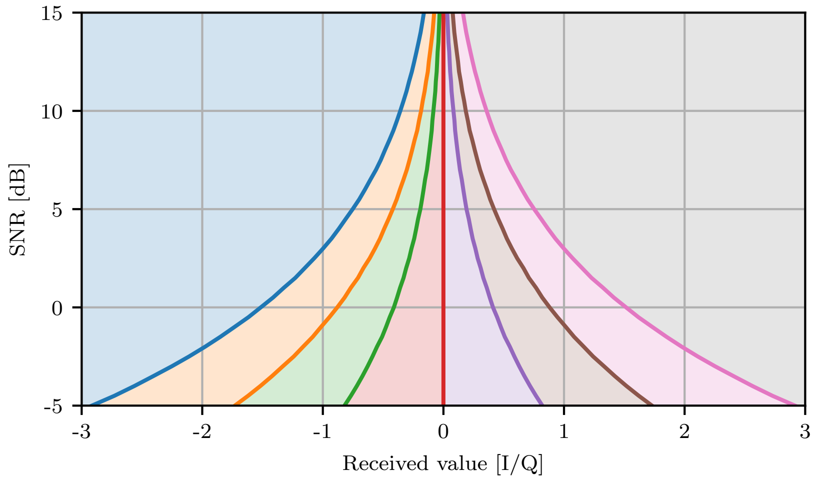


**Figure 6.6:** SNR dependent quantizer bounds for $\mathfrak{Re}\{\cdot\}$ / $\mathfrak{Im}\{\cdot\}$ QPSK components with $N_{\mathrm{q}} = 8$, $N_{\mathrm{qd}} = 1024$.

**Log-likelihood ratio computation** is facilitated by computing the values of conditional probability mass function $p\,(\bullet|\bullet)$ over the intervals defined by the found quantizer bounds. Thus, we compute the representative LLR

$$\tilde{c} = \log\left(\frac{p\left({a^\star}_q < \tilde{d} \leq {a^\star}_{q+1}|c=0\right)}{p\left({a^\star}_q < \tilde{d} \leq {a^\star}_{q+1}|c=1\right)}\right) \tag{6.11}$$

where we replace indices by their respective bounds. These values are required for the LUT to transform a quantized label $\tilde{q}$ into a representative LLR $\tilde{c}$ for further processing.

In Fig. 6.7, we observe the resulting representative LLR values for the computed intervals depending on SNR that are depicted in Fig. 6.6. With ascending SNR, and thus a better channel quality, we observe higher LLR values. Although the bounds tend towards zero, the LLRs grow faster.

## 6.2.2 Uniform quantization

We choose the finite quantizer bounds over the same interval we use for IBM based quantization, i.e. from $\{-N_\sigma\sigma\min(\mathcal{A}_{\mathrm{C}}), \ldots, N_\sigma\sigma\max(\mathcal{A}_{\mathrm{C}})\}$ with $N_{\mathrm{q}} - 1$ bounds. However, we choose uniformly distributed, or equidistant, bounds over this interval [DMB+20]. In order to obtain representative LLRs, we use the same approach we discuss in Sec. 6.2.1.

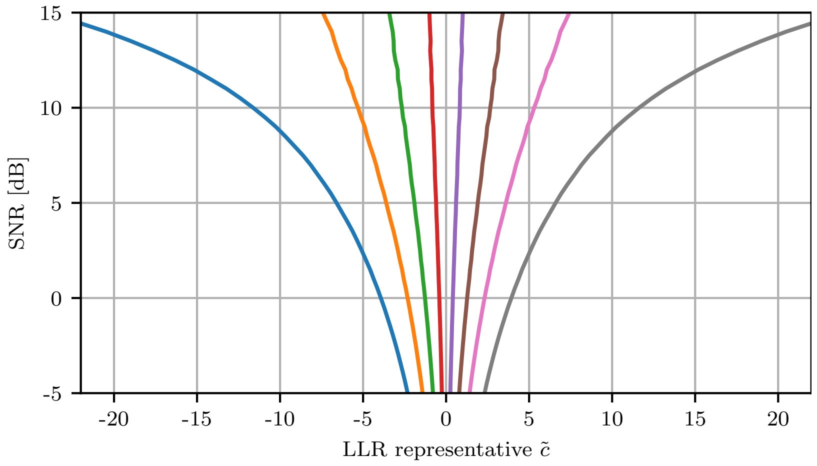


**Figure 6.7:** SNR dependent quantizer LLR representatives for $\mathfrak{Re}\{\cdot\}$ / $\mathfrak{Im}\{\cdot\}$ QPSK components with $N_{\mathrm{q}} = 8$, $N_{\mathrm{qd}} = 1024$. Each LLR curve corresponds to an interval in Fig. 6.6.

### 6.2.3 Fixed log-likelihood ratio quantization

Instead of SNR dependent quantization, we consider fixed LLR quantization (LLRq). We define a LLR value interval $\pm\mathfrak{B}$ and choose $N_{\mathrm{q}} - 1$ bounds. We choose the corresponding representative LLR values at equidistant positions over the interval $\pm\mathfrak{B}$.

## 6.3 Numerical cloud RAN evaluation

We investigate how data compression according to the IBM on the fronthaul link affects system performance for Generalized Frequency Division Multiplexing (GFDM) as well as OFDM in our proposed Cloud RAN architecture with JT and JR. We focus on the impact of coarser quantization on system performance first where we extend the results from our prior work [DMB+20]. For a rate limited fronthaul, coarser, or lower, quantization contributes to lower latency because of the reduced data rate. We focus on a fixed setup for all GFDM and Orthogonal Frequency Division Multiplexing (OFDM) simulations with $K_{\mathrm{s}} = 64$ subcarriers, $K_{\mathrm{on}} = 50$ active subcarriers, $K_{\mathrm{t}} = 5$ timeslots, $K = 256$ information bits, $N = 500$ coded bits, and QPSK modulation. We show that 3 bit IBM quantization already achieves close to floating point performance in frequency-selective channels. We extend our prior work [DMB+20] in three key aspects, namely quantizers for higher

SNRs, an additional quantizer, and multi RAP setups, and therefore multiple antennas. Hence, we investigate the performance of our FS Cloud RAN setup with distributed RAPs and JT and JR as introduced in Sec. 6.1.

## 6.3.1 Additive white Gaussian noise

Quantization strategies with different levels of quantization in an AWGN channel serve as a base line. Fig. 6.8 shows a comparison of OFDM and

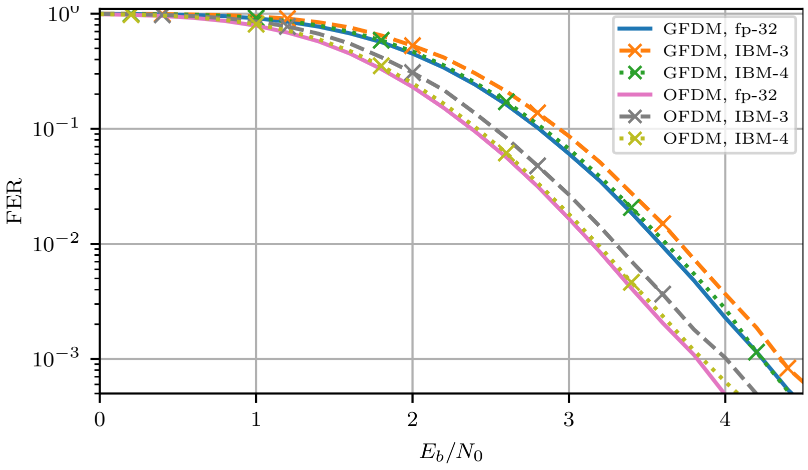


**Figure 6.8:** AWGN FER performance of IBM and fp-32 for OFDM and GFDM with $K_\mathrm{t} = 5$, $K_\mathrm{s} = 64$, $K_\mathrm{on} = 50$, $K = 256$, $N = 500$, and QPSK.

GFDM systems transmitted over an AWGN channel. Here, GFDM introduces a 0.5 dB performance loss due to non-orthogonality compared to OFDM. Further, 4 bit IBM quantization delivers fp-32 performance while 3 bit quantization only incurs a minor loss of about 0.1 dB. Thus, we conclude that IBM quantization is a suitable method to drastically reduce the required fronthaul bit rate in our Cloud RAN setup.

Fig. 6.9 continues with an in-depth analysis of different quantizer strategies. A fp-32 LLR result serves as a base line. While we focus on OFDM for clarity in this case, we note that GFDM results yield the same characteristics. The fixed LLR quantization (LLRq) shows the worst performance and requires at least 6 bit to close the performance gap. A 3 bit fixed LLRq incurs a performance loss of about 1 dB. However, the fixed LLRq strategy does not require any knowledge about the chosen quantizer in contrast to the other strategies. The uniform quantizer shows improved performance compared to the fixed LLRq and closes the performance gap with 5 bit while 4 bit

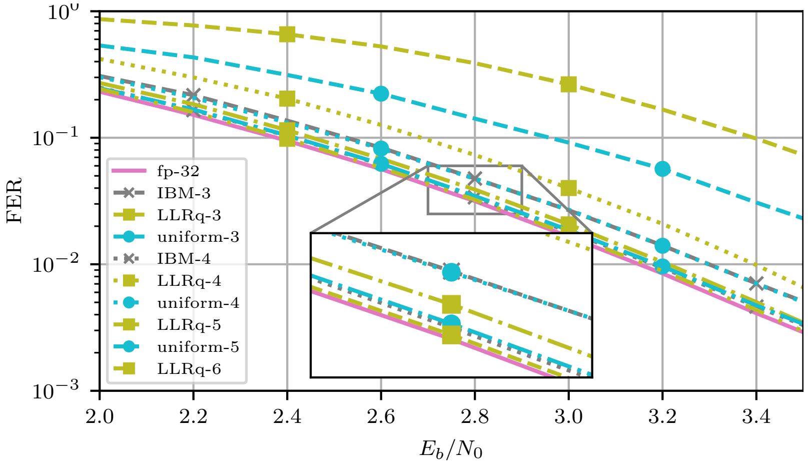


**Figure 6.9:** FER performance for fp-32, IBM, LLRq, and uniform quantizers with varying resolution for OFDM with $K_\mathrm{t} = 5$, $K_\mathrm{s} = 64$, $K_\mathrm{on} = 50$, $K = 256$, $N = 500$, and QPSK.

uniform quantization is on par with 3 bit IBM quantization. A coarse, 3 bit uniform quantizer incurs a performance loss of about 0.6 dB which is an improvement over the fixed LLRq strategy but still far from the floating point (fp-32) performance. Moreover, a uniform quantizer needs to signal the chosen quantizer for a specific SNR. We observe that the IBM quantizer is able to close the performance gap with only 4 bit. A 3 bit IBM quantizer only incurs a performance penalty of about 0.1 dB and hence may still be a viable option.

IBM quantization yields better error correction performance at lower quantization resolution and thus outperforms the other quantization strategies by at least 1 bit of resolution. Therefore, IBM quantization may be a viable option for FS Cloud RAN deployments.

## 6.3.2 Frequency-selective Rayleigh fading

After we established a base line in Sec. 6.3.1, our investigation continues in frequency-selective Rayleigh fading channels. We investigate FER performance for 3 bit and 4 bit IBM quantizers in Fig. 6.10. Here, a set of $N_Q = 100$ quantizers designed for SNRs in the range $-10$ dB to 40 dB in 0.5 dB steps is available. We mostly observe the same behavior that we already discussed in the AWGN case in Fig. 6.8. A 3 bit IBM quantizer incurs a FER performance penalty of about 0.1 dB while a 4 bit IBM quantizer is

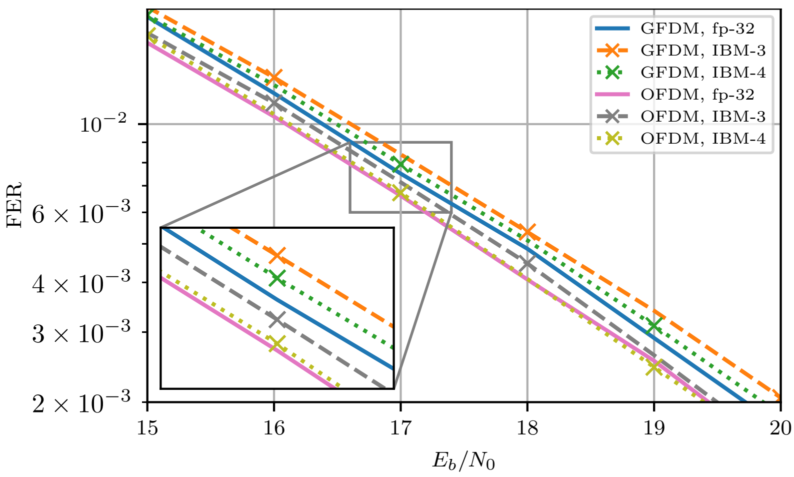


**Figure 6.10:** Frequency-selective Rayleigh fading FER performance of 3 bit and 4 bit IBM and floating point quantizers for GFDM and OFDM.

mostly able to achieve fp-32 performance. However, we note that the IBM quantizer in conjunction with GFDM performs slightly worse in terms of FER performance in comparison to a GFDM system with a fp-32 demapper. Besides, we observe the same non-orthogonality performance loss of about 0.5 dB for GFDM in comparison to OFDM that we already observed in our AWGN simulations. Still, GFDM offers potentially higher spectral efficiency and lower latency as discussed in Chapter 5.

The results in Fig. 6.10 are different from our previously published results in [DMB+20] where we speculated that the insufficient range of available quantizers causes a FER performance loss. This work confirms our conjecture and emphasizes the importance of a sufficient range of IBM quantizers to achieve good FER performance.

Similarly to our AWGN investigation in Fig. 6.9, Fig. 6.11 shows frequency-selective Rayleigh fading channel FER performance results for fp-32, IBM, LLRq, and uniform quantizers with 3 bit to 5 bit quantization resolution for GFDM. A 3 bit uniform quantizer as well as a 4 bit fixed quantizer incur a FER performance penalty of about 0.7 dB. Further, a 4 bit uniform quantizer delivers FER performance on par with a 3 bit IBM quantizer. A 4 bit IBM quantizer is able to outperform all other quantization strategies and almost closes the gap to fp-32 FER performance.

Our FS Cloud RAN system with distributed RAPs and rate limited fronthauls requires to minimize the overhead caused by signaling indices

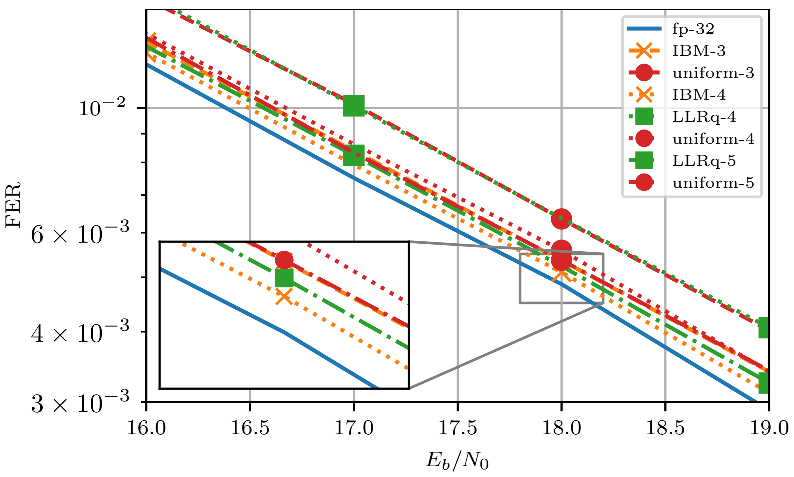


**Figure 6.11:** Frequency-selective Rayleigh fading FER performance for fp-32, IBM, LLRq, and uniform quantizers with 3 bit to 5 bit quantization resolution for GFDM.

$Q_{\text{idx}}$ of the employed quantizers. We assume block fading and thus, only need to signal a new set of used quantizers when the selected quantizers change due to changing CNRs. However, we need to forward the index $Q_{\text{idx}}$ of the used quantizer per active subcarrier $K_{\text{on}}$ because of the changing CNR per subcarrier. This may cause an additional overhead of $K_{\text{on}} \log_2 (N_Q)$ for every received frame. Hence, minimizing the number of quantizers $N_Q$ minimizes the corresponding overhead. In our scenario with $K_{\text{on}} = 50$ active subcarriers, $K_{\text{t}} = 5$ timeslots and QPSK mapping, a 6 bit fixed LLR quantizer requires to convey $6K_{\text{on}}K_{\text{t}}M = 3000$ bit over a fronthaul while a 4 bit IBM quantizer only requires 2000 bit and $K_{\text{on}} \log_2 (N_Q)$ used quantizer signaling. With $N_Q = 32$ quantizers this results in 250 bit of signaling overhead and thus a total of 2250 bit are conveyed over the fronthaul.

We investigate FER performance depending on the number of quantizers $N_Q$ in Fig. 6.12. As a first result, we conclude that $N_Q = 32$ quantizers are sufficient. A further reduction in $N_Q$ results in a negligible FER performance loss for $N_Q = 16$ quantizers and a minor performance loss for $N_Q = 8$ quantizers. With $N_Q = \{2, 4\}$ quantizers we observe a more significant performance loss. Considering the additional signaling overhead, we still reap a significant rate reduction with IBM quantization even with $N_Q = 32$ quantizers while maintaining fp-32 FER performance. Here, we assume that each subcarrier potentially requires a different quantizer. However, with

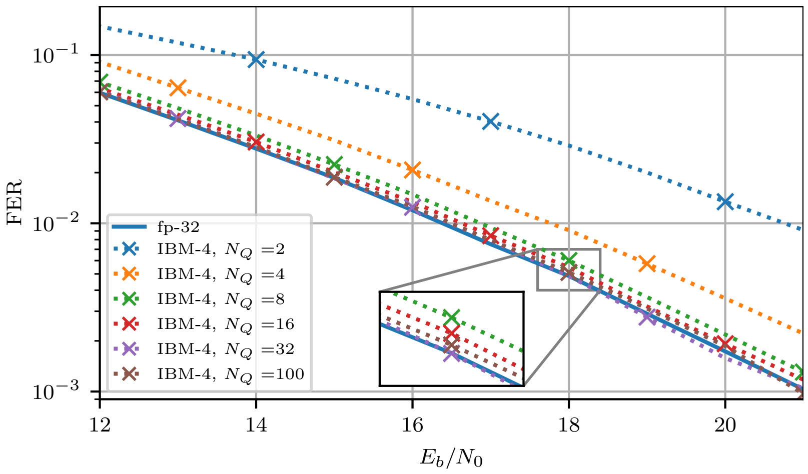


**Figure 6.12:** Varying number of quantizers $N_Q$ for GFDM.

higher coherence bandwidth, we may be able to use the same quantizer for a group of subcarriers. This approach would further reduce overhead.

### 6.3.3 Multiple distributed radio access points

Now we investigate the performance of our proposed setup with multiple RAPs $N_{\mathrm{RAP}}$. Here, we assume $N_{\mathrm{ant}} = 1$ antenna for a mobile device and one antenna per RAP $N_{\mathrm{ant}}/N_{\mathrm{RAP}} = 1$. Naturally the number of transmit $N_{\mathrm{T}}$ and receive $N_{\mathrm{R}}$ antennas changes depending on the number of $N_{\mathrm{RAP}}$ RAPs and the downlink or uplink scenario.

In Fig. 6.13 we consider an increasing number of RAPs $N_{\mathrm{RAP}}$, and thus receive antennas $N_{\mathrm{R}}$, for reception to leverage Joint Decoding (JD). The AWGN result is included as a benchmark. All simulations are conducted under the assumption that all channels are i.i.d.. However, in case of multiple antennas per RAP this assumption might be invalidated.

A second RAP yields a 7 dB FER performance improvement at FER $= 10^{-3}$ and four RAPs with $N_{\mathrm{R}} = 4$ yield another 4 dB improvement. Doubling the number of receive antennas further to $N_{\mathrm{R}} = 8$ yields improvements with declining benefit while the required hardware and complexity increase steeply. This tendency is exacerbated with even more receive antennas $N_{\mathrm{R}} = \{16, 32\}$ with diminishing FER performance gains. Given physical constraints to distribute RAPs, we assume that more than $N_{\mathrm{R}} = 4$ receive antennas are infeasible with $N_{\mathrm{RAP}} = 4$ under the condition that $N_{\mathrm{R}}/N_{\mathrm{RAP}} = 1$. The

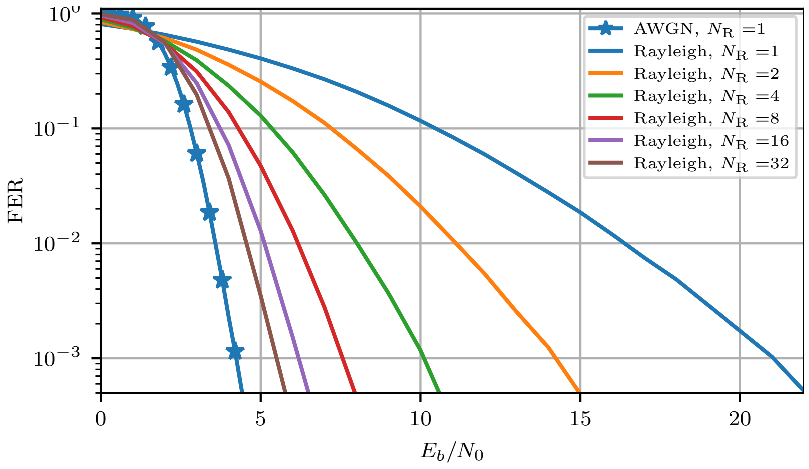


**Figure 6.13:** GFDM system with increasing number of receive antennas $N_R$ for fp-32 with $K_t = 5$, $K_s = 64$, $K_{on} = 50$, $K = 256$, $N = 500$, and QPSK.

results confirm that our proposed FS Cloud RAN setup is able to improve FER performance and thus reliability significantly.

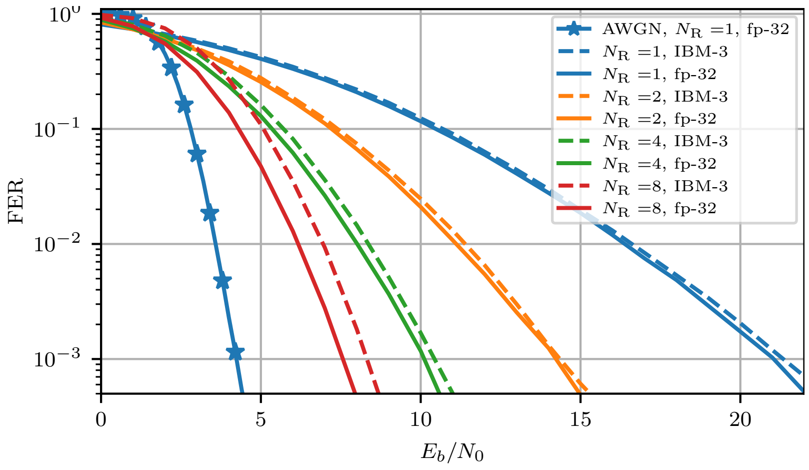


**Figure 6.14:** GFDM system with increasing number of receive antennas $N_R$ and IBM quantizers with $K_t = 5$, $K_s = 64$, $K_{on} = 50$, $K = 256$, $N = 500$, and QPSK.

Fig. 6.14 shows the results to how suitable IBM quantization is in conjunction with JD. First, we note that quantization is feasible in a multi RAP setup. However, one may notice that an increasing number of receive antennas $N_{\mathrm{R}}$ is more susceptible to quantization. With $N_{\mathrm{R}} = 8$, we recognize a FER performance loss of about 1 dB. As a conclusion, we state that IBM quantization is suitable for feasible multi RAP setups while a large number of RAPs would incur a FER performance penalty.

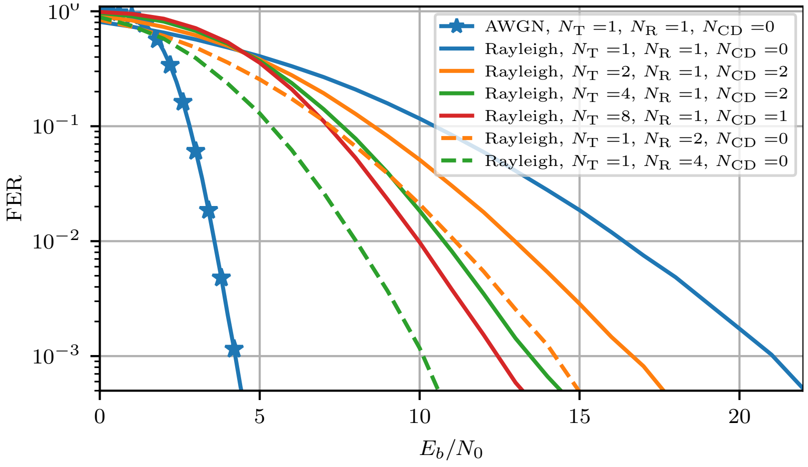


**Figure 6.15:** GFDM system with increasing number of transmit antennas $N_{\mathrm{T}}$ with $K_{\mathrm{t}} = 5$, $K_{\mathrm{s}} = 64$, $K_{\mathrm{on}} = 50$, $K = 256$, $N = 500$, and QPSK.

For Joint Transmission (JT), we consider Fig. 6.15. By increasing the number of transmit antennas from $N_{\mathrm{T}} = 1$ to $N_{\mathrm{T}} = 2$, we observe a FER performance improvement of about 4 dB at FER $= 10^{-3}$. A $N_{\mathrm{T}} = 4$ transmit antenna setup reaps another improvement of about 4 dB. Doubling the number of transmit antennas again to $N_{\mathrm{T}} = 8$, yields an improvement of about 1 dB. Again, all simulations are conducted under the assumption of i.i.d. Rayleigh channels from the transmitter to the receiver. To put things into perspective, we added the $N_{\mathrm{R}} = \{2, 4\}$ results for comparison. Thus, we conclude that JT is a viable option to improve performance but increasing receive diversity with JD promises higher gains by a large margin.

## 6.4 Summary

We discussed a Functional Split (FS) Cloud Radio Access Network (Cloud RAN) system with distributed RAPs and IBM quantization in this chapter.

To this end, we introduced our interpretation of Cloud RAN with FSs within the PHY layer. Further, the introduction of IBM quantization together with alternatives paves the way to enable Cloud RAN deployments with rate limited fronthauls. Then, a thorough analysis of these quantization approaches provides insight into the FER performance in AWGN and frequency selective Rayleigh fading channels. Finally, we considered Joint Transmission (JT) with CDD and Joint Reception (JR) with Joint Decoding (JD) to boost system reliability with a distributed RAP setup. Finally, we conclude that a Cloud RAN deployment with FS with IBM quantization and distributed RAPs is capable of boosting reliability tremendously for URLLC and I4.0 applications.

### 6.4.1 Contribution

Our contribution in this chapter includes an extension to our paper [DMB+20] where we use the results from [KY14] to reduce the fronthaul rate in a Cloud RAN deployment with distributed APs. We extend our prior work [DMB+20] in three key aspects, namely quantizers for higher SNRs, an additional quantizer approach, and a multiple distributed AP setup with JD and JT. We showed that 4 bit IBM quantization is sufficient to close the performance gap to a fp-32 implementation while 3 bit IBM quantization only incurs a minor performance penalty. All other quantization strategies cause FER performance to deteriorate more significantly. Further, $N_Q = 32$ quantizers for different SNRs over a sufficient range suffice to reap all diversity in a frequency-selective Rayleigh fading scenario. In contrast, previous works suggested that this may be insufficient while we show how to overcome the inherent difficulties in Rayleigh fading channels [DMB+20]. Finally, the JT extension with CDD shows the potential to improve FER performance, while JD shows promising benefits to drastically improve FER performance with $N_{\text{RAP}} = \{2, 4\}$ RAPs and IBM quantization beyond the improvements available with JT.

Chapter 7

# Medium access control

Mission-critical Closed-Loop-Control (CLC) applications exhibit periodic deterministic communication behavior with short packets. Industrial systems with these URLLC requirements may consist of tens to hundreds of devices. These applications pose new Quality-of-Service (QoS) challenges on wireless communication systems because they require ultra low latency which bars the application of MAC re-transmissions to improve reliability. Further, these industrial systems impose a real-time deadline constraint on a wireless communication system. Packet loss at the application level occurs not only due to failed packet reception but also due to a missed real-time deadline.

FER performance is often used as a Key Performance Indicator (KPI) in communications engineering for system evaluation. In contrast, automation engineers consider Mean Time To Failure (MTTF) as their KPI [DMW+17]. Thus, the authors in [DMW+17] proposed a method to obtain a relation between those KPIs. Hence, they elaborate how URLLC system requirements result in extremely low FER requirements which may lead to prohibitively high resource consumption.

In [3GP19a], the KPI for URLLC systems is availability. Availability is defined as the probability a communication system fulfills a set of QoS requirements at the application level [3GP19a]. Throughout [3GP19a], we find a typical 99.999 % availability requirement. The consolidated KPIs under the term availability are packet size, latency, survival time, number of devices and maximum supported mobile unit speed.

In essence, industrial applications are very sensitive to burst errors but can tolerate single packet loss. Burst errors are Consecutive Erroneous Packets (CEPs) on a single link that exceed the required survival time. If a communication system fails to successfully convey a new packet within a users' survival time, the industrial application may require an emergency halt. Hence, we propose to shift the focus from sum-rate maximization to burst error minimization as our KPI to boost availability [WE08, DBD20].

Our extensive burst error analysis of State-of-the-Art (SotA) Scheduling and Resource Allocation (S&RA) strategies shows that any dynamic Resource Allocation (RA) outperforms a static RA by a large margin. We build this chapter on two prior publications that we extend here [DBD19, DBD20].

**Link abstraction** Accurate Link Abstraction (LA) models are required for MAC or system level simulations to evaluate new S&RA algorithms that target URLLC requirements [PPM18]. The employed Link Abstraction (LA) models use a channel realization and given PHY and transform this knowledge into a link characterization, preferably a FER [LKK12]. Otherwise accurate system level simulations need to simulate the whole PHY to determine packet loss instead of employing a simple, accurate Link Abstraction (LA) model. In [DBD19] we investigate if and how Effective SNR Mapping (ESM) approaches can be used to characterize short polar-coded packets with Bit-Interleaved Coded Modulation (BICM) in OFDM systems at low FERs.

In this chapter, we contribute an extension to our previous work [DBD19] and a re-evaluation of these Exponential Effective SNR Mapping (EESM) and Mutual Information Effective SNR Mapping (MIESM) results for short packet LA. The addition of Average Effective SNR Mapping (AESM) constitutes one extension that we contribute in this chapter. We compare new error measures for FER curve fitting in order to find optimal ESM adjustment factors for low FERs. Typically, Mean Square Error (MSE) is used as an error measure [LKK12], but according to our findings this neglects small FER values which are of special interest for industrial radio systems. Thus, we propose a relative error measure to find optimal adjustment factors. We present simulation results that show how susceptible adjustment factors are to different system parameters. Finally, we demonstrate that accurate Link Abstraction (LA) is possible for the target system even at the desired working point. Ergo we contribute a suitable Modulation and Coding Scheme (MCS) set for our S&RA simulations that enables us to study S&RA in frequency selective block fading channels with polar codes and multicarrier modulation [Ars15].

**Scheduling and resource allocation** Based on our Link Abstraction (LA) model, we investigate how SotA Scheduling and Resource Allocation (S&RA) strategies perform with respect to burst errors in scenarios with short packets, low latency and low FER requirements. Furthermore, large scale and small scale fading contribute to burst errors and need to be taken into account [ETS18b]. We re-evaluate our prior work [DBD20] and contribute an extension to these investigations to meet URLLC requirements, e.g. fixed MCS setups [ADE+19]. We suggest new delay-sensitive approaches to S&RA that improve burst error resilience and conclude that it is important to shift the focus from sum-rate maximization to burst error minimization. Moreover, our investigation reveals the benefits and trade-offs between different scheduling and RA schemes. Further, recommendations are devised for efficient RA for URLLC systems based on burst error minimization for different scenarios that allow for QoS with Mean Time To Failure (MTTF) in mind. Any dynamic RA outperforms a static RA by a large margin under resource constraints. Finally, these investigations demonstrate that delay-sensitive scheduling as well as RA yield superior results in terms of burst error performance.

## 7.1 System view

We want to investigate S&RA for wireless industrial communication systems with accurate LA models. The PHY technologies include polar codes in Chapter 3, symbol mapping in Chapter 4 and multicarrier modulation in Chapter 5. Adaptive Modulation and Coding (AMC) enables efficient resource usage with multiple MCSs. These investigations require accurate Link Abstraction (LA) for precise S&RA decisions as well as fast system simulations.

A MCS combines the selected coderate $R$ and mapping order $M$ for a frame. A lower MCS index indicates a lower effective rate $R_{\text{eff}}$ and thus higher robustness against errors. However, a lower MCS requires more resources to convey the same amount of information bits $K$. Now, the task at hand is to schedule and allocate a fixed set of shared resources to a set of users $\mathcal{U}$ with $|\mathcal{U}| = N_U$.

Here we discuss our view on resources from a system level perspective. Then, we discuss further information that is required for our proposed S&RA algorithms. Finally, we discuss the information flow in our S&RA system. We already published to prior works where we build upon [DBD19, DBD20].

### 7.1.1 Resources

The available resources depicted in Fig. 7.1 are organized in slots on a resource grid as introduced in Sec. 5.2. Then, Scheduling and Resource Allocation (S&RA) operate on slot granularity that is sometimes called Transmission Time Interval (TTI). We assume a multicarrier system where the bandwidth $B_{\text{s}}$ is divided into $N_{\text{RB}}$ resource blocks with $N_{\text{sc,RB}}$ subcarriers per resource block. Similarly, a slot is divided in time into $K_\mathrm{t}$ timeslots, also referred to as OFDM symbols.

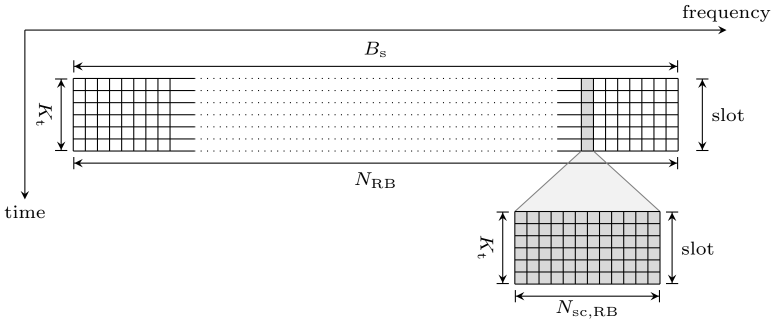


**Figure 7.1:** Resource grid with one slot, comprised of $K_\mathrm{t}$ timeslots, and bandwidth $B_\mathrm{s}$ divided into resource blocks $N_\mathrm{RB}$ and further divided into subcarriers.

Hence, the multicarrier system offers a resource set $\mathcal{R}$ that consists of $K_\mathrm{t}$ timeslots with $K_\mathrm{s} = N_\mathrm{RB}N_\mathrm{sc,RB}$ subcarriers. Consequently, each slot consists of $K_\mathrm{t}N_\mathrm{RB}N_\mathrm{sc,RB} = K_\mathrm{t}K_\mathrm{s}$ elements. We may assign an arbitrary subset $\mathcal{R}_u \subseteq \mathcal{R}$ to a user for transmission in a slot. Each user experiences a different i.i.d. frequency-selective block fading channel as discussed in Sec. 2.2. Thus, we assume that the channel is constant for one slot. Consecutive slots may experience correlated channel realizations according to our channel model from Sec. 2.2. Moreover, a frequency-flat channel per resource block is assumed by 5th Generation New Radio (5G NR) [ETS18a]. The Channel State Information (CSI) for each user comprises CNRs for all resource blocks that is available for S&RA.

The resource grid is scarce and it is shared among all users. The S&RA task is to allocate elements from the resource grid to users such that all users meet their communication requirements. In order to fulfill this task, we require LA to accurately, and efficiently, determine the most suitable resources to allocate to each user depending on the selected MCS.

We focus on Time-Division-Duplex (TDD) to organize uplink and downlink transmissions because we consider a I4.0 scenario in the 3.8 GHz band [ETS18d]. However, our approaches apply to Frequency-Division-Duplex (FDD) as well. In 5G NR terminology, a slot spans over 14 timeslots while a minislot consists of fewer timeslots [SWD$^+$18]. We apply the term slot for a TTI regardless of the number of timeslots.

### 7.1.2 System flowgraph

We consider Frequency-Division-Multiplex (FDM) and Time-Division-Multiplex (TDM) to allocate resources to users. With a FDM RA strategy entire resource blocks are allocated to a user [DPS18]. In contrast, a TDM RA strategy allocates a number of elements from every resource block to a user, i.e. each user is allocated a share of elements from every resource block. Different users may be allocated varying amounts of elements per resource block depending on CSI and success delay $N_{\mathrm{sd},u}$.

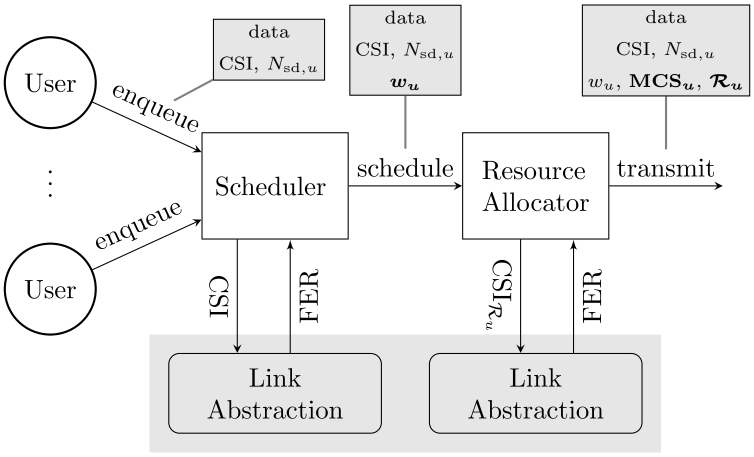


**Figure 7.2:** Scheduling and Resource Allocation (S&RA) flowgraph for one slot: Every user enqueues a new packet, the scheduler computes $w_u$ for every packet, then the allocator determines $\mathrm{MCS}_u$ and $\mathcal{R}_u$ for every user. Finally, all packets are multiplexed onto the slot resource grid and transmitted.

The system flowgraph in Fig. 7.2 illustrates the packet data flow. Our Scheduling and Resource Allocation (S&RA) flow is split into two consecutive tasks that operate on a slot. All $N_U$ users in the set of users $\mathcal{U}$ enqueue a packet with $K$ information bits, user CSI, and user success delay $N_{\mathrm{sd},u} \in \mathbb{N}^{0+}$. $\mathbb{N}^{0+}$ is the set of positive integers including zero. A $N_{\mathrm{sd},u} = 1$ indicates that a users the last transmission attempt in the previous slot by this user failed. The user success delay $N_{\mathrm{sd},u}$ is incremented in every consecutive slot with a failed transmission attempt and reset to $N_{\mathrm{sd},u} = 0$ as soon as a transmission succeeds in a slot. Thus, a larger value $N_{\mathrm{sd},u}$ indicates a longer burst error for a specific user. As mentioned before, we focus on burst error minimization in contrast to other works that focus on sum-rate maximization [WE08]. Thus, we require user success delay $N_{\mathrm{sd},u}$.

The scheduler assigns a weight $w_u$ to each packet according to the current scheduler strategy as discussed in Sec. 7.4. This step requires accurate LA to determine an accurate weight $w_u$. All packets for a slot are sorted in ascending weight order. Starting with the packet with the lowest weight $w_u$, the RA sequentially determines the current user $\mathrm{MCS}_u$ for a packet and the user allocated resources $\mathcal{R}_u$ as discussed in Sec. 7.5. After all packets are allocated or all resources are spent, transmission in a slot commences. The whole process is repeated for every slot with updated CSI and $N_{\mathrm{sd},u}$.

### 7.1.3 Packet arrival model

We restrict our investigations to mission-critical URLLC applications with periodic deterministic communication behavior [3GP19b]. This translates to CLC applications with cyclic communication behavior in the I4.0 context [UW20]. In either case, these applications expose hard real-time deadlines. Here, we discuss our packet arrival model that we assume for later evaluations.

We assume a deterministic packet arrival model where each active user enqueues a new packet for every communication cycle with duration $T_{\mathrm{cycle}}$ and bit size $K$ [DMW+17]. The cycle duration $T_{\mathrm{cycle}}$ constitutes the maximum latency $T_{\mathrm{latency}}$ after which a packet is discarded, i.e. the real-time deadline [SWD+18]. The slot duration $T_{\mathrm{slot}}$ is assumed to be $\frac{T_{\mathrm{cycle}}}{2} < T_{\mathrm{slot}} \leq T_{\mathrm{cycle}}$. Thus, only a single slot is eligible for transmission within one communication cycle and re-transmission is infeasible. Typical Motion Control (MC) applications expose cycle times in the range 0.25 ms to 2 ms [3GP19a, DMW+17].

These constraints constitute several communication system assumptions. Re-transmissions add a prohibitively high delay and thus are unavailable to improve reliability [SWD+18, FDG+18, SKMB21]. Packet segmentation would cause latency to grow beyond the maximum latency $T_{\mathrm{latency}}$ and thus their real-time deadline. If a communication system would support shorter latencies, this would be leveraged to reduce cycle duration $T_{\mathrm{cycle}}$ instead of re-transmissions [SWD+18].

Thus, we assume that all $N_U$ users enqueue a new packet with size $K$ for every slot. Hence, the considered S&RA strategies need to accommodate $N_U K$ bit in every slot, while unsuccessfully transmitted packets are discarded because their real-time deadline is exceeded. The industrial wireless communication systems are expected to operate in the 3.7 GHz to 3.8 GHz band with TDD [SWD+18, ETS18b]. The uplink and downlink are considered to reveal symmetric communication behavior [DMW+17]. Therefore the requirements on both, uplink and downlink, are the same. However, uplink and downlink packets are not interdependent, i.e. we consider end-to-end transmission in contrast to round trips [3GP19a]. Consequently, a symmetric *DUDU* TDD scheme is employed with a $1 : 1$ ratio between uplink and downlink, i.e. resources are split equally between uplink and downlink. According to [ETS18b] , we assume six downlink timeslots, or OFDM symbols, followed by two timeslots for downlink to uplink switching, and six timeslots for the uplink. Furthermore, we assume that the system introduces a suitable timing advance to enable uplink to downlink switching. Finally, the system switches between uplink and downlink and vice versa every six timeslots to enable low latency communication.

## 7.2 Link abstraction

The idea of Link Abstraction (LA) is to use the current CSI and transform it into an effective SNR and further into an expected FER as depicted in Fig. 7.3 [BAS+05, LKK12, DBD19].

The obtained effective SNR is used in S&RA algorithms to determine the user MCS $\mathrm{MCS}_u$ as well as the resources $\mathcal{R}_u$ allocated to that user. The effective SNR $\mathrm{SNR}_{\mathrm{eff}}$ represents an equivalent AWGN SNR. Further, in our simulations we use the selected $\mathrm{MCS}_u$ in conjunction with the effective SNR to obtain a single expected FER value to simulate packet loss. We extend our prior work [DBD19] with a larger range of Quadrature Amplitude Modulation (QAM) mappings.

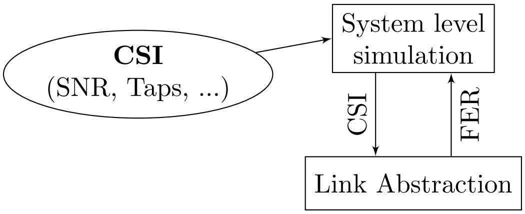


**Figure 7.3:** Link abstraction concept

The current CSI consists of path loss, large scale fading, and frequency-selective Rayleigh fading effects that are constant for the duration of a slot as described in Sec. 2.2. Since a user transmits one frame per slot, this assumption translates the channel into a block fading channel for a frame. In case of frequency-flat fading, LA reduces to obtaining a

$$\mathrm{CNR} = \frac{\left|\breve{h}\right|^2 \sigma^2}{\sigma_{\mathrm{n}}^2} \tag{7.1}$$

that we use to obtain the corresponding AWGN FER for our system level simulations. However, in case of a frequency selective Rayleigh block fading channel, LA is facilitated by transforming the per-subcarrier

$$\mathrm{CNR}_i = \frac{\left|\breve{h}_i\right|^2 \sigma^2}{\sigma_\mathrm{n}^2} \tag{7.2}$$

into an effective SNR $\mathrm{SNR}_\mathrm{eff}$ via Effective SNR Mapping (ESM). The methodology is to run extensive simulations for all desired MCSs with the described channel model, as well as AWGN simulations, and then find a suitable translation into effective SNR and further into a FER [BAS+05].

### 7.2.1 Effective SNR mapping

We consider multiple Effective SNR Mapping (ESM) approaches, namely Exponential Effective SNR Mapping (EESM) and Mutual Information Effective SNR Mapping (MIESM) [BAS+05]. Alternatively, one may use the Average Effective SNR Mapping (AESM) approach that yields results with lower accuracy. The presented approaches are described in [LKK12].

For ESM per-occupied-subcarrier CNRs are used to compute an effective SNR

$$\mathrm{SNR}_\mathrm{eff} = \beta \mathrm{f}^{-1}\left(\frac{1}{K_\mathrm{on}}\sum_{i=0}^{K_\mathrm{on}-1}\mathrm{f}\left(\frac{\mathrm{CNR}_i}{\beta}\right)\right) \tag{7.3}$$

where the function $\mathrm{f}(\ldots)$ is chosen according to the desired method for EESM or MIESM and $\beta$ is an adjustment factor, discussed in Sec. 7.2.2 [LKK12]. If one uses the identity function $\mathrm{f}(x) = x$, (7.3) boils down to Average Effective SNR Mapping (AESM), or the average SNR calculation, with

$$\mathrm{SNR}_\mathrm{eff} = \frac{1}{K_\mathrm{on}}\sum_{i=0}^{K_\mathrm{on}-1}\mathrm{CNR}_i \tag{7.4}$$

where $\beta$ cancels out.

**Exponential Effective SNR Mapping** uses $\mathrm{f}(x) = \exp(-x)$ where the equation (7.3) turns into

$$\mathrm{SNR}_\mathrm{eff} = -\beta \ln\left(\frac{1}{K_\mathrm{on}}\sum_{i=0}^{K_\mathrm{on}-1}\exp\left(-\frac{\mathrm{CNR}_i}{\beta}\right)\right) \tag{7.5}$$

and we need to find suitable adjustment factors $\beta$ depending on the current configuration. For (7.5) we require the LogSumExp (LSE) algorithm for numerical stability.

**Mutual Information Effective SNR Mapping** The function $\mathrm{f}(\ldots)$ for the MIESM approach is more involved. We need to calculate the mutual information depending on the current constellation $\mathcal{A}_{\mathrm{C}}$. The details to obtain these values are discussed in Appendix B where we arrive at (B.14) for $\mathrm{f}(x)$. We pre-compute the desired values over a wide SNR range with small step size and use the obtained data for linear interpolation. Similar to the EESM approach, we need to find suitable adjustment factors $\beta$.

### 7.2.2 Adjustment factor calibration

Our goal is to use accurate LA for our S&RA investigations. Thus, it is crucial to find the optimal adjustment factor

$$\beta_{\mathrm{opt}}^{\mathrm{MCS}} = \underset{\beta}{\operatorname{argmin}}\, \epsilon(\beta) \tag{7.6}$$

for any MCS that minimizes an error measure $\epsilon(\ldots)$ [LKK12]. The AWGN result $\mathrm{FER}_{\mathrm{AWGN}}^{\mathrm{MCS}}$ serves as a reference that we want to match with the $\mathrm{FER}_{\mathrm{ESM}}^{\mathrm{MCS}}$ curve. The error measure $\epsilon(\ldots)$ quantifies the difference between the AWGN FER curve $\mathrm{FER}_{\mathrm{AWGN}}^{\mathrm{MCS}}$ and the ESM FER curve $\mathrm{FER}_{\mathrm{ESM}}^{\mathrm{MCS}}$ over SNR or effective SNR respectively. Thus, our goal is to find a $\beta_{\mathrm{opt}}^{\mathrm{MCS}}$ where $\mathrm{FER}_{\mathrm{ESM}}^{\mathrm{MCS}}$ and $\mathrm{FER}_{\mathrm{AWGN}}^{\mathrm{MCS}}$ align for $\mathrm{SNR}_{\mathrm{eff}} = \mathrm{SNR}$.

First, we obtain FER performance simulations results for a specific MCS setup for AWGN channels as well as frequency-selective Rayleigh fading channels. Then, we calculate the effective $\mathrm{SNR}_{\mathrm{eff}}$ for all frames that we obtained for fading channels with a specific $\beta$. Afterwards, we sort all frames into bins according to their calculated $\mathrm{SNR}_{\mathrm{eff}}$ and obtain $N_{\mathrm{FER}}$ bins. We need a sufficient number of frames per bin to calculate a FER for that bin, and thus we chose the bin width to be 0.1 dB. Finally, we obtain an ESM FER $\mathrm{FER}_{\mathrm{ESM}}^{\mathrm{MCS}}(\beta)$ over SNR curve that depends on the adjustment factor $\beta$. We repeat this process to find the optimal adjustment factor $\beta_{\mathrm{opt}}^{\mathrm{MCS}}$ according to (7.6) that minimizes the error measure $\epsilon(\ldots)$.

Other works use MSE as an error measure $\epsilon_{\mathrm{MSE}}(\ldots)$ [LKK12]. Our observation is that MSE omits low FER values and thus we consider two alternatives, namely a relative error measure $\epsilon_{\mathrm{rel}}(\ldots)$ and a target FER error measure $\epsilon_{\mathrm{t}}(\ldots)$. The MSE error measure calculates

$$\epsilon_{\mathrm{MSE}}(\beta) = \sum_{i=0}^{N_{\mathrm{FER}}-1} \left|\mathrm{FER}_{\mathrm{ESM},i}^{\mathrm{MCS}}(\beta) - \mathrm{FER}_{\mathrm{AWGN},i}^{\mathrm{MCS}}\right|^2 \tag{7.7}$$

for all $N_{\mathrm{FER}}$ bins under the assumption that $\mathrm{SNR}_{\mathrm{eff}} = \mathrm{SNR}$. Similarly, the relative error measure calculates

$$\epsilon_{\text{rel}}\left(\beta\right) = \sum_{i=0}^{N_{\text{FER}}-1} \frac{\left|\text{FER}^{\text{MCS}}_{\text{ESM},i}\left(\beta\right) - \text{FER}^{\text{MCS}}_{\text{AWGN},i}\right|}{\text{FER}^{\text{MCS}}_{\text{AWGN},i}} \tag{7.8}$$

but normalizes with $\text{FER}^{\text{MCS}}_{\text{AWGN},i}$ to compensate for small FER values. Finally, we consider a target FER error measure

$$\epsilon_{\text{t}}\left(\beta\right) = \left|\text{FER}^{\text{MCS}}_{\text{ESM},t}\left(\beta\right) - \text{FER}^{\text{MCS}}_{\text{AWGN},t}\right| \tag{7.9}$$

where we choose a specific $\text{FER}^{\text{MCS}}_{\text{AWGN},t}$. Thus, we use only one evaluation point at a specific target SNR.

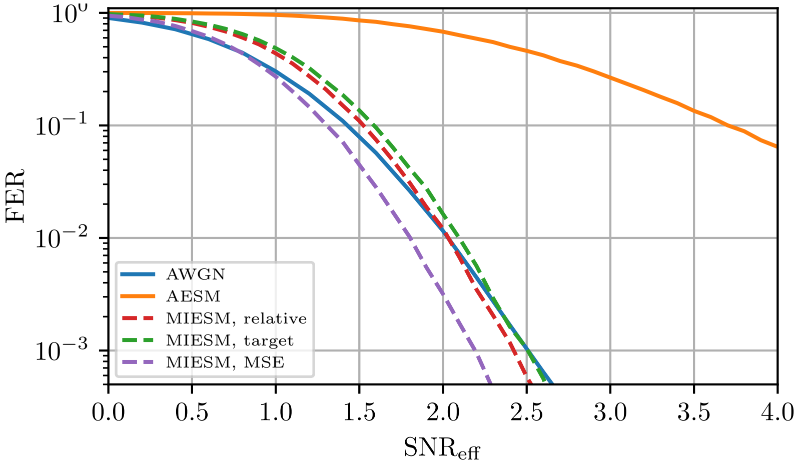


**Figure 7.4:** Exemplary ESM FER curve results for different error measures $\epsilon\left(\beta^{\text{MCS}}_{\text{opt}}\right)$ to match the target AWGN reference with OFDM, $K_{\text{t}} = 8$, $K_{\text{s}} = K_{\text{on}} = 32$, $K = 256$, $N = 512$, $L = 8$, and QPSK.

In Fig. 7.4 the resulting FER over effective SNR curves are shown. The goal is to match the FER over effective SNR curves as closely as possible to our AWGN reference. First, we observe that the AESM yields inaccurate results in comparison to the other approaches. They are not sufficiently close to the AWGN FER curve and the slope is considerably gentler. The MSE measure $\epsilon_{\text{MSE}}\left(\dots\right)$ yields FER values which are too low compared to the AWGN curve at low FER values. While the target FER measure $\epsilon_{\text{t}}\left(\dots\right)$ yields results which tend to be too high. The relative error measure $\epsilon_{\text{rel}}\left(\dots\right)$ yields results in between the other two approaches that are closer to the AWGN curve over a wide SNR range. Thus, we conclude that the relative error measure should be preferred to find optimal adjustment factors $\beta_{\text{opt}}^{\text{MCS}}$. Although we focus on a single configuration in Fig. 7.4, the results hold for other configurations with short blocks as well. Also, this investigation focuses on MIESM results but the EESM results lead to the same conclusion. Finally, the results lead to the conclusion that it is indeed possible to accurately perform LA for short codes.

## 7.3 Link abstraction evaluation

We want to analyze how well LA works for varying system parameters with polar codes and small packets. We conduct a series of simulations to evaluate the suitability of our approaches and extend the evaluations from [DBD19]. The impact of several varying parameters on the optimal adjustment factor are discussed. Finally, these investigations result in a set of MCSs selected from Fig. 7.5 with their corresponding optimal adjustment factors for use in our S&RA simulations.

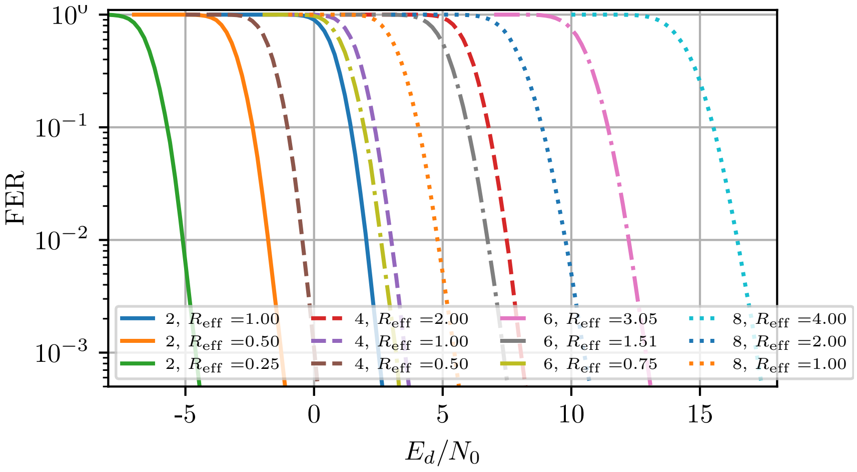


**Figure 7.5:** AWGN $E_d/N_0$ results for various mapping orders $M$ and effective rates $R_{\text{eff}}$ with $K = 256$ information bits.

All AWGN reference simulation curves are obtained by simulating $2^{17}$ frames per SNR with an early stop criterion of 8192 erroneous frames as shown in Fig. 7.5. We assume a channel as discussed in Sec. 2.2.2 with frequency-selective Rayleigh fading. For frequency-selective Rayleigh channel simulations we simulated $2^{19} = 524288$ frames for each data point, though again we use an early stop criterion with 8192 erroneous frames. The receiver assumes perfect CSI and SNR knowledge. Note that we use $\frac{E_d}{N_0}$ instead of $\frac{E_b}{N_0}$ in Fig. 7.5.

We assume a polar code with $K = 256$, $L = 8$, $\beta$-Expansion (BE) channel construction and varying $N$ as discussed in Chapter 3. The chosen constellations are discussed in Chapter 4 and we vary the mapping order $M \in \{2, 4, 6, 8\}$. Further, we assume an OFDM system with a Minimum Mean Square Error (MMSE) equalizer with $K_\mathrm{s} = K_\mathrm{on} = 32$ subcarriers by default. This is our setup for numerical evaluations and to investigate how suitable the previously presented approaches are for ESM. These evaluations focus on EESM and MIESM and exclude AESM because of its inferior accuracy as observed in Fig. 7.4.

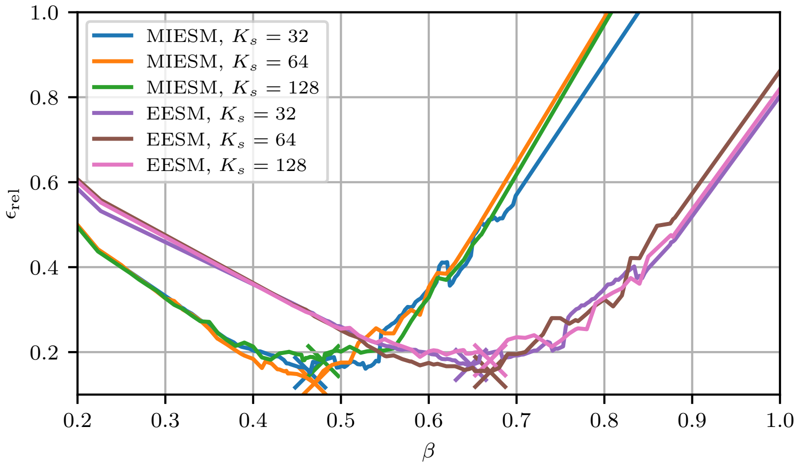


**Figure 7.6:** Relative error $\epsilon_\mathrm{rel}$ (...) over adjustment factor $\beta$ for varying subcarriers $K_\mathrm{s}$.

Our LA investigations start by comparing the results for different numbers of subcarriers $K_\mathrm{s}$ in Fig. 7.6. In both cases, EESM as well as MIESM, we observe that $K_\mathrm{s}$ has a negligible effect on the optimal adjustment factor $\beta_\mathrm{opt}^\mathrm{MCS}$, marked with an $X$. This results leads to several conclusions. It is sufficient to use a smaller number of subcarriers for our subsequent investigations, i.e. the bandwidth is divided into fewer subcarriers and thus, the subcarrier spacing increases correspondingly. Based upon this conclusion we infer that we only need to consider CNRs of active subcarriers and that it is possible to vary the number of subcarriers for our investigations. Thus, we can dynamically use a subset of all available subcarriers in our S&RA investigations.

Fig. 7.7 presents the impact of varying mapping orders $M$ on the optimal

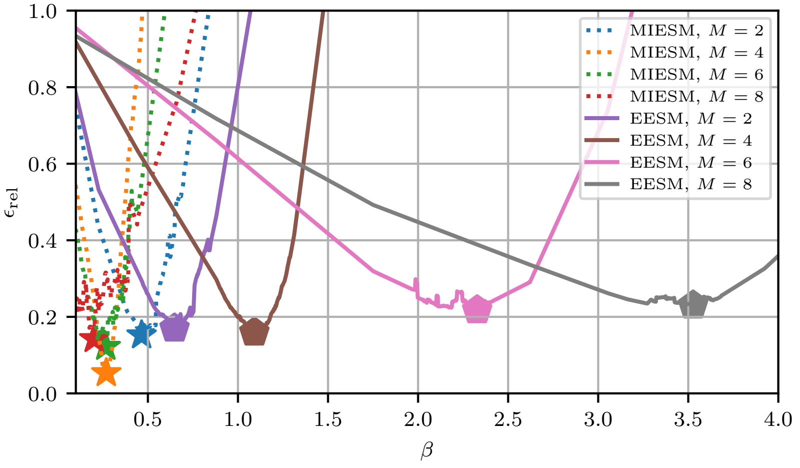


**Figure 7.7:** Relative error $\epsilon_{\text{rel}}$ (...) over adjustment factor $\beta$ for varying $M$ mapping orders with a $(2048, 256)$ code.

adjustment factor $\beta_{\text{opt}}^{\text{MCS}}$ for EESM and MIESM. Here, we incorporate $M = 6, 8$, or 64 and 256 QAM, in our simulations as an extension to [DBD19]. The first observation is that EESM yields widely varying optimal adjustment factors $\beta_{\text{opt}}^{\text{MCS}}$ for varying mapping orders $M$. MIESM is more robust against varying mapping orders and thus, shows smaller variation. Since MIESM explicitly accounts for the mapping order $M$, this result is to be expected. With MIESM the optimal adjustment factor errors $\epsilon\left(\beta_{\text{opt}}^{\text{MCS}}\right)$ are lower than with EESM. As a first result, we conclude that MIESM is preferable because it yields lower errors and less variance in the optimal adjustment factor $\beta_{\text{opt}}^{\text{MCS}}$.

An investigation on code rate $R$ dependence is shown in Fig. 7.8. While the information size $K = 256$ and mapping order $M = 2$ are fixed, we vary the block size $N_{\text{c}}$ and thus, the code rate $R$. We observe that the optimal adjustment factor $\beta_{\text{opt}}^{\text{MCS}}$ value increases with code rate. Again, MIESM yields lower errors and less variance. However, each code rate yields a distinct optimal adjustment factor that is required for S&RA simulations.

Next, we investigate the dependence of the optimal adjustment factor $\beta_{\text{opt}}^{\text{MCS}}$ on the effective rate $R_{\text{eff}}$ in Fig. 7.9 with $K = 256$. The results indicate that EESM yields similar $\beta_{\text{opt}}^{\text{MCS}}$ values for the same effective rate $R_{\text{eff}}$, while MIESM does not. We omit the results for further effective rates for the sake of clarity. Our findings for MIESM on effective rates are inconclusive. EESM offers the possibility to restrict the required number of adjustment factors

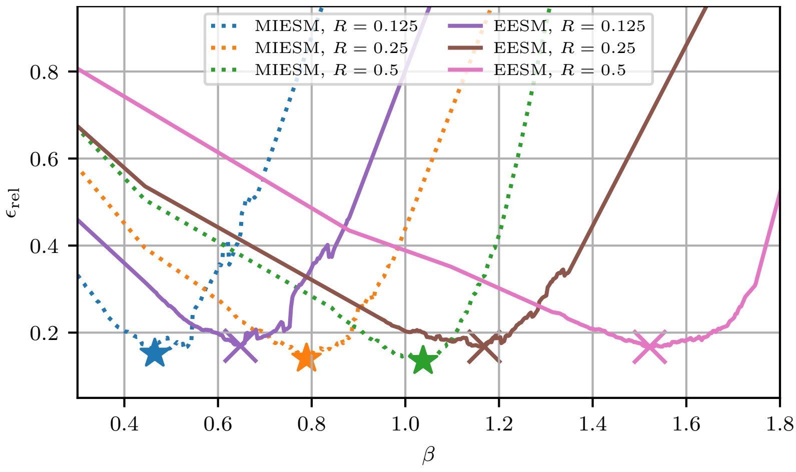


**Figure 7.8:** Relative error $\epsilon_{\mathrm{rel}}$ (...) over adjustment factor $\beta$ for varying coderates $R$.

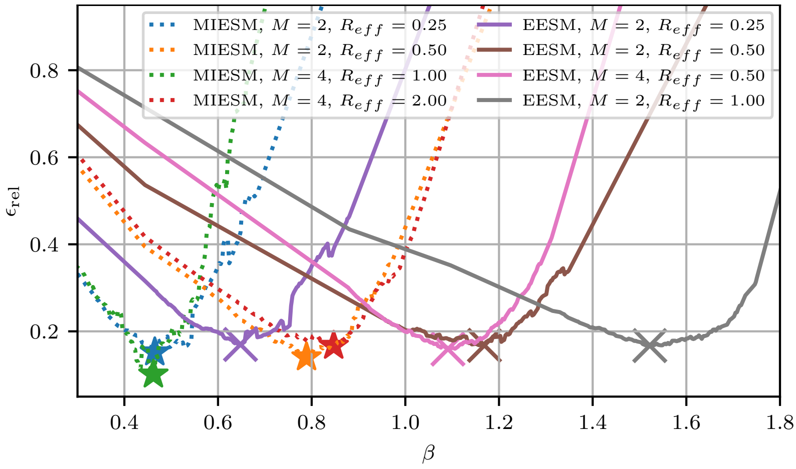


**Figure 7.9:** Relative error $\epsilon_{\mathrm{rel}}$ (...) over adjustment factors $\beta$ for varying effective rates $R_{\mathrm{eff}}$.

to one per effective rate. However, we choose one MCS per effective rate for S&RA and thus, this advantage is rather philosophical but impractical.

We want to minimize the number of adjustment factors to gain more flexibility. Correspondingly, Fig. 7.10 presents the investigation results on

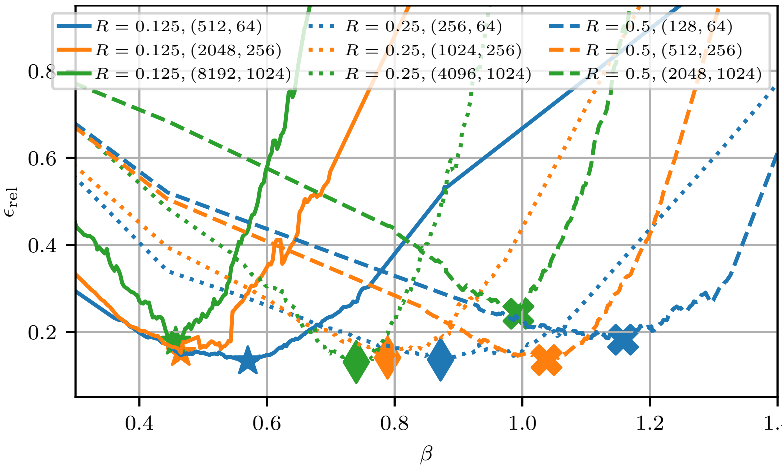


**Figure 7.10:** Relative error $\epsilon_{\text{rel}}(\ldots)$ over adjustment factor $\beta$ for varying coderates $R$ and block sizes $N_c$ with $M = 2$.

the influence of the code block length at constant code rate. We focus on MIESM results and note that the EESM simulation results are very similar and thus we omit them. Regardless of the block length, the optimal adjustment factors are grouped by code rate. While the optimal values are not exactly the same for different block lengths at the same code rate, they are sufficiently close. A larger block length $N$ reveals a tendency for lower adjustment factor $\beta$ and steeper error value ascend when moving away from the optimal adjustment factor. However, all optimal adjustment factors $\beta_{\text{opt}}^{\text{MCS}}$ for a constant code rate $R$ are sufficiently close such that choosing one adjustment factor only incurs a negligible performance degradation. Thus, it is sufficient to use one adjustment factor value per code rate. We consider it advisable to choose a representative system configuration that represents the expected mean value.

## 7.3.1 Link abstraction contribution

With our LA evaluations, we confirm that EESM and MIESM are well suited for small packets with polar codes and low error rates. Both ESM approaches yield small relative error values $\epsilon_{\text{rel}}(\ldots)$ for their respective optimal adjustment factors $\beta_{\text{opt}}^{\text{MCS}}$, i.e. both approaches closely match the FER over effective SNR curves to our AWGN target. We verified that it is sufficient to obtain one adjustment factor per code rate and mapping order, i.e. one adjustment factor per MCS. Varying number of subcarriers

**Table 7.1:** Properties of selected MCS values

| MCS index | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Coderate | 1/8 | 1/4 | 1/2 | 1/2 | 1/2 | 1/2 |
| Mapping | QPSK | QPSK | QPSK | 16QAM | 64QAM | 256QAM |
| $R_{\mathrm{eff}}$ | 0.25 | 0.50 | 1.00 | 2.00 | 3.01 | 4.00 |
| MIESM $\beta_{\mathrm{opt}}^{\mathrm{MCS}}$ | 0.465 | 0.789 | 1.038 | 0.847 | 0.752 | 0.670 |
| EESM $\beta_{\mathrm{opt}}^{\mathrm{MCS}}$ | 0.648 | 1.169 | 1.522 | 4.389 | 5.389 | 0.005 |
| SNR [dB] for $\mathrm{FER}_t = 10^{-1}$ | $-5.69$ | $-2.38$ | 1.44 | 6.79 | 11.43 | 15.53 |
| SNR [dB] for $\mathrm{FER}_t = 10^{-2}$ | $-5.08$ | $-1.78$ | 2.04 | 7.51 | 12.24 | 16.46 |
| SNR [dB] for $\mathrm{FER}_t = 10^{-3}$ | $-4.60$ | $-1.25$ | 2.53 | 8.05 | 12.91 | 17.20 |

as well as block size have a negligible impact. Based on these findings, we introduce a set of MCSs with corresponding MIESM adjustment factor values in Table 7.1 that we use for our S&RA investigations with small packets. We favor MIESM over EESM due to the slightly lower $\epsilon_{\mathrm{rel}}(\ldots)$ results and the lower variability for $\beta_{\mathrm{opt}}^{\mathrm{MCS}}$ for different configurations. The target $\mathrm{FER}_t$ is a threshold and if not met causes the system to use a more robust MCS. The reported thresholds $\mathrm{FER}_t$ in Tab. 7.1 are exemplary values for S&RA with further investigations in Fig. 7.15.

## 7.4 Scheduling

The scheduler task is to prioritize packets, i.e. determine the order, for subsequent RA [WE08]. We introduced the S&RA flowgraph in Sec. 7.1.2 and Fig. 7.2 while Fig. 7.11 illustrates the scheduling flow. All users $\mathcal{U}$ enqueue their packets for a slot and thus the scheduler operates on the set of packets $\mathcal{P}$. The considered system resources are equally split between the uplink and downlink with a switch after every slot as discussed in Sec. 7.1.1. Further, our packet arrival model in Sec. 7.1.3 assumes symmetric communication behavior between uplink and downlink. Thus, we assume that S&RA treats the uplink and downlink in the same fashion. The scheduler computes weights $w_u$ for all packets in $\mathcal{P}$ and orders them in ascending weight order $\mathcal{P}_{\mathrm{ord}}$. Subsequently, a RA sequentially allocates resources to users such that user packets with lower weight $w_u$ receive resources first. We focus on burst error minimization and thus, we consider the packet transmission success delay $N_{\mathrm{sd},u} \in \mathbb{N}^{0+}$ which quantifies a users delay in number of slots since the last successful reception. Therefore, a transmission is successful if a packet is received correctly. Moreover, we assume that the successful transmission information is available for S&RA. In other words, the success delay $N_{\mathrm{sd},u}$ quantifies the burst error length, or CEP length, per user as discussed in Sec. 7.1.2. It is important to associate the success delay with a user in contrast to other measure such as Head-of-Line Access Delay [CWC17] because packets from previous slots have reached their dead-line. A system needs to take emergency measures if a single user experiences a burst error above a threshold. In accordance with Sec. 7.1.3, we assume that every user enqueues a new packet in every slot deterministically. Besides success delay, a scheduling algorithm may use CSI as discussed in Sec. 7.2 to refine the packet order in $\mathcal{P}_{\mathrm{ord}}$. In the following we discuss our considered scheduling strategies, namely Round Robin (RR), Channel Aware (CA), Sum-Rate (SR), Delay Sensitive (DS), and Channel Aware Delay Sensitive (CADS).

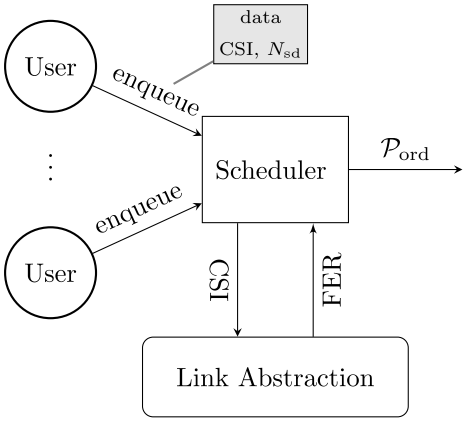


**Figure 7.11:** Scheduling flowgraph: all users enqueue a new packet in every slot deterministically. CSI and $N_{\mathrm{sd},u}$ information is available for every user.

**Round Robin (RR)** scheduling aims to distribute resources to all users equally. Thus, this strategy assumes $\mathcal{P} = \mathcal{P}_{\mathrm{ord}}$. Subsequently, the RR strategy implies that RA drops the last user packets in $\mathcal{P}_{\mathrm{ord}}$ for the current slot, in case not enough resources for all users are available.

**Channel Aware (CA)** scheduling prioritizes packets for users with currently poor channel conditions. The target is to ensure that all options to mitigate poor channel conditions during the subsequent RA are available. We assume users with good good channel conditions have a higher probability that more resources are suitable for transmission. Here, we consider AESM to be sufficient because only the packet order is important instead of absolute values. Thus, the weights $w_u = \mathrm{CNR}_{\mathrm{avg},u} = \frac{1}{N_{\mathrm{RB}}} \sum_{k=0}^{N_{\mathrm{RB}}-1} \mathrm{CNR}_k$ are averaged CNRs.

**Sum-Rate (SR)** scheduling prefers packets for users with superior channel conditions in a slot to maximize the number of transmitted packets and thus maximize sum data rate. SR serves as a reference because sum-rate maximization is a widely used target [WE08]. In contrast to CA scheduling, the weights $w_u = -\mathrm{CNR}_{\mathrm{avg},u}$ are negated and thus, the order in $\mathcal{P}_{\mathrm{ord}}$ is reversed.

**Delay Sensitive (DS)** scheduling prefers packets by users that experience a longer delay since the last successfully received packet. This strategy addresses problematic burst error conditions directly by prioritizing users that experience burst errors. Instead of CNRs, the weights $w_u = -N_{\mathrm{sd},u} \in \mathbb{N}^{0-}$ are negative success delay values per user. Since $N_{\mathrm{sd}}$ is an integer value, user packets with the same weight are ordered according to the RR strategy.

**Channel Aware Delay Sensitive (CADS)** scheduling combines the CA and the DS strategy. First, a scheduler uses the DS strategy to prioritize packets and afterwards it uses the CA strategy to determine the priority order of packets with the same user success delay $N_{\mathrm{sd},u}$.

Since the goal is to minimize burst errors, the scheduler favors DS over CA and computes the weights

$$w_u = -N_{\mathrm{sd},u} + \frac{1}{1 + \exp\left\{-\mathrm{CNR}_{\mathrm{avg},u}\right\}} \tag{7.10}$$

such that success delay $N_{\mathrm{sd},u}$ always takes precedence over CSI $\mathrm{CNR}_{\mathrm{avg},u}$. The DS scheduler produces integer weights $w_u = -N_{\mathrm{sd},u} \in \mathbb{N}^{0-}$. The CADS scheduler preserve the DS scheduler order but uses the CA scheduler to determine the order among packets with the same success delay $N_{\mathrm{sd},u}$. Here, we transform the CA scheduler weight $\mathrm{CNR}_{\mathrm{avg},u} \in \mathbb{R}^+$ into the interval $(0, 1)$ with $\frac{1}{1+\exp\{-\mathrm{CNR}_{\mathrm{avg},u}\}}$ while we preserve the CA scheduler order.

## 7.5 Resource allocation

After scheduling, we consider RA strategies. All RA strategies consume an ordered set of packets $\mathcal{P}_{\mathrm{ord}}$ and allocate these in *ascending scheduler weight order* as illustrated in Fig. 7.12. These packets receive allocations from the set of resources $\mathcal{R}$ a system offers in a slot as discussed in Sec. 7.1.1. Then, RA yields a Modulation and Coding Scheme $\mathrm{MCS}_u$ and allocated resources $\mathcal{R}_u$ per user. The set of user allocated resources $\mathcal{R}_u \subseteq \mathcal{R}$ is the subset of resources that is considered for one user and finally allocated to a specific user. The allocated packets are multiplexed onto their designated resources in a slot before transmission.

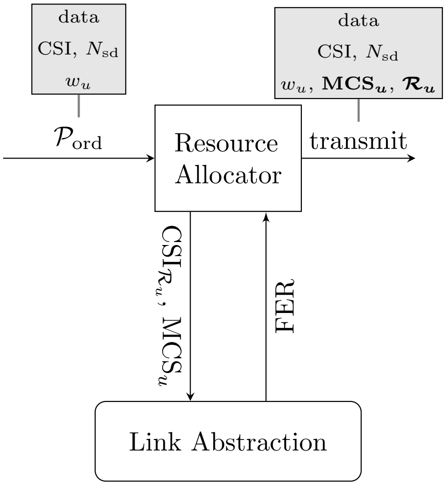

data
CSI, $N_{\mathrm{sd}}$
$w_u$
data
CSI, $N_{\mathrm{sd}}$
$w_u$, $\boldsymbol{\mathrm{MCS}_u}$, $\boldsymbol{\mathcal{R}_u}$
$\mathcal{P}_{\mathrm{ord}}$
Resource Allocator
transmit
$\mathrm{CSI}_{\mathcal{R}_u}$, $\mathrm{MCS}_u$
FER
Link Abstraction

**Figure 7.12:** Resource Allocation (RA) flowgraph

The allocator may use CSI to determine an expected FER for a specific MCS. While a more robust MCS, i.e. a MCS with a lower index, is desirable from a reliability point of view, we need to consider multiple users which compete for shared resources. A more efficient effective rate $R_{\mathrm{eff}}$, and thus higher MCS index, for links with high SNR allows to spend more resources on links with low SNR and in turn improve system robustness against burst errors.

The general RA approach to select an MCS starts with the highest MCS. The allocator selects the required resources for the current MCS and queries link abstraction for the resulting FER. Here, the allocator selects the most suitable resources over the whole bandwidth. By setting $\mathrm{FER}_t$, we define an effective SNR threshold for the every MCS. AWGN simulations provide

FER over SNR curves that we use to obtain switch points between MCSs. We require accurate LA to conduct precise simulations where a FER is an accurate measure to simulate packet loss. If the obtained FER is smaller than the threshold $\mathrm{FER}_t$, the current user packet is assigned the selected MCS along with the allocated resources $\mathcal{R}_u$. In case $\mathrm{FER} > \mathrm{FER}_t$, the error probability is above our target $\mathrm{FER}_t$ threshold and a lower MCS is selected that allows for more robust transmission. This process is repeated until an MCS that satisfies the threshold is found. The RA algorithm continues with the next user until all user packets are allocated or until resources are exhausted. Unallocated packets are dropped and the user success delay count $N_{\mathrm{sd},u}$ is increased by one for the user corresponding to a dropped packet.

Throughout our work, we mostly consider $E_b/N_0$ for our SNR definition. However, for our system level investigations, the $E_d/N_0$ SNR is more suitable. Thus, we use this definition here.

We require some form of multiplexing to combine packets from different users and convey them over the shared resource grid in a slot as discussed in Sec. 7.1.1 [Fre14]. The discussed S&RA strategies determine how multiplexing is facilitated on a slot by slot basis. Generally, two options to allocate resources exist, namely Time-Division-Multiplex (TDM) and Frequency-Division-Multiplex (FDM) [DPS18]. With TDM, we allocate the required amount of resources for a packet equally distributed over the whole bandwidth. Thus, the TDM allocator selects the same number of elements from each block, e.g. three elements from each of the $N_{\mathrm{RB}}$ resource blocks for a total of $3N_{\mathrm{RB}}$ elements. This approach offers some advantages, e.g., frequency diversity, and resource allocation. We only need to determine the MCS and allocate an equal amount of resources from every resource block. In case of stale CSI, we may still leverage diversity to reliably receive a packet. However, we might need to compensate for bad channel conditions in some resource blocks by spending more resources over all.

With FDM, the allocator assigns the resource blocks with the best CSI for a user to that user packet to leverage multi-user diversity. In this case, we need to select the MCS and allocate specific resources $\mathcal{R}_u$ that depend on each other and make this process more involved. However, we avoid the need to compensate for bad channel conditions within a sub-band of our overall bandwidth. In contrast to TDM, we are more prone to imperfect CSI because the selected resources might in fact be in a state of deep fade at time of transmission. In this case, the chosen MCS in conjunction with the selected resources might not be able to cater a successful transmission. Full inclusion of FDM with all RA strategies is an extension to our prior work [DBD20].

With these considerations in mind, we discuss our considered RA strategies.

**Static** RA does not consider any dynamic information but uses a static $\mathrm{MCS}_u$. This approach is simple because it discards the need to dynamically select a MCS. However, it wastes resources on users with superior channel conditions while it might be able to cater more users otherwise.

With $K = 256$ and $\mathrm{MCS}_u = 0$, i.e. $R = \frac{1}{8}$ and QPSK, we need $N_\mathrm{d} = 1024$ complex symbols to convey one user packet. With our further slot assumptions and $B_\mathrm{s} = 100\,\mathrm{MHz}$, we expect a maximum of 7290 information elements per slot. We are only able to serve 7 users in that system. In accordance with [ADE+19], we may assume QPSK and $R = 0.5$ which corresponds to $\mathrm{MCS}_u = 2$ in Tab. 7.1. While there are enough resources to serve $N_U = 28$ users in this case, users with poor channel conditions may not be allocated with sufficient resources at all.

**Dynamic** RA uses LA to compute an expected FER. Starting with the highest MCS index in Tab. 7.1 its corresponding expected FER is calculated. The highest MCS index corresponds to the configuration with the highest effective rate $R_\mathrm{eff}$. If the expected FER is above the threshold $\mathrm{FER}_t$, the next lower MCS index is selected and the process starts over. Thus, the $\mathrm{MCS}_u = \mathrm{MCS}_{\mathrm{dyn},u}$ for the current user packet is determined as the highest MCS index that satisfies the threshold $\mathrm{FER}_t$. In case of TDM, the required resources for this MCS are now allocated equally from every resource block. In case of FDM, the selected resources $\mathcal{R}_u$ correspond to resource blocks that are now allocated to that user packet. This strategy enables efficient use of available resources and supports more users on average than the static strategy. In case we are unable to allocate sufficient resources for a user packet that packet is dropped. Finally, we assume that re-transmission or incremental redundancy schemes are unavailable because the additional latency would exceed the packet real-time deadline.

**Backoff** RA is an extension to dynamic RA. First, it determines $\mathrm{MCS}_u$ according to the dynamic RA strategy. However, the backoff strategy takes success delay into account. The finally selected MCS is computed with $\mathrm{MCS}_u = \mathrm{MCS}_{\mathrm{dyn},u} - N_{\mathrm{sd},u}$. Thus, the allocator grants extra resources to user packets with a corresponding user in a burst error state to minimize the burst error probability. With this strategy we are able to efficiently use available resources but focus on burst errors specifically.

**Failsafe** RA is a more aggressive extension to backoff RA. Again, the allocator use the dynamic RA but then calculates

$$\mathrm{MCS}_u = \begin{cases} \mathrm{MCS}_{\mathrm{dyn},u} & \text{if } N_{\mathrm{sd},u} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.11}$$

such that it switches into a failsafe mode for a user that experiences a burst error. The $\mathrm{MCS}_u = 0$ implies that the RA strategy selects the MCS with the best error protection but lowest effective rate $R_{\mathrm{eff}}$. Accordingly, $\mathrm{MCS}_u = 0$ implies that it requires more resources than any higher $\mathrm{MCS}_u > 0$. Still, the failsafe strategy uses resources efficiently while errors are weighted even heavier. However, this aggressive strategy might quickly exhaust resources and packets might be dropped despite excellent channel conditions.

## 7.6 System level simulations

In this section we investigate the impact of different parameters on burst errors. The results we present here are partially published in our prior work [DBD20]. Here, we re-evaluate this work and extend it in several areas. We investigate application level availability with success delay probability

$$P\left(N_{\mathrm{sd}}\right) = P\left(\text{successful reception after max. } N_{\mathrm{sd}} \text{ packets}\right)$$

curves for packet transmission success delay $N_{\mathrm{sd}}$ as discussed in Sec. 7.1.2. The success delay probability $P\left(N_{\mathrm{sd}}\right)$ is computed as follows. First, we sum up all transmitted packets by all users in a simulation run. Second, we compute how often individual $N_{\mathrm{sd}}$ values appear. Finally, the $P\left(N_{\mathrm{sd}}\right)$ values are computed, e.g. for the $P\left(N_{\mathrm{sd}} \leq 2\right)$ value for $N_{\mathrm{sd}} = 2$, we accumulate the number of occurrences for $N_{\mathrm{sd}} = 0, 1, 2$ and determine their percentage among all transmitted packets. Thus, a $P\left(N_{\mathrm{sd}}\right)$ is a cumulative distribution function that indicates the delay until the next successful packet transmission for all users in the system.

### 7.6.1 Assumptions

We assume an industrial radio setup at 3.8 GHz and 100 MHz bandwidth, 60 kHz subcarrier spacing, and thus $N_{\mathrm{RB}} = 135$ with the structure discussed in Sec. 7.1.1 which corresponds to 5G NR band *n78* [DPS18, SKMB21, ETS21]. While we consider the downlink, we assume a TDD configuration with balanced uplink and downlink resources. Thus, each resource blocks consists of $N_{\mathrm{d}} = 54$ data symbols that are assigned to one user in case of FDM. In case of TDM, a user receives the same number of resources from each resource block, e.g. 2 symbols from every resource block as discussed in Sec. 7.1.1. The required resources for a user are determined based on the MCSs presented in Tab. 7.1. We simulate $2 \cdot 10^6$ slots where every user transmits one packet with 32 byte or 256 bit of information per slot.

All devices are equipped with a single antenna, have 24 dBm maximum transmit power, and $F = 9\,\mathrm{dB}$ receiver noise figure [ETS18b]. The noise floor calculation assumes room temperature at $T = 300\,\mathrm{K}$.

We always assume Non-Line-Of-Sight (NLOS) conditions. The large scale parameters are $\sigma_{\mathrm{SF}} = 8\,\mathrm{dB}$ shadowing deviation, $\rho_{\mathrm{s}} = 5\,\mathrm{m}$ correlation distance and a path loss exponent $\eta = 3$ [ETS18b]. We assume that all devices move at $v = 15\,\mathrm{m\,s^{-1}}$ velocity which determines the channel coherence time. For small scale parameters, we assume $\sigma_{\mathrm{RMS}} = 46.8\,\mathrm{ns}$ and $\tau_{\max} = 250\,\mathrm{ns}$ with an exponential power delay profile as discussed in Sec. 2.2 [DHC+19].

### 7.6.2 Simulation results

Communication system availability is defined as the success delay probability $P(N_{\mathrm{sd}})$ for burst errors $N_{\mathrm{sd}} \leq 2$ that are marked with crosses in all result figures. We require availability above 99.999 %, indicated with a dashed horizontal line.

First, we discuss the results in Fig. 7.13 where we compare scheduler strategies with TDM and *DYNAMIC* RA for $N_U = 42$ users at 20 m distance. First, the DS scheduler delivers best performance. Second, the CADS scheduler is on par with the DS scheduler. Thus, we conclude that a scheduler for a TDM system must focus on delay sensitivity. The CA and RR schedulers both exhibit the poorest performance while the SR scheduler represents a middle ground.

We continue this discussion with Fig. 7.14 with FDM and the distance is increased to $d_{\mathrm{g}} = 30\,\mathrm{m}$. The SR strategy exhibits the poorest performance which highlights our proposed focus shift to burst error minimization. The RR strategy delivers better performance than the SR strategy but falls short to the other strategies.

The DS, CA, and CADS strategies exhibit the best performance. These scheduler strategies are almost able to provide the required 99.999 % burst error resilience at 30 m in FDM mode. In conjunction with our scheduler results gathered in our TDM simulation, we conclude that the CADS should be preferred to meet burst error resilience requirements present in URLLC systems.

Fig. 7.15 presents the results of a TDM system with a CADS scheduler and a *DYNAMIC* RA strategy with $N_U = 42$ users at 20 m distance for different target FERs $\mathrm{FER}_t$. The burst error resilience increases with decreasing

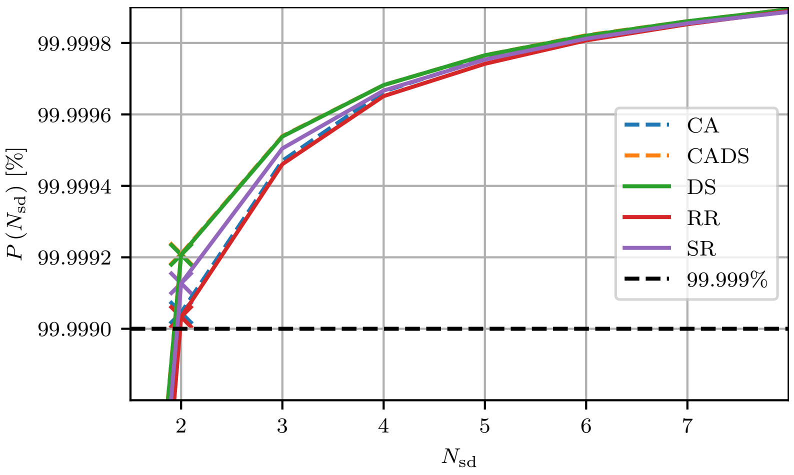


**Figure 7.13:** Scheduler strategies with TDM, *DYNAMIC* $N_U = 42$, $d_{\mathrm{g}} = 20\,\mathrm{m}$

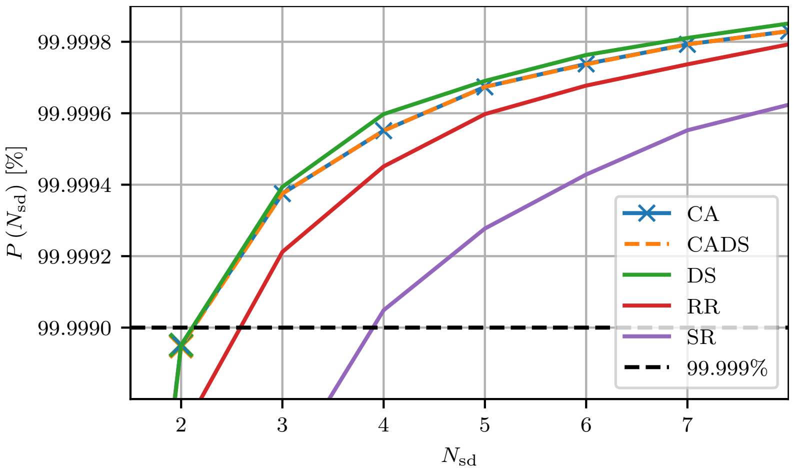


**Figure 7.14:** Scheduler strategies with FDM, *DYNAMIC*, $N_U = 42$, $d_{\mathrm{g}} = 30\,\mathrm{m}$

$\mathrm{FER}_t$ values. However, we conclude that it is sufficient to use $\mathrm{FER}_t = 0.1$ because lower values have a negligible impact on burst error performance. Especially $\mathrm{FER}_t$ values below $10^{-2}$ do not boost burst error resilience while resource usage may increase.

Next, Fig. 7.16 shows a comparison of the *DYNAMIC*, *BACKOFF*, and *STATIC* RA strategies for TDM and FDM with $N_U = 16$ users at $20\,\mathrm{m}$

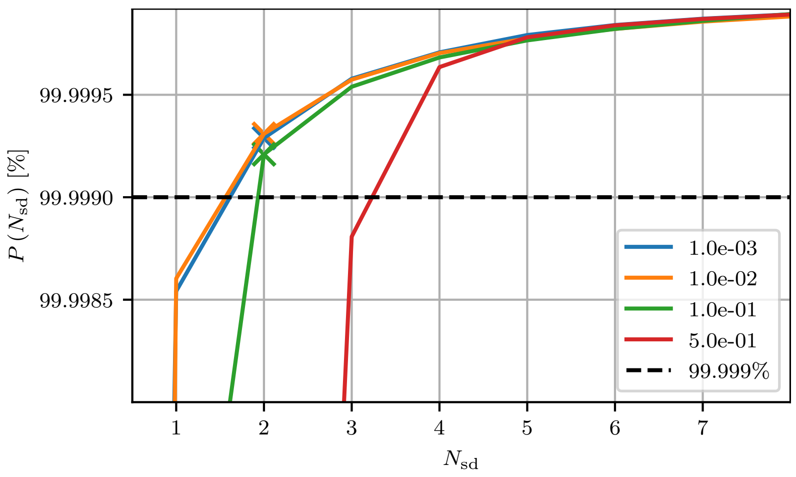


**Figure 7.15:** Target FERs with TDM, CADS, *DYNAMIC*, $N_U = 42$, $d_g = 20\,\text{m}$

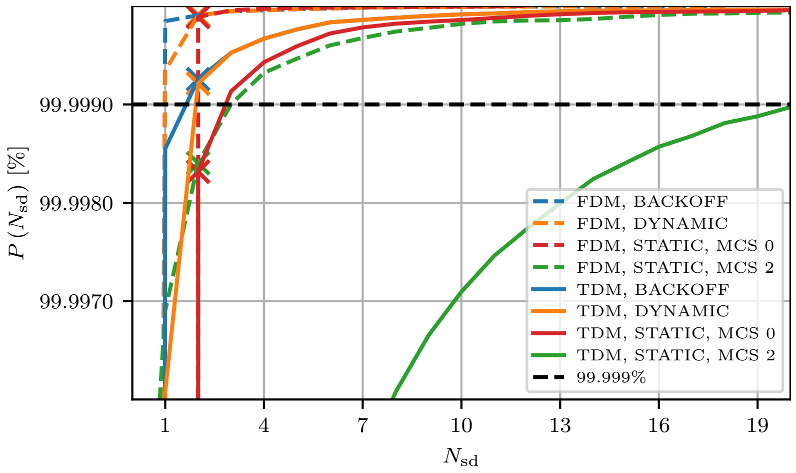


**Figure 7.16:** RA strategies with CADS, $N_U = 16$, $d_g = 20\,\text{m}$

distance and perfect CSI. The results for the *FAILSAFE* strategy are omitted because they duplicate the *BACKOFF* results in this case. With perfect CSI knowledge the FDM mode exhibits superior performance for all RA strategies. With MCS = 2 in accordance with [ADE+19], we observe that the burst error probability increases significantly. In case of FDM a static MCS with MCS = 2 yields better results due to an additional degree of freedom, namely allocated resource blocks. Even with $\mathrm{MCS}_u = 0$ and FDM, the *STATIC* RA strategy still outperforms any dynamic TDM strategy at success delay $N_{\mathrm{sd}} = 1$. The *STATIC* RA strategy in conjunction with TDM is unable to meet the 99.999 % target. With a static $\mathrm{MCS}_u = 0$ the system is already overloaded with $N_U = 16$ users and the allocator drops some packets in every slot due to a lack of resources. In summary, we point out that dynamic RA strategies are a strict requirement for realistic numbers of users. Strategies that incorporate burst error state knowledge improve performance further.

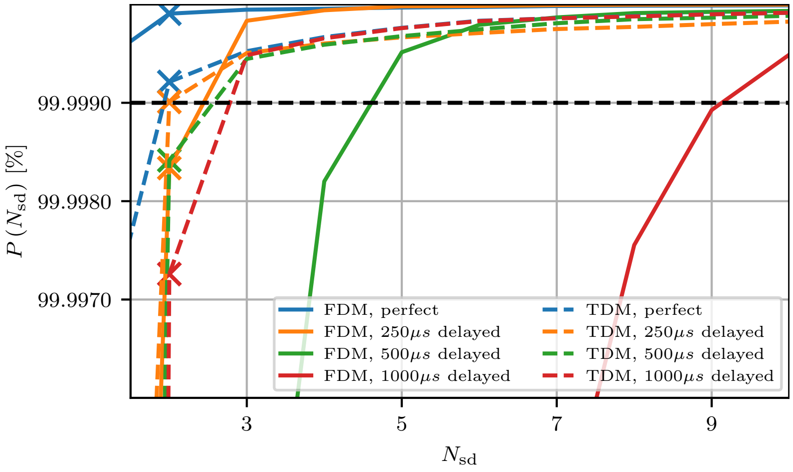


**Figure 7.17:** Delayed CSI with CADS, *DYNAMIC*, $N_U = 16$, $d_g = 20\,\mathrm{m}$

In contrast to Fig. 7.16, the picture changes quite significantly with imperfect CSI in Fig. 7.17. While FDM based *DYNAMIC* RA performs best with perfect CSI, a 250 µs delay causes a significant performance loss. In case of TDM, the performance loss is way lower. At 250 µs a TDM based RA is able to deliver $N_{\mathrm{sd}} = 2$ burst error resilience just above 99.999 %. Even at 1 ms CSI delay, a TDM based RA is able to cope with $N_{\mathrm{sd}} = 3$ burst errors. The FDM performance degrades dramatically with increasing CSI delay.

We compare allocator strategies with imperfect CSI for FDM in Fig. 7.18 and for TDM in Fig. 7.19. As expected, the *BACKOFF* RA strategy yields the best results with perfect CSI and serves as a reference. Again, we observe that the *DYNAMIC* RA strategy degrades. However, the $N_{\mathrm{sd}}$ aware RA strategies exhibit different behavior for both multiplexing modes. In case of FDM, the *FAILSAFE* strategy yields the best results with imperfect CSI but the *BACKOFF* strategy is unable to yield similar performance. In

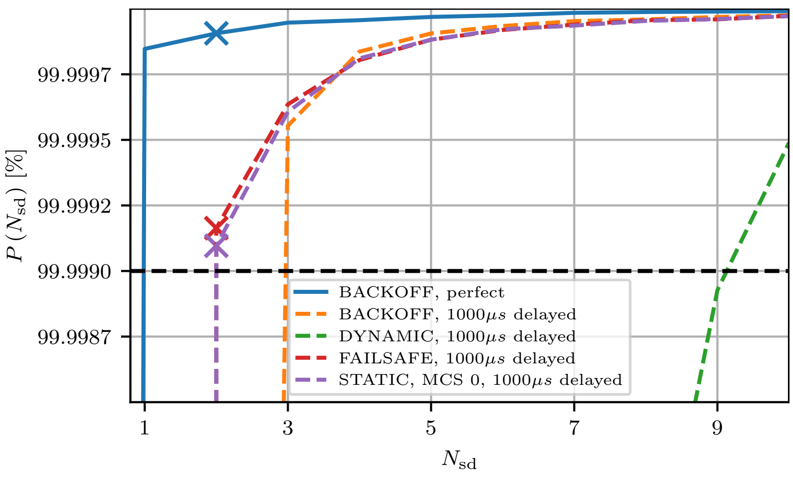


**Figure 7.18:** RA strategies for FDM with CADS, $N_U = 16$, $d_{\text{g}} = 20\,\text{m}$

contrast in TDM mode, the *BACKOFF* RA strategy delivers almost the same performance as the *FAILSAFE* strategy. Thus, we summarize that $N_{\text{sd}}$ aware allocator strategies are strictly necessary for systems with strict burst error requirements.

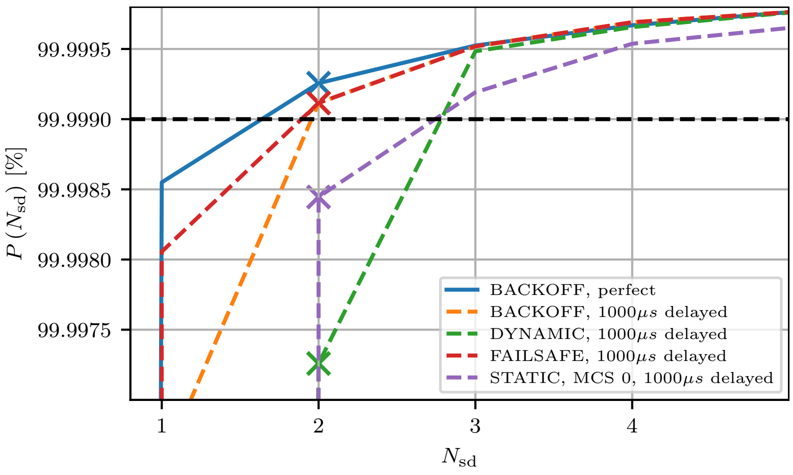


**Figure 7.19:** RA strategies for TDM with CADS, $N_U = 16$, $d_{\text{g}} = 20\,\text{m}$

In Fig. 7.20 $N_U = 16$ users move at different constant distances to an AP.

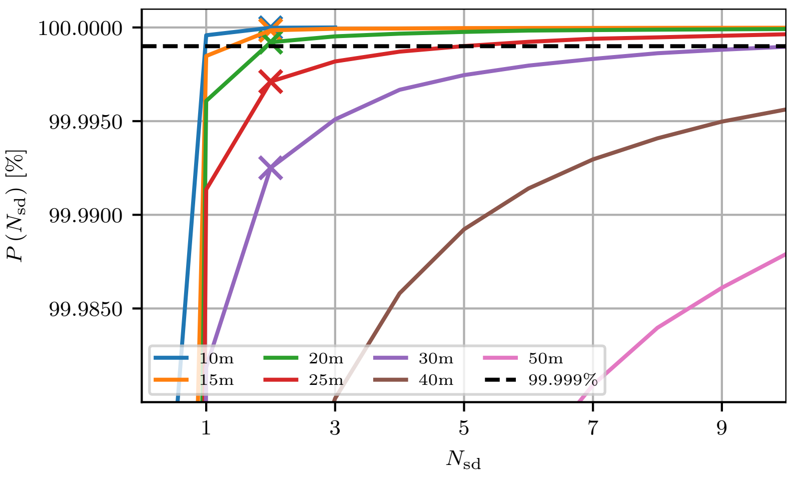


**Figure 7.20:** Varying distance to $d_g$ a AP for TDM with CADS, *DYNAMIC* RA, $N_U = 16$

Here we consider TDM with CADS and a *DYNAMIC* RA strategy. Even at a distance $d_g = 25\,\text{m}$ the communication system fails to provide the required $N_{sd} = 2$ burst error resilience requirement. In order to cover larger areas other measures are required, e.g. multiple APs as discussed in Chapter 6 and suggested in [ETS20a].

After we established that the 99.999 % availability requirement is only met for distances up to 20 m from the AP in Fig. 7.20, we investigate burst errors in greater detail in Fig. 7.21. For both, TDM and FDM, the mean $\text{SNR}_{\text{eff}}$ for burst errors with $N_{sd} \geq 2$ is below $-5.69\,\text{dB}$. For TDM with the assumed $\text{FER}_t = 0.1$, the maximum $\text{SNR}_{\text{eff}}$ for $N_{sd} \leq 3$ is above $-5.69\,\text{dB}$, while for FDM it is below. However, imperfect CSI would reverse these results as shown in Fig. 7.18. Thus, we conclude that burst errors are caused by channel outages due to fading and shadowing and a single AP system is incapable of mitigating these. Deployments with multiple APs are supposedly a suitable counter-measure.

The results for varying system bandwidths $B_s$ are presented in Fig. 7.22 for $N_U = 9$ at 30 m distance with TDM, CADS, and *DYNAMIC* RA. We chose $N_U = 9$ users to ensure that a system with $B_s = 20\,\text{MHz}$ bandwidth is not overloaded. First, a higher bandwidth results in more burst errors because a higher bandwidth implies more noise power with the same 24 dBm transmit power. With 20 MHz bandwidth we observe a higher burst error probability for $N_{sd} = 1$ then for 50 MHz bandwidth. The 20 MHz system is

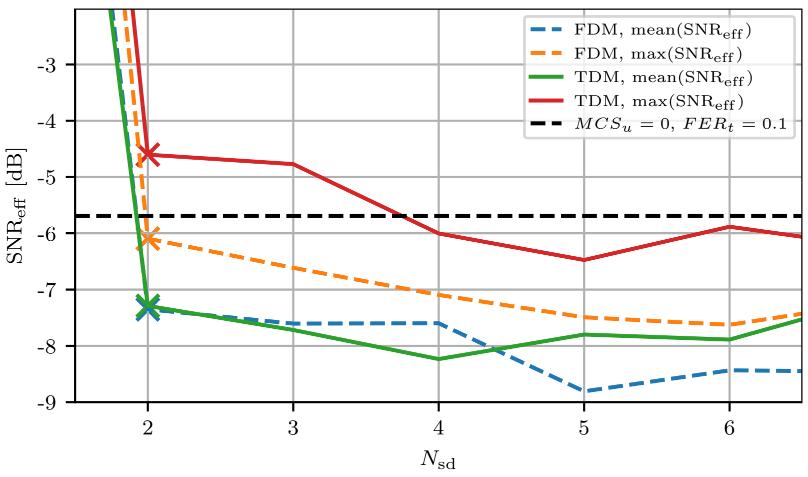


**Figure 7.21:** Burst error analysis with CADS, *FAILSAFE* RA, $N_U = 16$, $d_\mathrm{g} = 20\,\mathrm{m}$, perfect CSI

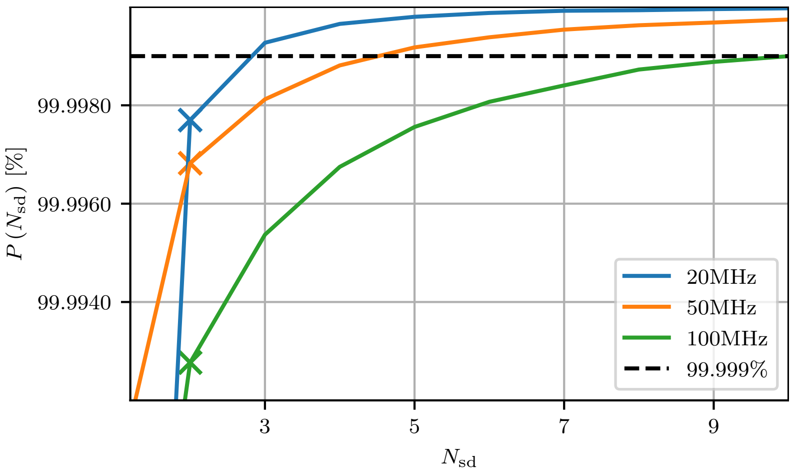


**Figure 7.22:** Varying system bandwidth $B_\mathrm{s}$ for TDM with CADS, *DYNAMIC* RA, $N_U = 9$, $d_\mathrm{g} = 30\,\mathrm{m}$

already fully loaded and thus prone to dropped packets because it exhausted its resources for the current slot. A reduced bandwidth may only be an option in scenarios with only a few users.

Similar to varying amount of CSI delay, we investigate the impact of user

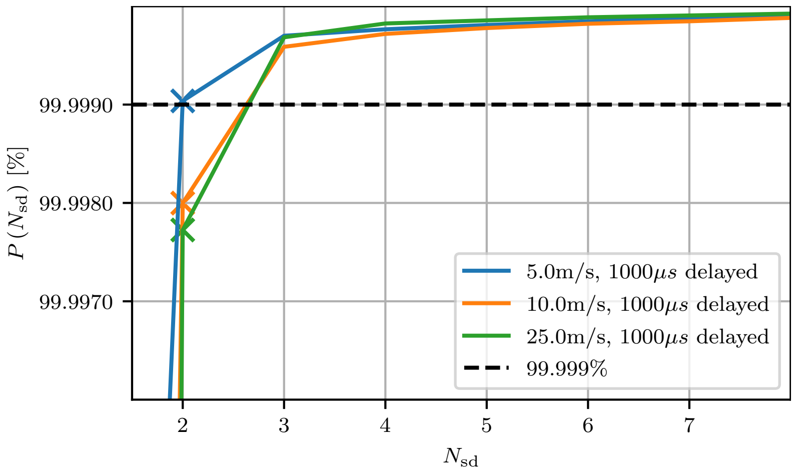


**Figure 7.23:** Varying $v$ user velocity for TDM with CADS, *DYNAMIC* RA, $N_U = 16$, $d_g = 20\,\mathrm{m}$

velocity with constant 1 ms CSI delay in Fig. 7.23. We observe that higher velocity results in more burst errors because velocity translates into more rapidly changing channels and thus, CSI may become stale quicker.

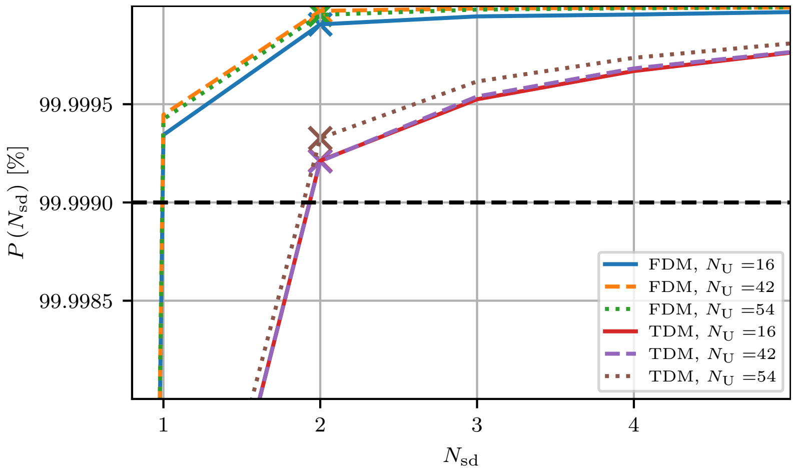


**Figure 7.24:** Varying $N_U$ users for TDM with CADS, *DYNAMIC* RA, $d_g = 20\,\mathrm{m}$

We analyze the influence of the number of users $N_U$ in a system that receive resources in every slot on system performance in Fig. 7.24. We would expect higher resilience to burst errors for systems with fewer users $N_U$. However, the contrary seems to be correct. We observe slightly better results for systems with more users. Though, the differences are very small.

## 7.7 Summary

In this chapter we discussed a MAC layer for a URLLC system with periodic deterministic communication behavior with real-time constraints. First, we evaluated the LA algorithms AESM, EESM, and MIESM. This includes investigations into suitable error measures, and into how multiple parameters impact the optimal adjustment factor that tweaks the LA performance. Based on these findings, we derived a suitable set of MCSs that we use in our S&RA investigations to combat burst errors. Our S&RA investigations commenced with a thorough discussion of our considered system, including the signal processing flow, multiplexing options, and an introduction to our considered strategies. Finally, we conducted a series of experiments to provide an in-depth analysis of burst error dependence. This analysis carved out that dynamic S&RA strategies are strictly necessary and that they benefit greatly from error state knowledge to quickly mitigate and thus minimize burst errors.

### 7.7.1 Contribution

First, we contributed a re-evaluation of the EESM and MIESM results for short packet LA and extend upon them [DBD20]. The addition of AESM constitutes one extension that we contribute in this chapter. This evaluation leads to the conclusion that accurate LA for short packets and URLLC requirements is possible. Further, MIESM with a relative error measure yields the best performance. Additionally, simulation results show how susceptible adjustment factors are to different system parameters. Based on these findings, we derived a suitable set of MCSs and corresponding optimal adjustment factors that we use in our S&RA investigations to combat burst errors. This enables us to study S&RA in frequency selective block fading channels with polar codes and multicarrier modulation [Ars15].

Next, we re-evaluated our prior work [DBD20] where we investigated how SotA Scheduling and Resource Allocation (S&RA) strategies perform with respect to burst errors in scenarios with short packets, low latency and low FER requirements. Here, we contribute an extension to these investigations to meet URLLC requirements, e.g. fixed MCS setups [ADE$^{+}$19]. Further, the fully integrated TDM and FDM multiplexing approaches constitute an additional contribution in this work. This leads to the conclusion that while FDM is able to provide better burst error resilience with perfect CSI knowledge, TDM is favorable in case CSI is imperfect, even if only slightly. Additionally, we suggest new delay-sensitive approaches to S&RA that improve burst error resilience and conclude that it is important to shift the focus from sum-rate maximization to burst error minimization. On top of that, one single antenna AP is only capable of providing a 99.999 % availability URLLC QoS for users with up to 20 m distance, due to fading and shadowing, while larger coverage areas require multiple APs. We performed an in-depth analysis of burst error dependence and carved out that dynamic S&RA strategies are strictly necessary unless surplus resources are available. Specifically, any dynamic RA outperforms a static RA by a large margin under resource constraints. Dynamic S&RA strategies benefit greatly from error state knowledge to quickly mitigate and thus minimize burst errors. Finally, these investigations demonstrate that delay-sensitive scheduling as well as RA yield superior results in terms of burst error performance.

# Chapter 8

# Testbed implementation

The technologies we discussed in previous chapters are implemented in an Over-the-Air (OTA) testbed that we discuss in-depth in this chapter. The implementation is the culminated work gathered through various projects, including Innovative Wireless Technologies for Industrial Automation (HiFlecs), TACtile interNET 4.0 (TACNET 4.0), and Industrial Radio Lab Germany (IRLG). We start with a discussion of our testbed concept, the software environment, and the used and implemented software components and their usage. Afterwards, we discuss the available hardware. Finally, we present two testbeds and conducted measurements that demonstrate the capabilities of our implementation. The testbed at the NEOS building, Bremen, in Sec. 8.7.4 employs our full implementation while the older testbed at Bosch Hildesheim in Sec. 8.7.3 was deployed with alterations that are discussed in the corresponding section. The capabilities of interest are the achievable latencies under different parameterizations and reliability investigations for our Cloud RAN setup. The Cloud RAN setup includes distributed antennas and APs to improve reliability by providing spatial diversity, and thus redundancy.

We contribute an open-source software OTA communication system implementation built upon GNU Radio with multiple Out-Of-Tree (OOT) modules [Dem22a, DRKK22, DG22, Dem22b, Dem22c, DL22]. The USRP Hardware Driver (UHD) provides the connection between GNU Radio and our implementation and the connected USRPs to leverage their capabilities. These capabilities include, continuous high rate sample streams, exact receive and transmit timing information, and high precision synchronization across multiple devices. With distributed APs to improve reliability through spatial diversity, the chosen approach shows a significant improvement to counter fading induced communication outages. Thus, distributed APs provide an important option to minimize burst errors in URLLC communication systems. Our latency measurements confirm that low latency communication systems are achievable and fully implementable in software. Moreover, the accompanying reliability measures demonstrate that our system is able to provide ultra reliable communication. Especially, our measurements at the NEOS building add valuable insight into reliability and latency properties while the measurements at Bosch Hildesheim comprise a good first understanding of the system properties. The GNU Radio OTA software implementation is able to provide lower latency than current Long Term Evolution (LTE) and 5G NR systems [SFVS20]. We show that DSP algorithms alone are only a partial contributor to system latency. The whole system needs to be tightly integrated and optimized in a future work in order to deliver the highest possible performance and consequently lowest latency. Thus, we contribute detailed insight into a full software implementation of a communication system along with valuable pointers what to optimize further.

## 8.1 Testbed concept

The testbed includes numerous parts in software and hardware. Here we present an overview as illustrated in Fig. 8.1 and references to more details in their corresponding sections. All aspects of our testbed are designed with this concept in mind.

We consider a factory hall where multiple AGVs operate as introduced in Fig. 2.1. In this testbed we use Götting KG Kinetic Automat for Transport Enhancements (Götting KATEs) as discussed in Sec. 8.6. All Götting KATEs are controlled by a central AGV control server that requires periodic, deterministic communication with small packets. By default this application uses IEEE 802.11 (Wi-Fi) to communicate, however, Wi-Fi only allows for a 100 ms reliable communication periodicity. We replace this system with our custom SDR solution to accommodate for these high reliability and low latency requirements with short packets. Thus, the application may decrease the communication periodicity to 20 ms which is the lowest periodicity supported by the application.

Our communication system is comprised of modules on the AGVs and a RAN as described in Sec. 8.6. Most importantly, we consider USRPs as RAPs to realize a distributed AP setup to improve spatial diversity in the uplink and downlink. All USRPs are synchronized and connected to our Cloud RAN system where all DSP is performed. Thus, all DSP is either computed on a Central Processing Unit (CPU) on the cloud platform or on a CPU in a computer on an AGV.

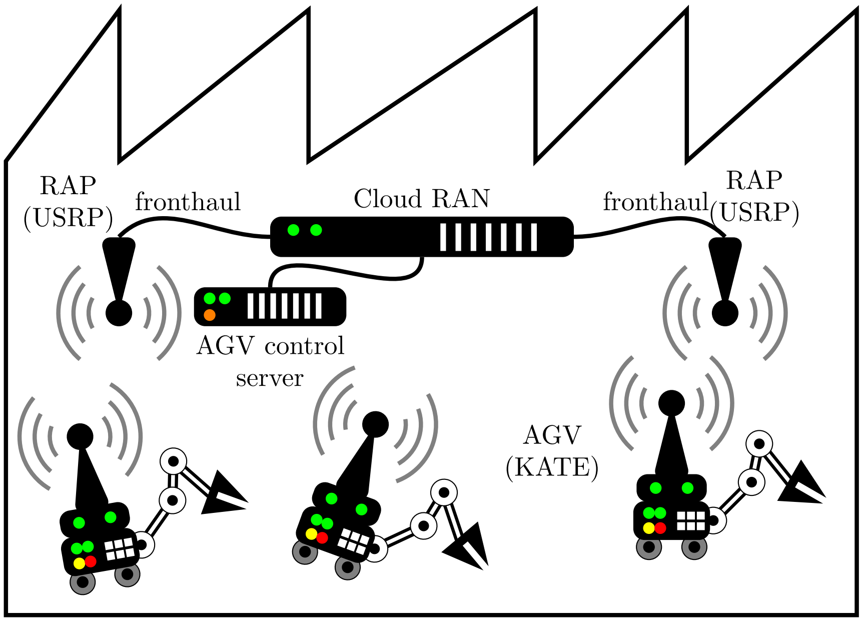


**Figure 8.1:** Demonstrator concept with its core physical components. All DSP is implemented in software and executed on the Cloud RAN platform or on a computer on an AGV.

We use a GNU Radio based software environment to implement all the necessary DSP and support functionality as discussed in Sec. 8.4. The polar code and GFDM based implementation consists of various components that we discuss in detail in Sec. 8.5. Since the software implementation is very flexible, we start with a discussion on a suitable configuration for the application at hand in Sec. 8.2. It is important to note that a simulation environment and a live communication system exhibit significant structural differences. In Sec. 8.3 we discuss the design of our implementation and how components are usable in both simulations and our OTA live streaming implementation while maintaining the flexibility to employ different system designs.

## 8.2 Testbed configuration

The presented testbed implementation is very flexible. However, the number of parameters and their inter-dependencies are numerous. Thus, we present a default configuration that is assumed throughout this chapter for the uplink as well as the downlink.

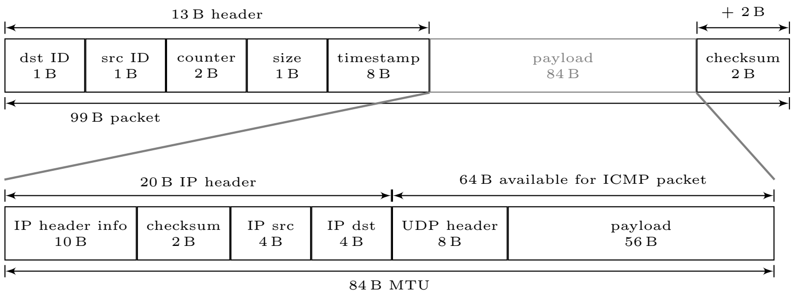
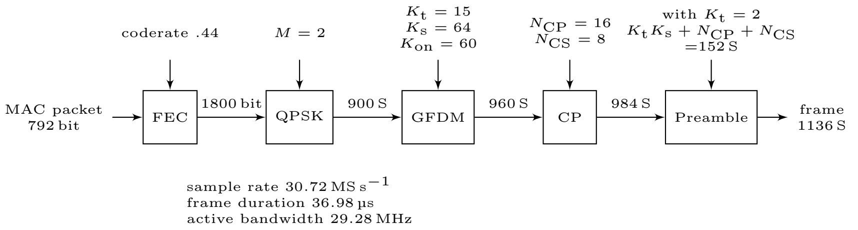


**Figure 8.2:** Uplink and downlink frame structure with 99 B frames that consist of our custom 15 B MAC header and an encapsulated 84 B IP MTU structure with a minimal 20 B IPv4 header, fixed 8 B UDP header, and up to 56 B payload.

We configure our GNU Radio flowgraph such that it accepts packets with a Maximum Transmission Unit (MTU) of 84 B. In Sec. 8.4.1 we discuss how the Linux network stack ensures seamless integration including packet segmentation and re-assembly. The frame and packet structure is shown in Fig. 8.2. We assume a 99 B, or 792 bit, fixed size OTA packet with our custom 15 B MAC header and 84 B payload that corresponds to a 84 B Maximum Transmission Unit (MTU). Shorter payloads are zero-padded to fill a fixed size MAC packet in order to keep system complexity at bay. The 15 B MAC header consists of a 1 B destination IDentifier (ID), 1 B source ID, 2 B frame counter, and 1 B payload size indicator. Additionally, we add an 8 B high precision timestamp. Finally, a 16 bit, or 2 B CRC checksum is appended to every packet that is re-used for polar code list decoding.

As illustrated in Fig. 8.2, the MAC payload may carry Internet Protocol (IP) packets with a 84 B MTU [Pos81a]. The 84 B MTU includes a 20 B IP header without any optional fields [Pos81a]. Thus, the effective maximum packet payload is 64 B which is also the minimum packet size to ensure that `ping` Internet Control Message Protocol (ICMP) packets are not fragmented [Pos81b]. A User Datagram Protocol (UDP) packet with a fixed 8 B UDP header may carry up to a 56 B payload without fragmentation [Pos80].

Further, we want to make sure that we always operate on 1 B multiples because smaller units incur management overhead without benefit, e.g. larger packet size header fields for bits instead of bytes. Modern CPUs operate on bytes and thus it is beneficial to keep all configuration aligned to the 1 B multiple principle to avoid additional complexity. Thus, data structures are aligned to byte-multiples, e.g. C/C++ `int8_t`, or 8bit integer (int8), and `float`, or fp-32. Software applications allow for more rapid development and CPUs are optimized to execute on byte multiples and thus offset any custom data structures.



coderate .44
$M = 2$
$K_\text{t} = 15$ $K_\text{s} = 64$ $K_\text{on} = 60$
$N_\text{CP} = 16$ $N_\text{CS} = 8$
with $K_\text{t} = 2$ $K_\text{t}K_\text{s} + N_\text{CP} + N_\text{CS}$ =152 S
MAC packet 792 bit
FEC
1800 bit
QPSK
900 S
GFDM
960 S
CP
984 S
Preamble
frame 1136 S
sample rate 30.72 MS s$^{-1}$
frame duration 36.98 µs
active bandwidth 29.28 MHz

**Figure 8.3:** PHY configuration for fixed size 99 B, or 792 bit, MAC packets.

The PHY configuration is presented in Fig. 8.3 where a 99 B MAC packet is transformed into a 1136 S frame. In order to align the configuration with 5G NR systems, we assume a 30.72 MS s$^{-1}$ sample rate that saturates the Universal Serial Bus (USB) transport link of an Ettus USRP B210 (B210) in a $N_\text{ant} = 2$ antenna configuration[1]. With these constraints in mind, we configure the FEC encoder with a $R = 0.44$ coderate and fix the mapping to QPSK. Thus, we assume a fixed MCS because dynamic MCS selection requires additional work that is out of scope for this demonstrator. The GFDM system uses $K_\text{t} = 15$ timeslots, $K_\text{s} = 64$ subcarriers, and $K_\text{on} = 60$ active subcarriers and thus the multicarrier system may convey $N_\text{d} = 900$ complex symbols per frame. We set the CP and Cyclic Suffix (CS) length to $N_\text{CP} = 16$ and $N_\text{CS} = 8$ respectively. The preamble requires $K_\text{t} = 2$ additional timeslots with its own CP and CS. As a result, a complete frame consists of 1136 S at 30.72 MS s$^{-1}$ which results in a 36.98 µs on air duration, or burst duration. Finally, the occupied bandwidth in this system is 29.28 MHz.

Additionally, the system needs to organize multiple access and duplexing. Hence, the configuration accommodates TDD to integrate uplink and downlink transmissions as well as TDM to support multiple User Equipments (UEs). Since we expect periodic deterministic communication behavior, the system organizes access in cycles and assigns transmit slots to each UE and the RAN. To this end, the $T_\text{cycle}$ parameter controls the duration until another packet may be transmitted, i.e. a single UE may only transmit in its designated slot every $T_\text{cycle}$. Finally, we employ a static S&RA strategy where we statically assign resources to each device in every cycle. Any

[1]`https://kb.ettus.com/About_USRP_Bandwidths_and_Sampling_Rates`

dynamic behavior would require a feedback channel and a corresponding protocol that is out of scope for this work.

## 8.3 Software modem system design

Software system design is inherently difficult [Mar08]. Thus, it is strictly required to follow software design principles to build a complex system such as a Cloud RAN system that aims to deliver its promised benefits. These benefits are flexibility, reusability, efficiency and performance [MGG+12, GM13]. Still, a software system offers greater flexibility and is more accessible [Gra13]. Benefits include more rapid development cycles and thus features may be deployed faster. Abstracting away hardware implementation details offers the potential for broader adoption. Hardware development, even if not Application Specific Integrated Circuit (ASIC) focused, is inherently difficult and cumbersome. The author in [Gra13] makes a case for Field Programmable Gate Array (FPGA) development in case requirements cannot be met otherwise but recommends to stick to software development. Often, software development for RAN is carried out with several programming languages including C and C++ for performance critical code as well as Python for support functionality and testing [WO20]. However, one should never underestimate the burden of other legacy programming languages with outdated programming paradigms and missing features, such as an appropriate module import mechanism. Further, future development may benefit from more recent programming languages such as Rust.

### 8.3.1 Streaming and simulation software architecture

Arguably most communications engineering research is performed via simulations. Investigations may be scaled over a large set of parameters, e.g. SNRs, to obtain results in a timely manner. Importantly, resource intensive components of a simulation may slow down investigations but generally do not prohibit them. Further, such components may introduce latency but researchers control time within a simulation. In a simulation environment it may be useful to leverage inter-frame parallelism to speed up simulations because such implementations may be fed with data at an arbitrary rate.

In order to obtain useful models, measurements are conducted where recorded data may be replayed and analyzed at a later date. Again, the latency between recording and analysis may be high but not important. Beyond that, researchers may replay a recording and thus go back and forth in time. Thus, in these scenarios researchers have full control over all aspects of their investigation, e.g. time and hardware resources. Researchers require easy access to their simulation environment to alter and re-assemble it in all aspects. Under these constraints, optimized components help to obtain results faster and to focus on components that are currently under heavy investigation and thus are changed rapidly. However, optimized components need to integrate seamlessly into the simulation environment.

On the contrary, a streaming communications system is targeted at OTA live data. New data, e.g. samples, are produced at a constant rate. Thus, a communication system must be able to consume samples at that rate, extract the conveyed data, and reduce the overall rate. A slow component may introduce latency or limit the maximum sample rate that a system digests. If a component is unable to process data at a required rate, eventually the system will be flooded with data and be required to drop samples. Running multiple instances of such a component in parallel to meet the required rate in turn requires an environment that allows for concurrent operation while maintaining consistency. However, such parallelism does not decrease latency but may even increase it in cases where these components need to wait for more data such as in case of inter-frame parallelism.

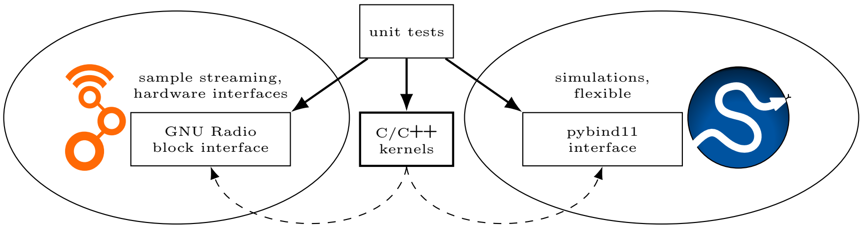


**Figure 8.4:** Software architecture with computational kernels, integration into different interfaces, and appropriate unit tests.

We opt for the flexible overall software architecture depicted in Fig. 8.4. Our software design encapsulates small, fixed computational kernels. These computational kernels are highly optimized and implemented in C/C++. A computational kernel is a C++ class that holds the required internal data structures and provides the optimized methods to compute a specific task, e.g. polar encoding. For our demonstrator testbed, we integrate these kernels into GNU Radio to leverage the streaming architecture for live OTA transmissions. Prior to an optimized implementation the kernel functionality and suitability is investigated in simulations in a Python implementation. Afterwards, the simulation infrastructure may be repurposed to verify the correct operation of such a kernel in unit tests as discussed in Sec. 8.3.2.

As indicated in Fig. 8.4, we are able to plug an optimized kernel into different interfaces. The GNU Radio block interface integration allows us to easily interface with hardware and leverage the multi-threading capabilities that GNU Radio provides. We use *pybind11* to integrate an optimized kernel into our Python simulations to speed them up and spend more resources on other components [JRS+22]. In fact, we use the SotA core scientific libraries *NumPy* and *SciPy* that heavily rely on that approach and thus, we follow their example [HMW+20, VGO+20].

### 8.3.2 Test driven development

As we want to leverage GPP capabilities to enable Cloud RAN, we need to follow software design principles to achieve this goal. The flowgraph blocks in Fig. 8.5 are decomposed into their components to enable easily verifiable unit tests to ensure correct operation. Here, we rely on SotA open source unit test frameworks, such as Python `unittest` and C++ `cppunit` to enable test driven development [Pyt21, FLS+21]. This enables rapid progress while maintaining correct operation because we can confidently optimize while verifying correct operation. Further, we facilitate integration tests to ensure units interact as expected. Finally, we expect a continuous integration system that builds every set of changes, runs all tests, and reports errors, if any, automatically. Unit tests are the foundation to build high-performance software that enables real-time operation and thus Cloud RAN.

### 8.3.3 Object oriented programming

In order to achieve fast function execution, Object Oriented (OO) programming aides to separate initialization tasks and actual function execution. This implies that each processing block in our transmitter and receiver chain is composed of one or more classes that hold a processing block's configuration, e.g. information and codeword size. Only data associated to individual frames is passed around during program execution, e.g. LLRs or complex symbols.

Another example would be a polar decoder where a constructor creates necessary buffers upon instantiation which is only performed once. The actual `decode` function reuses these structures. Naturally, we are interested in the `decode` function execution time because it impacts system throughput and latency, with possibly millions of function calls, while constructor and destructor tasks are only performed once.

## 8.4 Software environment

In this section, we discuss GNU Radio and the USRP Hardware Driver (UHD) along with important used software libraries that comprise our software environment. Further, the Linux host system and important configurations are discussed. In Sec. 8.5 we assume this environment to discuss our implemented OOT modules and how they are used.

### 8.4.1 Linux host

The Operating System (OS) and its configuration have a major impact on performance. We conduct all of our experiments on Ubuntu 20.04 with a Linux 5.x generic kernel. We expect that SDRs are run on a server or embedded system and thus we expect a Linux system. Further, essential SotA tools such as software package managers, can only be expected on these platforms. First, we increase the network buffer size to 62.5 MB, and thus follow *UHD* recommended settings[2]. Further, it is important to set the CPU performance governor to `performance` instead of `ondemand` or `powersave` which are common default options[2]. Finally, real-time scheduling is enabled for UHD and GNU Radio flowgraphs[2].

In order to plug our GNU Radio flowgraphs into the host network stack, we employ Linux Foo over UDP (FOU) tunneling [Cor14]. This approach allows us to easily add virtual network devices to a host and connect a flowgraph to them via UDP. A flowgraph adds GNU Radio *Socket PDU* blocks to bind and listen to UDP ports. With this approach we seamlessly integrate our wireless communication system into an IP based system. Thus, the full stack of network tools to configure and measure our system are available including `ping`, `ip` tools with flow control and rate limiting.

### 8.4.2 GNU Radio

GNU Radio is a modular, multi-threaded framework for SDR applications that offers a lot of standard capabilities for signal processing, visualization, infrastructure, and hardware interfaces in order to develop new waveforms [LMA+22]. A developer may focus on the actual algorithms at hand while GNU Radio deals with all the software design implications, e.g. multi-threading is a challenging software design topic itself [Hin13]. GNU Radio is freely available under the terms of the GNU General Public License v3.0 or later (GPLv3+) [Fre07].

[2]`https://kb.ettus.com/USRP_Host_Performance_Tuning_Tips_and_Tricks`

"If you've spent years learning tricks to make your multithreaded code work at all, let alone rapidly, with locks and semaphores and critical sections, you will be disgusted when you realize it was all for nothing. If there's one lesson we've learned from 30+ years of concurrent programming, it is: just don't share state . It's like two drunkards trying to share a beer. It doesn't matter if they're good buddies. Sooner or later, they're going to get into a fight. And the more drunkards you add to the table, the more they fight each other over the beer. The tragic majority of multithreaded applications look like drunken bar fights." [Hin13]

GNU Radio offers two Application Programming Interfaces (APIs) to convey data from one DSP block to the next in a multi-threaded system, namely the stream and the message passing API. The streaming API is targeted at efficient, thread-safe data conveyance between blocks that target continuous data. This interface allows for the highest sample throughput. The message passing API targets thread-safe online re-configuration and packetized data exchange between blocks. Besides, both APIs provide thread-safe multi-threading by scheduling each block in its own thread and taking care of concurrency implications. Thus, developers may focus on the DSP software implementation at hand.

Conceptionally, GNU Radio is a framework with numerous modules that extend its functionality while the core runtime offers a block scheduler and a standardized interface. A lot of modules with specific blocks are available as part of the GNU Radio core distribution, e.g. *gr-fft* for Fastest Fourier Transform in The West (FFTW), *gr-uhd* for UHD, and *gr-digital* for common DSP operations. Besides these in-tree modules, Out-Of-Tree (OOT) modules exist to extend GNU Radio with user specific functionality. We use OOT modules to plug our DSP functions into GNU Radio. GNU Radio relies on optimized software libraries that we employ in our OOT modules and thus introduce here shortly.

**Vector-Optimized Library of Kernels (VOLK)** A library of math functions which are typically used in signal processing [DDA+22]. It makes use of Single-Instruction-Multiple-Data (SIMD) extensions which are present in many modern GPP hardware architectures such as x86 64bit (x86) Streaming SIMD Extensions (SSE), x86 Advanced Vector Extensions (AVX), or ARM (ARM) NEON (NEON). It abstracts individual implementations for specific hardware and provides a canonical interface to all of them.

**Fastest Fourier Transform in The West (FFTW)** One of the fastest known software implementations for Fourier transforms [FJ05]. It is a de facto standard for many software projects, both commercial and open-source.

## 8.4.3 USRP hardware driver

We consider USRPs for our testbed implementation and thus, we require the USRP Hardware Driver (UHD) to exchange data between the device and a host [Ett21]. UHD facilitates multiple essential functions that include efficient sample streaming from and to a device, and access to numerous frontend parameters such as transmit and receive gain, and carrier frequency. Further, it provides an efficient abstract interface for all USRP products regardless of underlying hardware details such as the data transport technology, e.g. IEEE 802.3 Ethernet (Ethernet), and Universal Serial Bus (USB). Besides, various approaches to reduce system load and latency and improve reliability are available and investigated, such as UHD with Data Plane Development Kit (DPDK) [BM20].

USRPs are SotA SDR frontends that offer features beyond continuous sample streaming and interactive RF frontend configuration. We require a high and sustained sampling rate with low latency transfers between the frontend and our CPU. SDR frontends need to offer low latency data transfers between the frontend and the host and they need to offer a timed commands feature to start a burst transmission at a precise time. Timed commands need to enable users to control frontend properties with high timing precision as well, e.g. re-tune to a different radio frequency. In combination with the UHD, USRPs provide precise transmit timing control, as well as precise receive timing information. This feature set is particularly required for communication technologies that rely on precise synchronization between clients such as LTE or 5G NR.

The *gr-uhd* module provides access to UHD and is part of the GNU Radio core distribution. Earlier versions of the UHD were part of the GNU Radio core distribution and USRPs were first developed as an SDR frontend for GNU Radio [Sta06, PND$^+$11].

## 8.4.4 Software benchmarking tools

It is crucial to gather performance metrics to identify resource hot spots. Naturally, the results depend on the chosen system parameters, the chosen platform, and implementation details may have an even greater impact on the exact results [Gia16]. We focus on GPP hardware in order to leverage software technologies and flexibility.

We mainly use the two programming languages Python and C++. In most cases we are interested in a functions execution time in order to determine the share of resources we need to dedicate to it. It is possible to measure latency with C++ `std::chrono` or derivatives thereof [cpp21a, Gia16]. In case of Python function benchmarks,the Python `time` and `timeit` modules provide time measurement tools [Pyt21]. The function execution duration is computed via timestamps right before and after a function call.

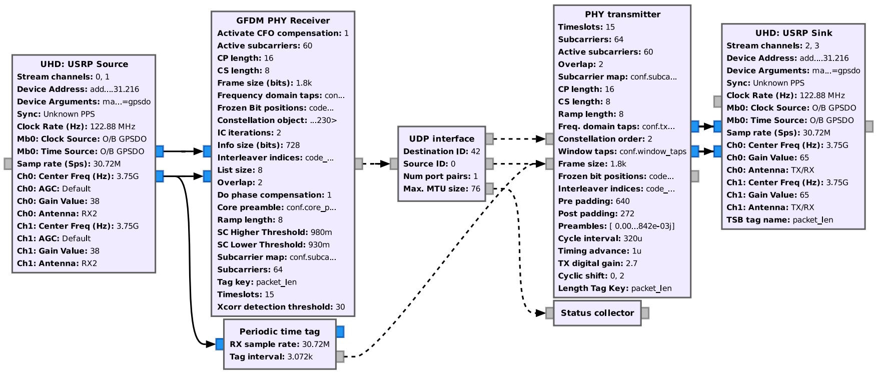
It is inherently difficult to measure latency and throughput though and full of caveats. Difficulties to measure range from finite clock resolution to complex CPU architectures with caches, speculative execution, multi-threading, to frequency-scaling. Further, current compilers are very good at optimizing code for speed, up to the point where the function under test is optimized out if not handled properly. In order to minimize issues, we adopt the micro-benchmarking tool `benchmark` that enables accurate DSP processing latency measurements in C++ [Goo21]. Here, latency is defined as the time it takes to execute one function call. Throughout this work, we measure a whole set of functions that need to be executed repeatedly in an OTA system, e.g. decode, demodulate, or demap. However, any kind of setup computations, such as of memory allocation, are excluded from this measurement because they are executed once during system initialization and re-used in every call to the function under test. Moreover, the throughput is defined as the number of elements that are processed by the corresponding function per second. Depending on the function this might be the number of bits, LLRs, or complex samples that are passed to the function under test, or that the function under test yields. For example a decoder achieves a coded throughput on the input where it considers the $N$ bit per function call while information throughput at the output considers the $K$ bit per function call. Thus, the information throughput is defined as the number of information bit $K$ divided by the function execution latency in $\mathrm{bit\,s^{-1}}$. All values are directly measured and computed via the micro-benchmarking tool `benchmark`.

## 8.5 Implemented software components

Based on our software environment, discussed in Sec. 8.4, we discuss our software components that comprise the testbed communication application. The GNU Radio flowgraph in Fig. 8.5 represents a top-level view that we discuss in more detail in this section. The *UHD: USRP Source* block serves as a receiver interface for USRPs. Similarly, the *UHD: USRP Sink* block serves as the transmitter interface for USRPs. While these blocks are separate in this flowgraph, they may represent a single USRP or a multiple USRP configuration where transmit and receive antennas are distributed arbitrarily between physical devices.

The *Periodic time tag* block consumes time tags offered by the UHD to achieve exact timing information for accurate burst transmit timing[3] as discussed in Sec. 8.5.1. The *status collector* block gathers metadata, such as SNR estimates, which is ultimately stored in a database for analysis and visualization.

The *UDP Interface* block is a hierarchical block that is discussed in Sec. 8.5.1. The hierarchical transmitter and receiver blocks will be discussed shortly because each consists of numerous blocks from several OOT modules.



**UHD: USRP Source**
**Stream channels:** 0, 1
**Device Address:** add....31.216
**Device Arguments:** ma...=gpsdo
**Sync:** Unknown PPS
**Clock Rate (Hz):** 122.88 MHz
**Mb0: Clock Source:** O/B GPSDO
**Mb0: Time Source:** O/B GPSDO
**Samp rate (Sps):** 30.72M
**Ch0: Center Freq (Hz):** 3.75G
**Ch0: AGC:** Default
**Ch0: Gain Value:** 38
**Ch0: Antenna:** RX2
**Ch1: Center Freq (Hz):** 3.75G
**Ch1: AGC:** Default
**Ch1: Gain Value:** 38
**Ch1: Antenna:** RX2

**GFDM PHY Receiver**
**Activate CFO compensation:** 1
**Active subcarriers:** 60
**CP length:** 16
**CS length:** 8
**Frame size (bits):** 1.8k
**Frequency domain taps:** con...
**Frozen Bit positions:** code...
**Constellation object:** ...230>
**IC iterations:** 2
**Info size (bits):** 728
**Interleaver indices:** code_...
**List size:** 8
**Overlap:** 2
**Do phase compensation:** 1
**Core preamble:** conf.core_p...
**Ramp length:** 8
**SC Higher Threshold:** 980m
**SC Lower Threshold:** 930m
**Subcarrier map:** conf.subca...
**Subcarriers:** 64
**Tag key:** packet_len
**Timeslots:** 15
**Xcorr detection threshold:** 30

**Periodic time tag**
**RX sample rate:** 30.72M
**Tag interval:** 3.072k

**UDP interface**
**Destination ID:** 42
**Source ID:** 0
**Num port pairs:** 1
**Max. MTU size:** 76

**PHY transmitter**
**Timeslots:** 15
**Subcarriers:** 64
**Active subcarriers:** 60
**Overlap:** 2
**Subcarrier map:** conf.subca...
**CP length:** 16
**CS length:** 8
**Ramp length:** 8
**Freq. domain taps:** conf.tx...
**Constellation order:** 2
**Window taps:** conf.window_taps
**Frame size:** 1.8k
**Frozen bit positions:** code...
**Interleaver indices:** code_...
**Pre padding:** 640
**Post padding:** 272
**Preambles:** [ 0.00...842e-03j]
**Cycle interval:** 320u
**Timing advance:** 1u
**TX digital gain:** 2.7
**Cyclic shift:** 0, 2
**Length Tag Key:** packet_len

**Status collector**

**UHD: USRP Sink**
**Stream channels:** 2, 3
**Device Address:** add....31.216
**Device Arguments:** ma...=gpsdo
**Sync:** Unknown PPS
**Clock Rate (Hz):** 122.88 MHz
**Mb0: Clock Source:** O/B GPSDO
**Mb0: Time Source:** O/B GPSDO
**Samp rate (Sps):** 30.72M
**Ch0: Center Freq (Hz):** 3.75G
**Ch0: Gain Value:** 65
**Ch0: Antenna:** TX/RX
**Ch1: Center Freq (Hz):** 3.75G
**Ch1: Gain Value:** 65
**Ch1: Antenna:** TX/RX
**TSB tag name:** packet_len

**Figure 8.5:** Top-level GNU Radio flowgraph with $N_{\text{ant}} = 2$ that is executed on our Cloud RAN and AGV communication system.

## 8.5.1 Testbed software integration

The overall configuration of our OTA implementation is subsumed in the GNU Radio OOT module *gr-tacmac* [Dem23]. Here, we present the hierarchical flowgraphs that constitute the overall OTA demonstrator shown in Fig. 8.5. The flowgraph in Fig. 8.5 includes all DSP to transmit and receive in the uplink and downlink. Furthermore, this flowgraph is executed on our Cloud RAN system as well as on our AGV communication system as discussed in Sec. 8.6. The presented hierarchical flowgraphs are available in the GNU Radio Graphical User Interface (GUI), the GNU Radio Companion (GRC), however, we implement them as hierarchical Python blocks as well because this approach offers greater flexibility.

The GRC is a GUI to simplify flowgraph creation and execution. Within GRC the *Pad source* and *Pad sink* blocks indicate inputs and outputs for hierarchical GNU Radio blocks. These *Pad* blocks result in block ports in

[3]`https://kb.ettus.com/UHD#Sample_streaming`

e.g. Fig. 8.5. We omit the *Pad* blocks in most flowgraphs in favor of greater clarity.

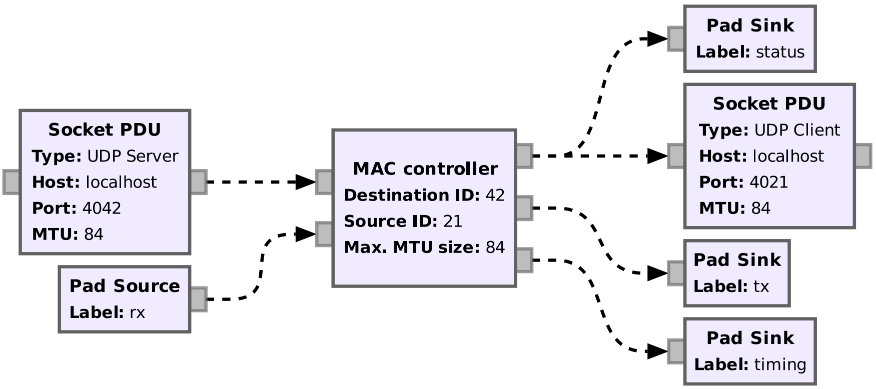
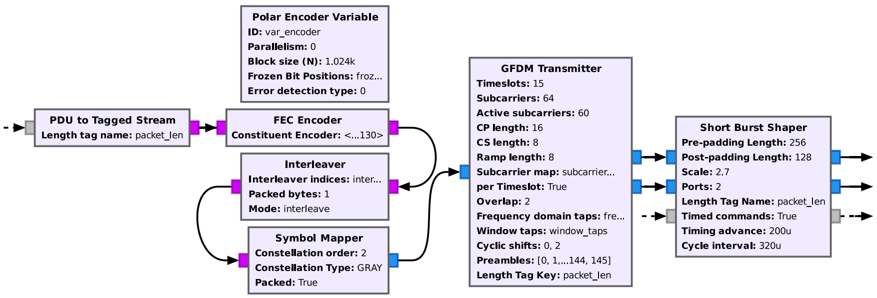


**Figure 8.6:** Internals of the *gr-tacmac UDP interface* block in Fig. 8.5 for FOU.

The GNU Radio flowgraph is connected to the Linux network stack via FOU as discussed in Sec. 8.4.1. This MAC functionality shown in Fig. 8.6 is encapsulated in the hierarchical flowgraph *UDP interface* in Fig. 8.5. We use *Socket PDU* blocks to connect to UDP ports and we implement a *MAC controller* block that takes care of MAC processing. Within the message passing API, a Protocol Data Unit (PDU) is a structured unit that contains a dictionary with meta data and a uniform data vector, e.g. a byte vector. The *MAC controller* block controls transmission and reception. Based on the given configuration, it adds a custom header to each packet based on source and destination ID, the packet length, a timestamp, and a checksum as discussed in Sec. 8.2. At the receiver, it parses this header information, rejects invalid packets and extracts data packets. Packets above the configured `MTU size` are rejected. Instead, we rely on Linux network stack features to perform packet segmentation and re-assembly. Fig. 8.6 shows a client that is configured with the shown configuration and connects to a base station that replicates the *Socket PDU* UDP server, *MAC controller*, and *Socket PDU* UDP client structure for each client. We use the GNU Radio message passing API because it allows to connect an arbitrary number of thread-safe connections to a single port and thus enables packet interleaving. Besides, this part of the overall flowgraph processes packets and thus is predestined for this API.

The transmitter flowgraph is presented in Fig. 8.7. It accepts PDUs that are converted to the stream API in the *PDU to Tagged Stream* block and then propagate through the whole PHY processing chain. This includes a polar


**Polar Encoder Variable**
**ID:** var_encoder
**Parallelism:** 0
**Block size (N):** 1.024k
**Frozen Bit Positions:** froz...
**Error detection type:** 0

**PDU to Tagged Stream**
**Length tag name:** packet_len

**FEC Encoder**
**Constituent Encoder:** <...130>

**Interleaver**
**Interleaver indices:** inter...
**Packed bytes:** 1
**Mode:** interleave

**Symbol Mapper**
**Constellation order:** 2
**Constellation Type:** GRAY
**Packed:** True

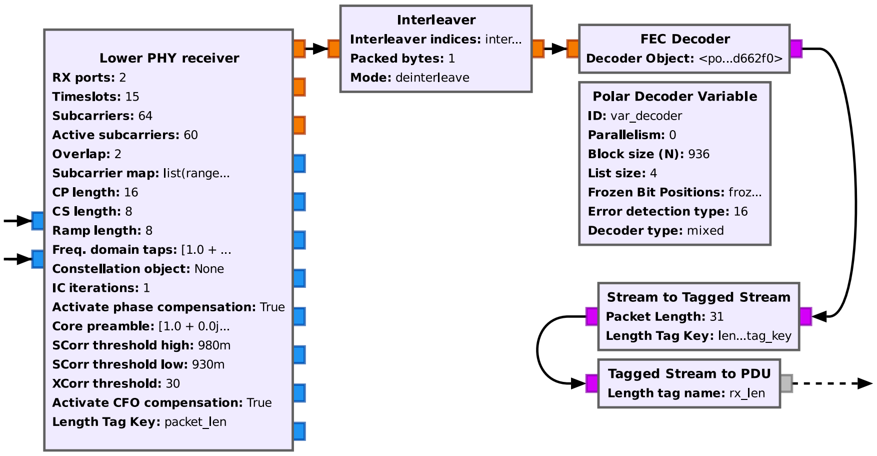
**GFDM Transmitter**
**Timeslots:** 15
**Subcarriers:** 64
**Active subcarriers:** 60
**CP length:** 16
**CS length:** 8
**Ramp length:** 8
**Subcarrier map:** subcarrier...
**per Timeslot:** True
**Overlap:** 2
**Frequency domain taps:** fre...
**Window taps:** window_taps
**Cyclic shifts:** 0, 2
**Preambles:** [0, 1,...144, 145]
**Length Tag Key:** packet_len

**Short Burst Shaper**
**Pre-padding Length:** 256
**Post-padding Length:** 128
**Scale:** 2.7
**Ports:** 2
**Length Tag Name:** packet_len
**Timed commands:** True
**Timing advance:** 200u
**Cycle interval:** 320u

**Figure 8.7:** Internals of the *gr-tacmac PHY transmitter* block in Fig. 8.5 with $N_{\text{ant}} = 2$.

encoder in the *FEC Encoder* block, an *Interleaver* block, a *Symbol Mapper* block and a *GFDM Transmitter* which are discussed in detail sections 8.5.5, 8.5.4, 8.5.3, and 8.5.2.

The *Short Burst Shaper* block handles zero-padding and adds timed-command stream tags for a *UHD: USRP Sink* block to control the exact transmit time. Zero-padding is required for burst transmissions to ensure that data samples are not corrupted by switching any frontend components. This block may consume periodic timing information to better align timed burst transmission with a USRP device.


**Lower PHY receiver**
**RX ports:** 2
**Timeslots:** 15
**Subcarriers:** 64
**Active subcarriers:** 60
**Overlap:** 2
**Subcarrier map:** list(range...
**CP length:** 16
**CS length:** 8
**Ramp length:** 8
**Freq. domain taps:** [1.0 + ...
**Constellation object:** None
**IC iterations:** 1
**Activate phase compensation:** True
**Core preamble:** [1.0 + 0.0j...
**SCorr threshold high:** 980m
**SCorr threshold low:** 930m
**XCorr threshold:** 30
**Activate CFO compensation:** True
**Length Tag Key:** packet_len

**Interleaver**
**Interleaver indices:** inter...
**Packed bytes:** 1
**Mode:** deinterleave

**FEC Decoder**
**Decoder Object:** <po...d662f0>

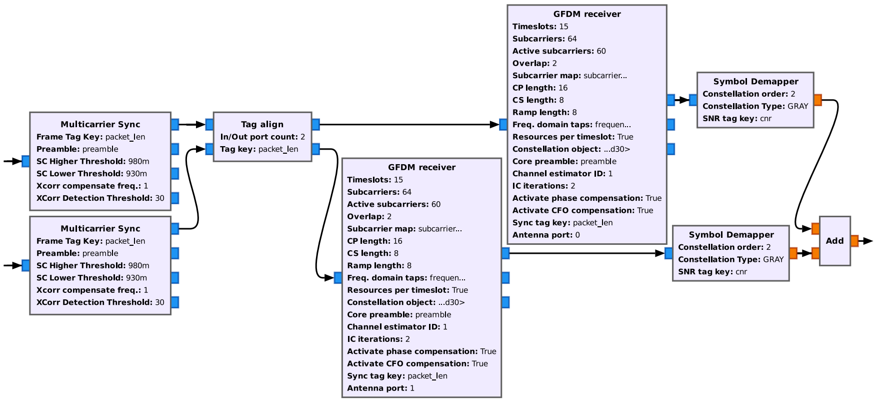
**Polar Decoder Variable**
**ID:** var_decoder
**Parallelism:** 0
**Block size (N):** 936
**List size:** 4
**Frozen Bit Positions:** froz...
**Error detection type:** 16
**Decoder type:** mixed

**Stream to Tagged Stream**
**Packet Length:** 31
**Length Tag Key:** len...tag_key

**Tagged Stream to PDU**
**Length tag name:** rx_len

**Figure 8.8:** Internals of the *gr-tacmac GFDM PHY Receiver* block in Fig. 8.5 with $N_{\text{ant}} = 2$.

The details of the receiver block in Fig. 8.5 are shown in Fig. 8.8 while the lower PHY receiver is shown in Fig. 8.9 for clarity. The *Lower PHY receiver* block provides LLRs that are deinterleaved in the *Interleaver* block and polar decoded in the *FEC Decoder* block. Since we consider constant size packets only, the *Stream to Tagged Stream* and *Tagged Stream to PDU* blocks can be used to convert chunks of bytes into PDUs that contain PHY receiver meta data from stream tags in a corresponding dictionary. This information includes precise timing information, DSP delay, per antenna stream SNR estimates, and per antenna CNR estimates.



**Multicarrier Sync**
**Frame Tag Key:** packet_len
**Preamble:** preamble
**SC Higher Threshold:** 980m
**SC Lower Threshold:** 930m
**Xcorr compensate freq.:** 1
**XCorr Detection Threshold:** 30

**Multicarrier Sync**
**Frame Tag Key:** packet_len
**Preamble:** preamble
**SC Higher Threshold:** 980m
**SC Lower Threshold:** 930m
**Xcorr compensate freq.:** 1
**XCorr Detection Threshold:** 30

**Tag align**
**In/Out port count:** 2
**Tag key:** packet_len

**GFDM receiver**
**Timeslots:** 15
**Subcarriers:** 64
**Active subcarriers:** 60
**Overlap:** 2
**Subcarrier map:** subcarrier...
**CP length:** 16
**CS length:** 8
**Ramp length:** 8
**Freq. domain taps:** frequen...
**Resources per timeslot:** True
**Constellation object:** ...d30>
**Core preamble:** preamble
**Channel estimator ID:** 1
**IC iterations:** 2
**Activate phase compensation:** True
**Activate CFO compensation:** True
**Sync tag key:** packet_len
**Antenna port:** 0

**GFDM receiver**
**Timeslots:** 15
**Subcarriers:** 64
**Active subcarriers:** 60
**Overlap:** 2
**Subcarrier map:** subcarrier...
**CP length:** 16
**CS length:** 8
**Ramp length:** 8
**Freq. domain taps:** frequen...
**Resources per timeslot:** True
**Constellation object:** ...d30>
**Core preamble:** preamble
**Channel estimator ID:** 1
**IC iterations:** 2
**Activate phase compensation:** True
**Activate CFO compensation:** True
**Sync tag key:** packet_len
**Antenna port:** 1

**Symbol Demapper**
**Constellation order:** 2
**Constellation Type:** GRAY
**SNR tag key:** cnr

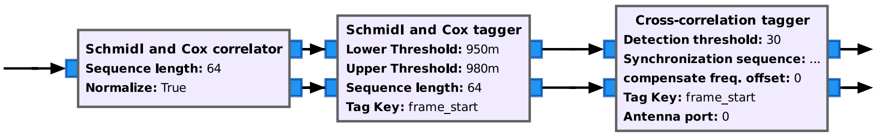
**Symbol Demapper**
**Constellation order:** 2
**Constellation Type:** GRAY
**SNR tag key:** cnr

**Figure 8.9:** Internals of the *gr-tacmac Lower PHY receiver* in Fig. 8.8 block with PHY synchronization, GFDM demodulation, and demapping receiver flowgraph for $N_{\text{ant}} = 2$.

The flowgraph in Fig. 8.9 details a lower PHY receiver for a $N_{\text{ant}} = 2$ antenna setup that leverages diversity to improve reliability. A continuous stream, e.g. at $30.72\,\text{MS}\,\text{s}^{-1}$ per antenna port, is fed to a *Multicarrier Sync* hierarchical block as discussed in Sec. 8.5.2. The start of frame stream tags are aligned in the *Tag align* block to ensure each *Multicarrier Sync* stream produces the same number of aligned frames. UHD provides a convenient interface that aligns sample streams across devices and we leverage this feature. The *GFDM receiver* blocks extract frames based on aligned frame start sample tags and perform GFDM demodulation as discussed in Sec. 8.5.3. Finally, the *Symbol Demapper* blocks use the received complex samples together with stream tags that contain SNR and CNR estimates to compute LLRs for FEC decoding. Here we use receive diversity to obtain multiple observations of the same frame and thus, we can use the *Add* block to sum up LLRs in accordance with (6.4) to enhance the FEC decoder process with more information.

## 8.5.2 Synchronization

Reliable and robust synchronization is paramount to any communication system because all further DSP relies on synchronized data. In [DKJ15], it has been shown that frame synchronization is a computationally expensive operation. We employ the multicarrier synchronization proposed in [AKE08], which is an improved Schmidl&Cox-based preamble approach [SC97, GMMF14]. A Schmidl&Cox preamble $\boldsymbol{y}_\mathrm{P}$ consists of two identical parts. This preamble is extended with a CP and CS to a baseband preamble $\boldsymbol{x}_\mathrm{P}$ and prepended to a frame $\boldsymbol{x}$ to obtain a baseband signal $\boldsymbol{s}$ that is ready for transmission. The improved Schmidl&Cox (Schmidl&Cox) approach relies on coarse synchronization with a fixed-lag auto-correlation followed by a cross-correlation with the known preamble. By combining both correlations, this approach yields high precision timing information. We use a Zadoff-Chu sequence to generate preamble symbols. Other strategies to obtain a Schmidl&Cox preamble are available as well [ZM09]. Also, Schmidl&Cox preambles are well suited for SNR and CNR estimation [ZM09]. Here, we discuss our efficient implementation *XFDMSync* [DG22].


Schmidl and Cox correlator
Sequence length: 64
Normalize: True

Schmidl and Cox tagger
Lower Threshold: 950m
Upper Threshold: 980m
Sequence length: 64
Tag Key: frame_start

Cross-correlation tagger
Detection threshold: 30
Synchronization sequence: ...
compensate freq. offset: 0
Tag Key: frame_start
Antenna port: 0

**Figure 8.10:** Internals of the *XFDMSync Multicarrier Sync* hierarchical flowgraph in Fig. 8.9.

The GNU Radio flowgraph in Fig. 8.10 shows the *XFDMSync* Schmidl&Cox synchronization blocks that are encapsulated in a *Multicarrier Sync* block in Fig. 8.9. A continuous stream of samples from a source, e.g. a USRP, is fed to the *Schmidl and Cox correlator* block. First, the algorithm performs a fixed-lag auto-correlation of length $N_{y_\mathrm{P}}/2$ which yields a frame timing estimation with moderate accuracy. Furthermore, the fixed-lag auto-correlation is used to estimate a Carrier-Frequency-Offset (CFO). Auto-correlation peak detection as well as CFO calculation is performed by the *Schmidl and Cox tagger* block. This block searches for a peak in a window that start with the first sample above the *Upper Threshold* and ends with the last sample above the *Lower Threshold*. The block adds a GNU Radio sample tag to the peak sample. Afterwards, the *Cross-correlation tagger* block computes a cross-correlation with $\boldsymbol{y}_\mathrm{P}$ around the detected peak. In this window, element-wise multiplication of the fixed-lag auto-correlation and cross-correlation values results in a high precision timing synchronization

[AKE08]. The result is a sample stream with sample tags that denote high precision timing information. Fig. 8.9 includes multiple hierarchical synchronization blocks and employs a *Tag align* block to ensure tags from multiple parallel synchronization streams are aligned. Eventually, subsequent blocks extract received frame vectors $\tilde{\boldsymbol{s}}$ that we can decompose into a received preamble $\tilde{\boldsymbol{x}}_{\mathrm{P}}$ and a received frame $\tilde{\boldsymbol{x}}$.

It is important to note that synchronization runs at full receiver input rate, e.g. 61.44 MS/s. Measurements on a AMD Ryzen Threadripper 3970X (TRX3970X) with a GNU Radio *Probe rate* block indicate that the presented set up is able to sustain 120 MS/s throughput for a single stream. A two stream configuration may sustain 110 MS/s throughput including additional tag alignment work. The bulk of the computional burden lies on the *Schmidl and Cox correlator* block which may be separated into multiple blocks that are executed concurrently if the need for higher throughput arises.

## 8.5.3 GFDM implementation

We already discussed multicarrier modulation theory in Chapter 5, here we discuss implementation aspects. The *gr-gfdm* module [DRKK22] is a joint effort that we presented in previous works [DBD$^+$17b, DBD17a] and it is freely available under the terms of the GPLv3+ [Fre07]. Besides the GNU Radio API integration, *pybind11* provides a Python API for our C++ implementation [Pyt21, JRS$^+$22]. The approach to separate the optimized C++ implementation from a specific API ensures that we benefit from algorithmic optimizations in our OTA demonstrator as well as in simulations as discussed in Sec. 8.3. Like GNU Radio, *gr-gfdm* heavily relies on Vector-Optimized Library of Kernels (VOLK) and FFTW for optimizations as discussed in Sec. 8.4.2 [DDA$^+$22, FJ05].

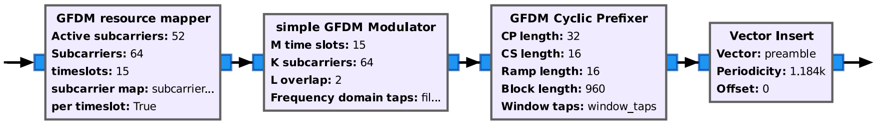
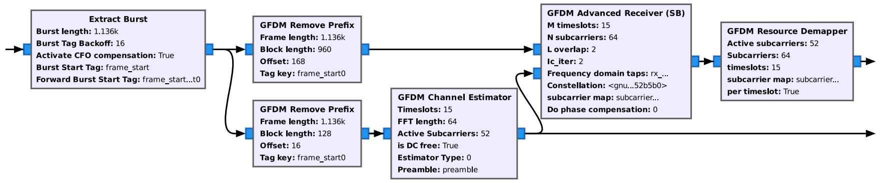


**Figure 8.11:** Internal flowgraph of the *gr-gfdm GFDM Transmitter* block in Fig. 8.7.

The transmitter flowgraph in Fig. 8.11 implements the GFDM multicarrier modulation discussed in Sec. 5.3.3 and corresponds to the hierarchical *GFDM Transmitter* block in Fig. 8.7. The input is a stream of complex symbols $\boldsymbol{d}$ where $K_{\mathrm{t}}K_{\mathrm{on}}$ symbols are mapped to a resource grid in the *GFDM resource mapper* block, and transformed to GFDM frames $\boldsymbol{y}$ in the *simple GFDM*

*Modulator* block. In case of OFDM, it is possible to replace the *simple GFDM Modulator* block by a GNU Radio *FFT* block. Subsequently, CP and CS are added in the *GFDM Cyclic Prefixer* block and finally, a pre-computed preamble $\boldsymbol{x}_\mathrm{P}$ is pre-pended to every such frame. Suitable preambles $\boldsymbol{x}_\mathrm{P}$ are discussed in Sec. 8.5.2. The *gr-gfdm* module provides a *GFDM Transmitter* block that includes all previously discussed blocks. Our software architecture specifically aims for this flexibility by separating the block functionality in computational kernels from the API. Since most blocks shown in Fig. 8.11 are computationally lightweight, this measure offers the potential to decrease transmitter DSP latency by avoiding multi-threading overhead.


**Extract Burst**
**Burst length:** 1.136k
**Burst Tag Backoff:** 16
**Activate CFO compensation:** True
**Burst Start Tag:** frame_start
**Forward Burst Start Tag:** frame_start...t0

**GFDM Remove Prefix**
**Frame length:** 1.136k
**Block length:** 960
**Offset:** 168
**Tag key:** frame_start0

**GFDM Remove Prefix**
**Frame length:** 1.136k
**Block length:** 128
**Offset:** 16
**Tag key:** frame_start0

**GFDM Channel Estimator**
**Timeslots:** 15
**FFT length:** 64
**Active Subcarriers:** 52
**is DC free:** True
**Estimator Type:** 0
**Preamble:** preamble

**GFDM Advanced Receiver (SB)**
**M timeslots:** 15
**N subcarriers:** 64
**L overlap:** 2
**Ic_iter:** 2
**Frequency domain taps:** rx_...
**Constellation:** <gnu...52b5b0>
**subcarrier map:** subcarrier...
**Do phase compensation:** 0

**GFDM Resource Demapper**
**Active subcarriers:** 52
**Subcarriers:** 64
**timeslots:** 15
**subcarrier map:** subcarrier...
**per timeslot:** True

**Figure 8.12:** Internal flowgraph of the *gr-gfdm GFDM receiver* block in Fig. 8.9.

The receiver counterpart of the GFDM transmitter is depicted in Fig. 8.12. The *Extract Burst* block expects a GNU Radio stream tag as discussed in Sec. 8.5.2 and extracts received frames $\tilde{\boldsymbol{s}}$ including the preamble and GFDM frame. The *GFDM Remove Prefix* blocks extract the modulated receive frame $\tilde{\boldsymbol{y}}$ and the preamble $\tilde{\boldsymbol{y}}_\mathrm{P}$. Next, the *GFDM Channel Estimator* block produces a new channel estimate for every input preamble that is interpolated to the GFDM frame size as well as an SNR estimate. Finally, a CNRs estimate is produced. Both, the SNR and the CNRs estimate, are propagated to downstream blocks via GNU Radio stream tags.

The core of the *gr-gfdm* module is the *GFDM Advanced Receiver (SB)* block. This block equalizes and demodulates frames, and further performs interference cancellation. The final processing block is the *GFDM Resource Demapper* block that yields complex received symbols $\tilde{\boldsymbol{d}}$ for downstream processing.

### 8.5.4 Symbol mapping

The theory of Bit-Interleaved Coded Modulation (BICM) is discussed in Chapter 4. The blocks in Fig. 8.13 represent the necessary DSP operations for BICM that are included in the *gr-symbolmapping* GNU Radio OOT module [Dem22a].

The *Interleaver* block is configurable in multiple ways. It offers an inter-

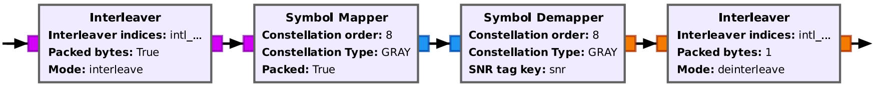


**Figure 8.13:** *gr-symbolmapping* blocks flowgraph

leaver and a deinterleaver mode, it may be configured for multiple input and output types, and it can operate on *packed* bytes. The implications of *packed* versus *unpacked* bytes are discussed in Sec. 4.6. Finally, The actual interleaver indices are an arbitrary list that must only contain unique indices, while the corresponding deinterleaver indices are automatically derived. The interleaver and deinterleaver implementations were upstreamed into GNU Radio as part of the *gr-blocks* module [LMA+22].

The *Symbol Mapper* block accepts packed or unpacked bytes and emits complex symbols from various constellations, including QPSK, 16QAM, and 256QAM as discussed in Chapter 4. The *Symbol Demapper* block processes complex symbols and produces LLRs for every bit. The actual LLR values may be dynamically scaled via GNU Radio stream tags that specify the current SNR or stream tags that contain a vector with CNRs that are applied repeatedly. Thus, CNRs are used under the assumption that a multicarrier system maps symbols to resources per timeslot.

### 8.5.5 Polar codes

We consider polar codes for URLLC with short packets. The error correction performance as well as the raw implementation latency are discussed in Chapter 3. Here, we discuss *gr-polarwrap*, a module that integrates the *polar-codes* project into a GNU Radio OOT [DL22, Dem22c]. The OOT module *gr-polarwrap* implements the GNU Radio *FECAPI* to provide polar encoder and decoder objects that are plugged into *FEC Encoder* and *FEC Decoder* blocks as shown in Fig. 8.14. The *FECAPI* provides a convenient separation between FEC implementations and the GNU Radio streaming and message passing APIs that we utilize.

### 8.5.6 Latency measurements

We focus on latency measurements because communication system latencies are considered to be crucial for URLLC applications. The authors in [BMH19] conducted profiling measurements and suggested to use GNU Radio stream tags with timestamps for latency measurements [Blo21]. We pick up this idea and implement functionality to measure DSP latency in the GNU Radio OOT module *gr-latency* [Dem22b]. The software benchmarking tools,

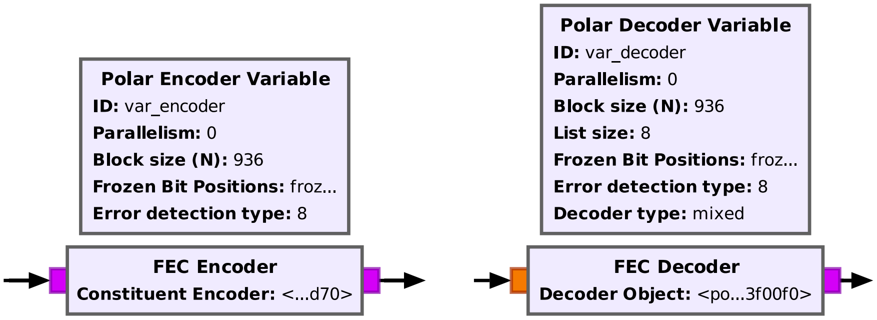
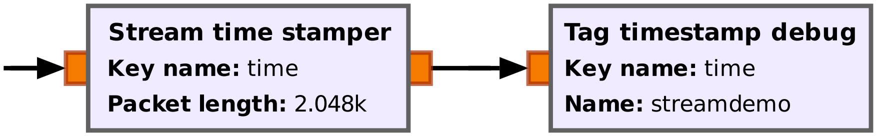
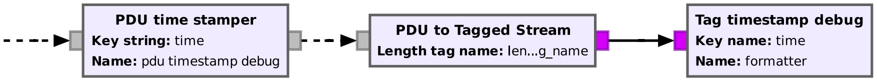


**Figure 8.14:** *gr-polarwrap* blocks and GNU Radio *FECAPI* integration.

discussed in Sec. 8.4.4, follow a different paradigm and thus, we consider this OOT module to be a crucial component for live latency measurements.



**(a)** *gr-latency* streaming latency measurement



**(b)** *gr-latency* message passing latency measurement

**Figure 8.15:** *gr-latency* components

The available GNU Radio blocks in this module are shown in Fig. 8.15. Either, we add timestamps periodically to stream samples with the *Stream time stamper* block or we use the message passing API with the *PDU time stamper* block. In any case, a high precision timestamp obtained with C++ `std::chrono` is added and later the propagation delay, or latency, is measured in a downstream *Tag timestamp debug* block. The high precision timestamp with ns resolution is obtained from the local system clock. It is important to note that some OSs lack clocks with µs precision and render this tool as well as many others useless.

## 8.6 Hardware components

The demonstrator setup comprises many hardware components that we want to introduce in this section. It consists of AGVs, computers, USRPs and radio equipment.

Our Cloud RAN is illustrated in Fig. 8.16, where we consider systems with an AMD Ryzen Threadripper 3970X (TRX3970X), 64 GB Random Access Memory (RAM), and a AMD Radeon RX 550 (Radeon RX 550) Graphics Processing Unit (GPU). Further, each system is equipped with an Intel Ethernet Converged Network Adapter X710-DA2 (Intel X710)[4] to connect USRPs over IEEE 802.3 Ethernet (Ethernet) with 10 Gbit $\mathrm{s}^{-1}$. One or more Ettus USRP N310 (N310) are connected to a Cloud RAN to realize a distributed AP setup. Multiple distributed AP are synchronized via an Ettus Octoclock-G (Octoclock-G). Our hardware platform focuses on x86 hardware with AVX SIMD extensions that are mostly available on hardware from Advanced Micro Devices (AMD) and Intel [Int21]. Multiple other GPP Instruction Set Architectures (ISAs) are available such as ARM NEON or RiscV (RiscV) but are out of scope for this work.

Our AGVs are Götting KATEs[5] and thus these devices are our mobile units as shown in Fig. 8.17. We replace the provided communication module with our solution by simply plugging it in at the internal Götting KATE Ethernet port to replace the default Wi-Fi bridge. Each Götting KATE carries an additional power supply Jauch Quartz Power Station JES1500WHA (JES1500)[6] to power our experimental radio equipment. Here, we use a B210 connected via USB to a compact computer with an AMD Ryzen 9 5900X (Ryzen 5900X) CPU, 32 GB RAM and a Radeon RX 550. The GPU serves as an optional debug interface only.

We selected the Radeon RX 550 because it is reliably and fully supported within Linux via the *AMDGPU* driver[7]. This feature sets the Radeon RX 550 apart from most other cheap discrete GPUs where manufacturer driver support is a constant source of frustration. The purpose of a GPU in our demonstrator project is to serve as a reliable debugging and visualization tool.

For the Hildesheim testbed in Sec. 8.7.3 we use a NI USRP-2974 (USRP-

[4]`https://ark.intel.com/content/www/us/en/ark/products/83964/intel-ethernet-converged-network-adapter-x710da2.html`

[5]`https://www.goetting-agv.com/kate`

[6]`https://www.jauch.com/en-INT/products/battery_technology/mobile-lithium-energy-storage/getPrm/energysolutions/POWERSTATIONEN/JES1500WHA/`

[7]`https://cgit.freedesktop.org/xorg/driver/xf86-video-amdgpu/`

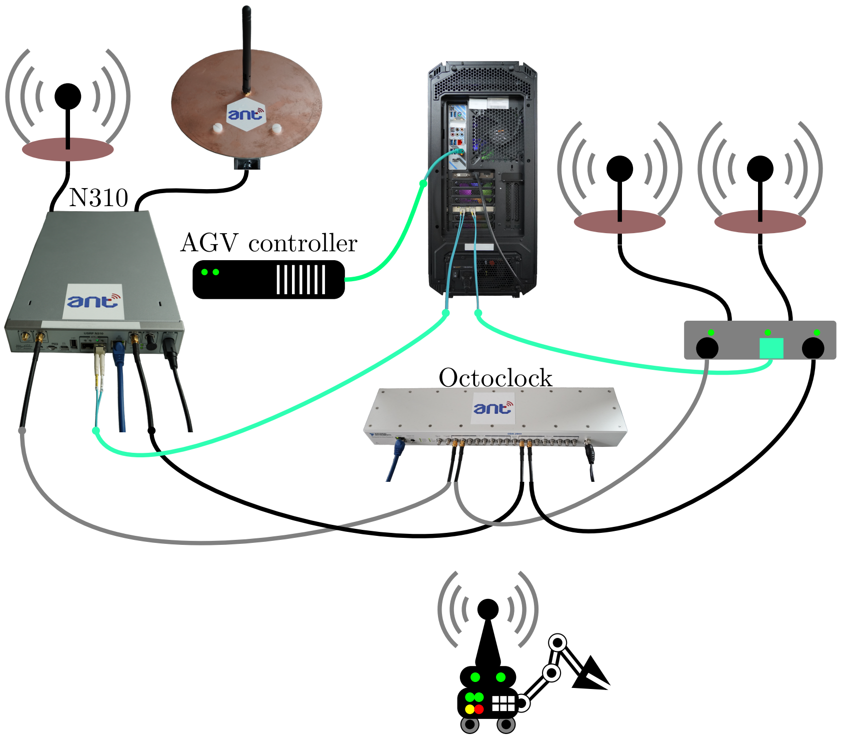


**Figure 8.16:** Cloud RAN hardware and connections with Ettus USRP N310s (N310s) synchronized via an Octoclock-G, antennas, AGV controller, AGV, and a TRX3970X host.

2974)[8] with an integrated computer. However, the embedded x86 hardware only supports a single antenna configuration because of its limited compute power. Thus, we prefer modularity in our setup to effectively upgrade components when and where necessary.

We use Pasternack PE51084[9] rubber duck antennas with a frequency range from 3.3 GHz to 3.8 GHz with a custom-designed ground plane. The AP antennas are mounted at fixed positions in the testbed while the mobile antennas are mounted on top an AGV.

The presented system is measured with `/usr/bin/time` to measure memory usage. Across machines, our measurement tool reports a peak of 220 MB

[8]`https://www.ni.com/en-us/support/model.usrp-2974.html`

[9]`https://www.pasternack.com/rubber-duck-antenna-108mm-sma-male-3300-3800-mhz-2-dbi-pe510`
`aspx`

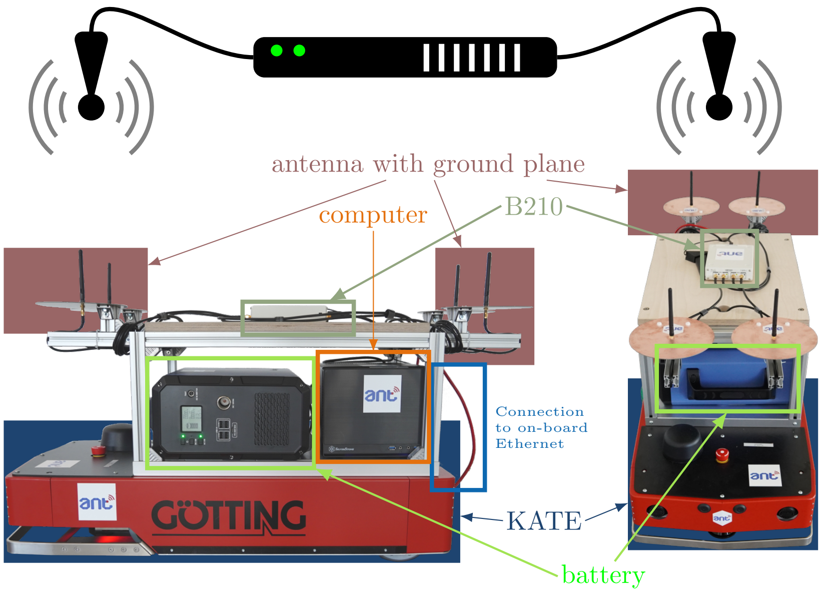


**Figure 8.17:** AGV that consists of a Götting KATE connected to our DSP computer, an additional battery, antennas with custom ground planes, and a B210.

RAM usage even under heavy load in a client configuration. The Cloud RAN configuration measurement results in 222 MB RAM usage, which is slightly higher. It should be noted that the system processes approximately 2 Gbit s$^{-1}$ with a 30.72 MS s$^{-1}$ sample rate and $N_\mathrm{R} = 2$ receive antennas.

## 8.7 Testbed

The software and hardware components are assembled in various experiments to obtain performance data. We start with experiments to compare latency for different system scenarios that include DSP flowgraph latency, multi-threaded GNU Radio flowgraph latency and hardware frontend communication. These tests culminate in OTA experiments with the `ping` utility to determine system latency. Besides, we conduct system throughput experiments. The presented communication system is deployed in testbeds that we discuss and present obtained measurement results to reveal the communication system performance and validate their practicability. Finally, we conduct reliability measurements in these testbeds.

## 8.7.1 Latency

We want to analyze the latencies we measure in the presented system. In contrast to previous chapters, we are interested in the overall latency. However, we split our latency investigations into DSP latencies, multi-threaded DSP latencies, and Round Trip Time (RTT) measurements.

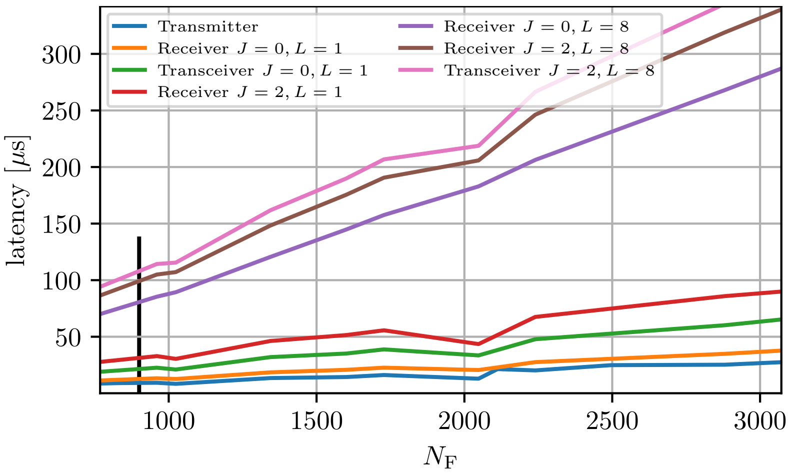


**Figure 8.18:** Aggregate DSP latency for all considered PHY components without synchronization as discussed in Sec. 8.4.4.

Fig. 8.18 shows the aggregated latencies for a transmitter, receiver, and transceiver with increasing frame size $N_F$ and different configurations for Interference-Cancellation (IC) iterations $J$ and polar code list size $L$. The component DSP latencies are investigated in previous chapters, specifically Sec. 3.4, 4.6, and 5.6. The results in Fig. 8.18 exclude synchronization. The black vertical line indicates the chosen parameterization according to Sec. 8.2. The receiver causes the bulk of the measured latency while the transmitter is responsible for a minor latency increase. The largest impact on latency is caused by the polar decoder with a list size $L = 8$, while the number of IC iterations has a lighter impact on latency. Again, we observe a lower latency if $N_F$ is a power of two as discussed in Sec. 5.6. We selected these parameters for this comparison because we identified them to be the ones with the highest impact on latency in previous chapters. We conclude that it is most beneficial to focus on polar Successive Cancellation List (SCL) decoder optimization to reduce the overall DSP latency.

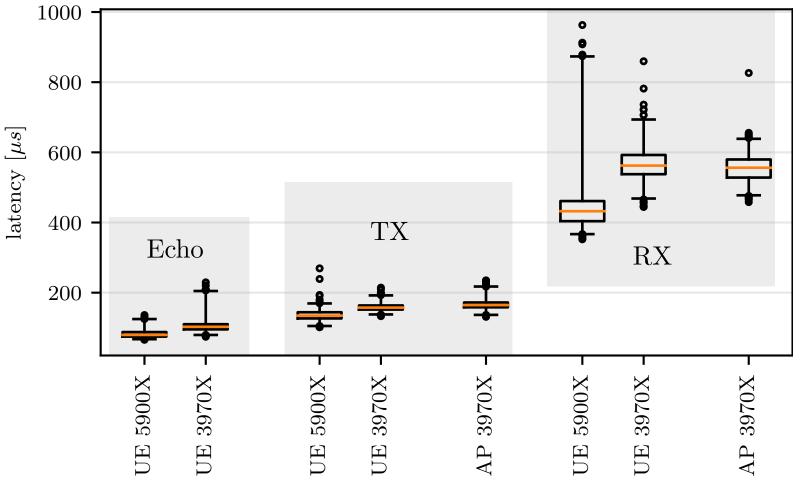


**Figure 8.19:** DSP latencies in GNU Radio on Ryzen 5900X and TRX3970X.

We leverage GNU Radio features to build a full OTA transceiver. Fig. 8.19 shows the measured latencies in a full OTA system in a box plot with $3\sigma$ whiskers. All measurements are obtained by sending a `ping`, or ICMP echo request, from a host in AP configuration to a client in UE configuration. The receivers are configured with $J = 2$ IC iterations and a dynamic polar decoder approach. First, a Successive Cancellation (SC) decoder attempts to decode a frame and only in case this fails, the SCL decoder with $L = 8$ is used. This approach minimizes latency in most cases, and more importantly, makes more resources available for other tasks. The *Echo* entries show the time it takes for a host to respond with an ICMP echo reply, i.e. the duration that starts when an ICMP request leaves our system until a corresponding reply enters our system again. These upper layer processing delays are introduced by the Linux host and differ slightly between the chosen platforms. On a Ryzen 5900X the mean latency is 82.2 µs, while on a TRX3970X the mean is measured at 107.3 µs. This result is an indicator that a higher single core CPU performance is beneficial to lower latency.

The mean transmit DSP latency on a TRX3970X host in UE configuration is 158.3 µs, while the receive DSP latency is 565.3 µs. The AP configuration on a TRX3970X host yields 166.2 µs transmit and 554.1 µs receive latency. The mean latencies on a Ryzen 5900X UE host are 135.2 µs for transmit and 438.6 µs for receive processing. The Ryzen 5900X host exhibits higher latency spread than the other hosts. This DSP latency spread is an indicator to investigate optimizations in multi-thread scheduling.

Overall, the mean one way DSP latency from a TRX3970X AP to a TRX3970X UE is 731.5 µs, and 712.4 µs in the other direction. Similarly, the mean one way DSP latency from a TRX3970X AP to a Ryzen 5900X UE is 604.8 µs, and 689.3 µs in the other direction. Finally, the mean transceiver DSP latencies are 1294.1 µs with a Ryzen 5900X UE and 1443.9 µs with a TRX3970X UE.

These results are surprising at first glance because they are almost an order of magnitude higher than the aggregated DSP benchmarks suggest. However, multiple factors contribute to these results and show a way forward for future latency optimization investigations. We require an efficient multi-threading scheduler and decided to use GNU Radio. The GNU Radio scheduler focuses on throughput in contrast to latency [BMH19, MLM22, LMA+22]. A latency-optimized scheduler requires further knowledge on data and structural dependencies within a flowgraph [BMH19, MLM22]. Optimizing this scheduler is a potential task for future investigations, e.g. by grouping lightweight operations. This approach might minimize the deviations we observe in Fig. 8.19 as well. Further, the results in Fig. 8.18 lack synchronization and transmit control tasks, which are compulsory for OTA transmissions. Also, the interaction to integrate our GNU Radio flowgraph into the host system network layer is not part of our measurements in Fig. 8.18. We consider these investigations to be out of scope for this work and want to point the interested reader towards these topics for future research.

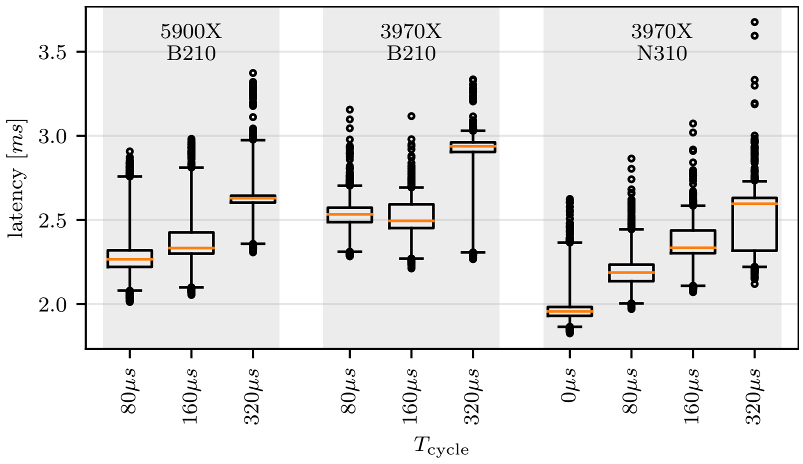


**Figure 8.20:** System RTT on Ryzen 5900X and TRX3970X.

The results in Fig. 8.20 show the application RTTs, or `ping` times. While the AP is always equipped with the same hardware, a TRX3970X CPU with a N310, the UE configurations differ. The N310 requires a $10\,\mathrm{Gbit\,s^{-1}}$ Network Interface Controller (NIC) on the host to sustain the constant data rates, which is only available on our TRX3970X hosts.

Besides the hardware configuration, GNU Radio flowgraph configuration is configured to allow transmission once every $T_{\mathrm{cycle}}$ cycle time. Hence, the system supports the transmission of 8 frames every $T_{\mathrm{cycle}}$. With, $T_{\mathrm{cycle}} = 320\,\mu\mathrm{s}$ cycle time a Ryzen 5900X B210 UE experiences a mean 2.625 ms RTT. Similarly, a TRX3970X B210 system experiences a 2.890 ms RTT, while with a N310 the mean RTT is 2.511 ms. Again, the lower RTT of the Ryzen 5900X system is attributed to the higher single core CPU performance. The B210 UEs reveal a higher RTT, likely because the B210 is connected via USB. In turn, the USB latency is higher than the Ethernet based connection latency, which requires that timed commands over USB receive an additional 520 µs timing advance, while 320 µs timing advance is sufficient for the N310. A lower timing advance results in samples that arrive at the frontend after their indented transmit time and are thus dropped, i.e. not transmitted at all.

Further, we observe some variance in our RTT measurements in Fig. 8.20. We do not conduct any investigation into the root causes of these variances and leave this topic open for future research. However, we can present some conjectures. Since we use a `generic` Linux kernel instead of a `lowlatency` or `realtime` kernel, the variance might be caused or at least boosted by sub-optimal Linux kernel scheduling decisions. Moreover, we conjecture that the block execution order in our flowgraph is re-ordered frequently and causes data stalls. Therefore, a latency optimized scheduler that makes use of additional information on data and structural dependencies might reduce variance as well as latency and RTT [BMH19, MLM22].

Next, we lower the cycle time to $T_{\mathrm{cycle}} = 160\,\mu\mathrm{s}$. The RTTs drop to 2.352 ms for a Ryzen 5900X B210 system, to 2.508 ms for a TRX3970X B210 system, and to 2.361 ms for a TRX3970X N310 system. Thus, the RTTs shrink by 272 µs, 382 µs, and 150 µs respectively. These numbers correspond roughly to the round-trip cycle time reduction. Thus, we conclude that a $T_{\mathrm{cycle}}$ cycle time reduction causes a latency reduction, which is important for communication system design, or configuration, i.e. keep cycle times to a minimum.

One step further, we lower the cycle time to $T_{\mathrm{cycle}} = 80\,\mu\mathrm{s}$. The RTTs drop to 2.274 ms for a Ryzen 5900X B210 system, to 2.528 ms for a TRX3970X B210 system, and to 2.184 ms for a TRX3970X N310 system. In contrast to the other configurations, the TRX3970X N310 system shows the lowest latencies now. The RTTs reduce by 79 µs, −2 µs, and 177 µs respectively. The systems with a B210, and thus a USB connection, are unable to fully leverage the lower cycle time.

For reference, a TRX3970X N310 system that sends packets immediately is able to provide a mean RTT of 1.97 ms, which is another 214 µs latency decrease. However, at this point all measures to enable cooperation with multiple units are disabled and such a system would suffer from packet collisions, i.e. self-interference. The TRX3970X DSP latency measurements reveal a 1443 µs delay and, thus, the mean fronted transceiver delay for a N310 is 527 µs including data transfer between the host and the USRP.

In this section, we demonstrated the capabilities of our SDR communication system solution. It is possible to build a low-latency communication system atop the general purpose GNU Radio framework and we identified areas for further optimizations.

### 8.7.2 Throughput

We focus on reliability and latency in our work. However, in this section we present throughput measurements for several $T_{\text{cycle}}$ configurations for one UE in Fig. 8.21. First off, all measurements appear to be flat lines over the full measurement run because we did not observe any packet errors. The obtained throughput measurements may be considered as an indicator that the presented system is able to provide highly reliable communication services. The $T_{\text{cycle}}$ parameter controls the duration until another packet may be transmitted, i.e. a single UE may only transmit in its designated slot every $T_{\text{cycle}}$.

Given the fixed configuration, these throughput results, 175 kB s$^{-1}$, 116.7 kB s$^{-1}$, and 87.5 kB s$^{-1}$, represent maximum possible throughput for the corresponding cycle times 320 µs, 480 µs, and 640 µs. However, the presented measurements exclude all protocol overhead as discussed in Sec. 8.2. A packet consists of 56 B payload, 8 B UDP header, 20 B IP header, and 15 B PHY header. Thus, a packet consists of 99 B and the corresponding PHY information data rate is 309.4 kB s$^{-1}$, 206.3 kB s$^{-1}$, and 154.7 kB s$^{-1}$ respectively. These numbers indicate that a short packet system would greatly benefit from reduced overhead. However, investigations in that direction are out of scope for this work.

The theoretical throughput of our system with QPSK mapping, $B_{\text{s}} =$ 29.28 MHz bandwidth, and the given GFDM configuration with $K_{\text{on}} = 60$ active subcarriers is 57.6 Mbit s$^{-1}$. However, this value omits any overhead including a CP. Considering whole frames with 99 B and 1136 S per frame, the theoretical throughput may be up to 21.42 Mbit s$^{-1}$. Including UDP,

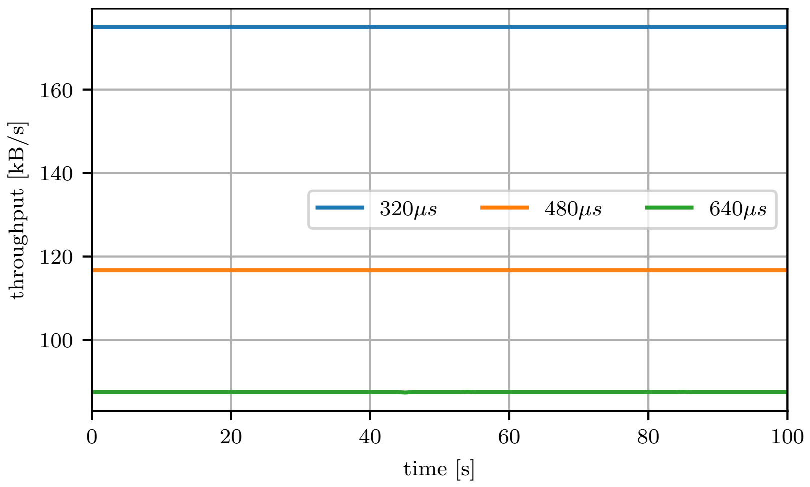


**Figure 8.21:** Throughput measurements for one UE with varying $T_{\text{cycle}}$ in 1 s buckets.

IP, and MAC overhead, the maximum throughput may be 12.11 Mbit s$^{-1}$. Moreover, the system is able to transmit 8 frames every $T_{\text{cycle}} = 320$ µs, 12 frames every $T_{\text{cycle}} = 480$ µs, and 16 frames every $T_{\text{cycle}} = 640$ µs. Thus, the maximum system throughput, or goodput, that is available to an application is 11.2 Mbit s$^{-1}$ to 11.56 Mbit s$^{-1}$, or 1400 kB s$^{-1}$ to 1445.2 kB s$^{-1}$, while the PHY throughput is 19.8 Mbit s$^{-1}$ to 20.44 Mbit s$^{-1}$. The results in Fig. 8.21 show that the system is able to achieve these goodput numbers.

### 8.7.3 Testbed measurements at Bosch Hildesheim

We investigated our communication system solution in a testbed at Bosch Hildesheim during project TACNET 4.0. The testbed serves as a demonstrator to validate that distributed APs improve reliability by leveraging spatial diversity. The gathered SNR measurement data confirms that the chosen approach with a full software implementation and distributed APs is a viable solution for future URLLC systems.

The testbed area, shown in Fig. 8.22, is located inside a factory hall with heavy machinery at Bosch Hildesheim. The AGV tracks cover an area of 6.3 m by 9 m and are surrounded by a metallic truss that carries the APs at 3 m height as indicated. The truss itself is located at a height of 3.35 m and is carried by posts. Two more trusses cross the area above the AGV tracks but are omitted for the sake of clarity. The metallic walls are mobile

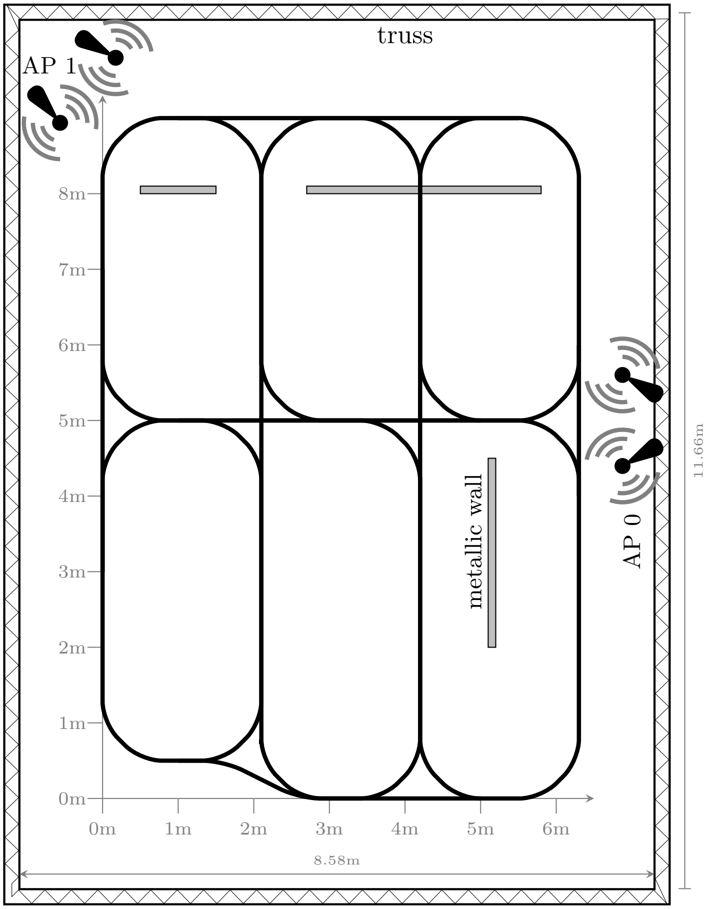


**Figure 8.22:** Testbed area at Bosch Hildesheim with APs positioned at 3 m height and AGV tracks.

and thus, are used to produce different measurement scenarios. We obtain measurements for a *metallic walls* scenario and a *free* scenario where the walls are removed from the testbed.

The APs in Fig. 8.22 consist of one transmit and one receive antenna each and thus constitute a $N_\mathrm{T} = 2$, $N_\mathrm{R} = 2$ setup. Due to issues with frequency synchronization between USRPs at the time, we emulate the two AP configuration by connecting all antennas to one USRP via High Frequency (HF) cables. The long cables from the USRP to AP1 cause approximately 5.22 dB loss that we compensate by adding 5.22 dB in our plots to ensure better comparability. The AGV is equipped with a USRP-2974 with $N_\mathrm{T} = 1$, $N_\mathrm{R} = 1$ and the GNU Radio flowgraph is executed in a Docker container. Due to the limited compute power of the embedded CPU, multiple transmit and receive antennas are unavailable.

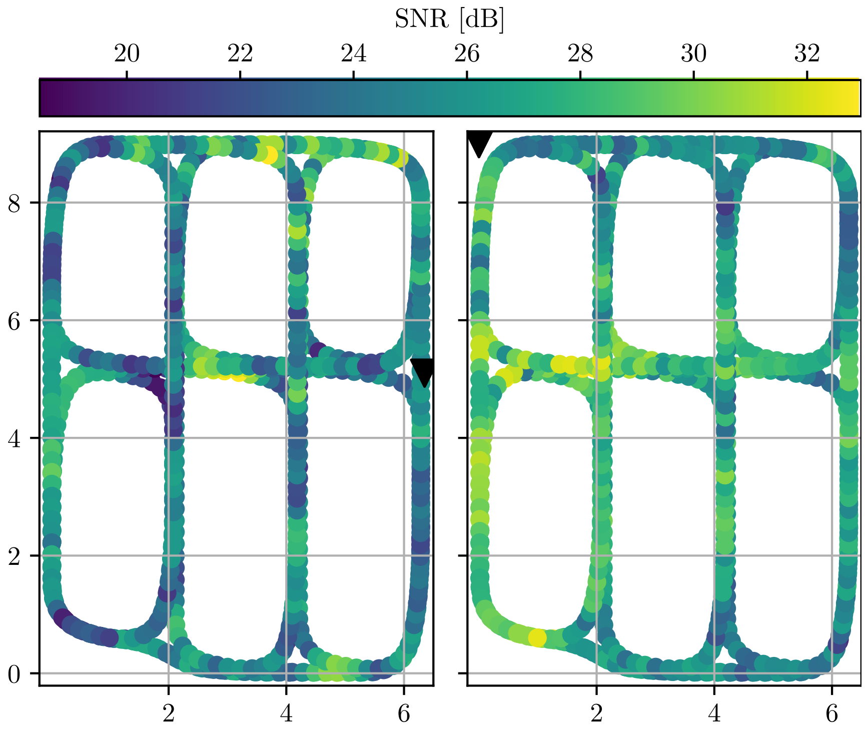


**Figure 8.23:** Testbed area SNR heatmap with scenario *free*. Black triangles indicate AP positions. AP0 results are in the left heatmap and AP1 results on the right. Axes units in m.

We gather SNR measurements while the AGV drives along the presented tracks. The heatmap in Fig. 8.23 presents the measured SNRs for both AP receive streams for the *free* scenario. The lower left corner, near the origin, serves as an example how the system benefits from our distributed approach. The AP0 heatmap indicates low SNR values for this AP near the origin while the AP1 heatmap reveals high SNR values in that area. In other words, the distributed system is able to compensate bad channel conditions on one link with better channel conditions on the other link.

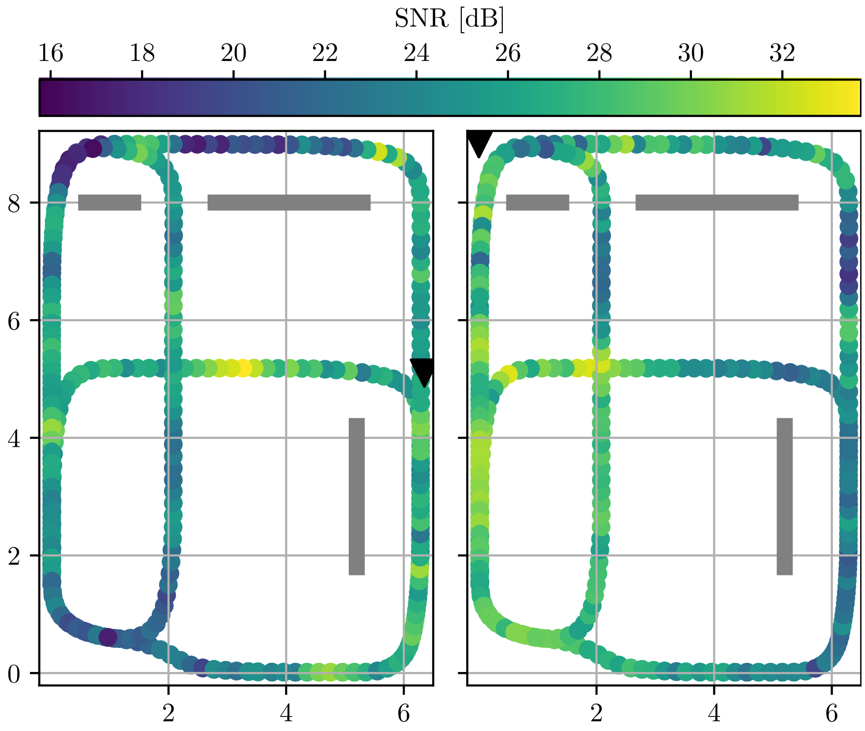


**Figure 8.24:** Testbed area SNR heatmap with scenario *metallic walls*. Black triangles indicate AP positions. Gray bars indicate metallic walls. AP0 results are in the left heatmap and AP1 results on the right. Axes units in m.

The heatmaps in Fig. 8.24 show the result for the *metallic walls* scenario. First, we can observe the influence of the walls on the measured SNR, e.g. by considering the upper left corner of the heatmap for AP0. The area behind the metallic wall from a AP0 point of view exhibits significantly lower SNR values. Further, AP0 is able to serve the AGV on the track to the right with a high SNR while AP1 exhibits better results for the track on the left. These results bolster our conclusion that distributed APs are a strict requirement for ultra reliable communication systems.

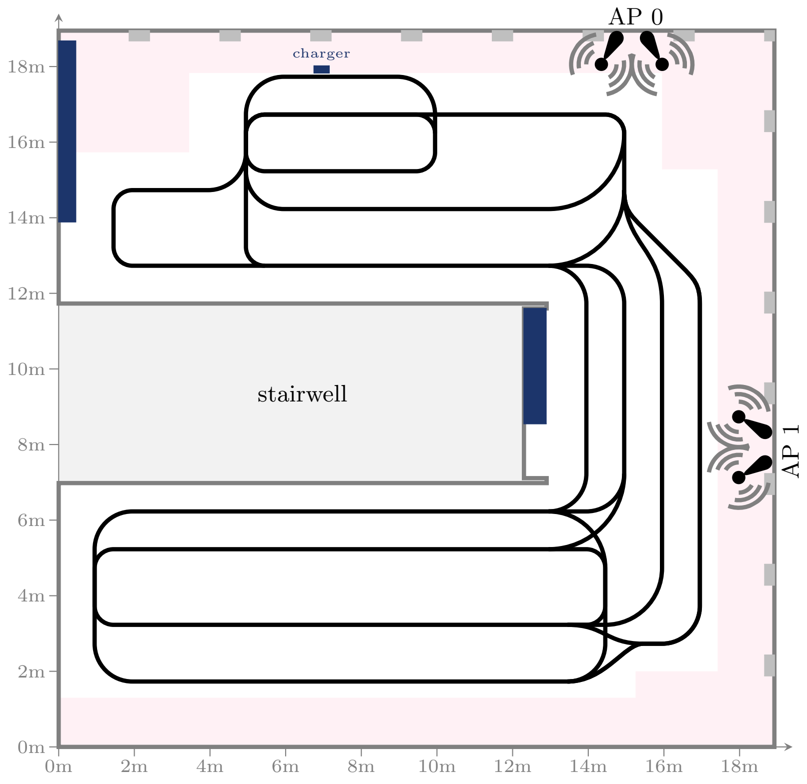


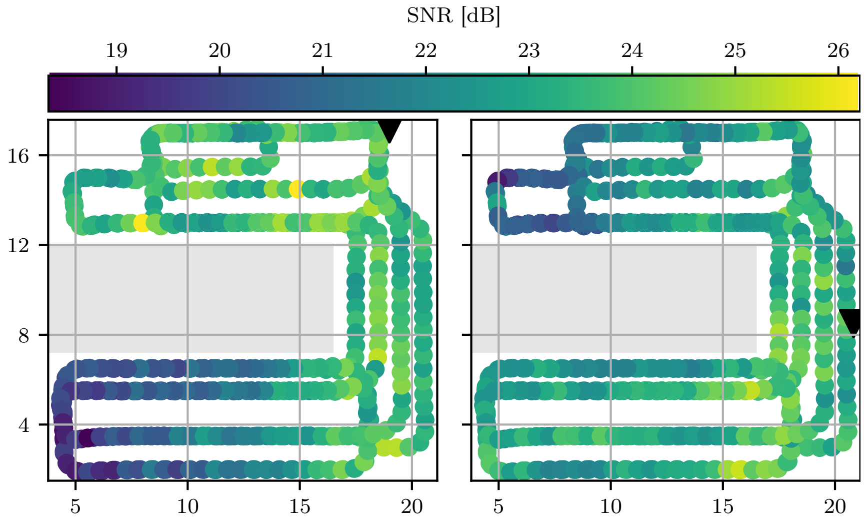
**Figure 8.25:** Testbed area in the NEOS building Bremen with APs positioned at 1.5 m height and AGV tracks. AGVs are restricted from the red areas.

### 8.7.4 Testbed measurements at NEOS

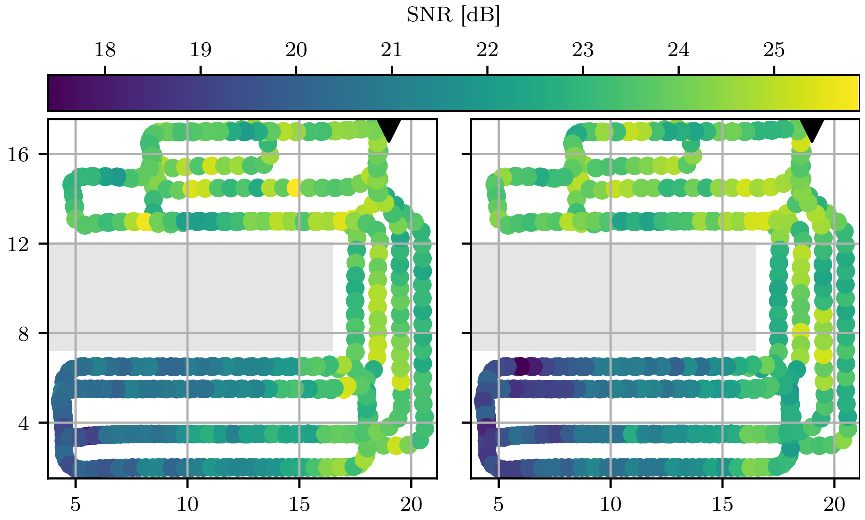
The measurement campaign in this section is an extension to the testbed measurements we present in Sec. 8.7.3. We published preliminary results for this testbed in [DBD23]. We execute multiple experiments to validate the capabilities of our full demonstrator. Most importantly, the testbed area is significantly larger and we are able to use all discussed software components without restrictions. This includes spatially separated and synchronized USRPs that compose a Cloud RAN with distributed APs. Additionally, a wider range of experiments with latency measurements, more SNR measurements, and packet error measurements reveal further insight. Also, the AGV carries our equipment for a fully capable $2 \times 2$ Multiple Input Multiple Output (MIMO) mobile unit. While the testbed is located in an office building, we consider it to be an industrial like environment due to the abundance of reflecting surfaces all around.

The testbed area is illustrated in Fig. 8.25 with a stairwell in the center that introduces NLOS channel conditions from an AP to an AGV in their respective areas. While the testbed area at Bosch Hildesheim covers $100.0\,\mathrm{m}^2$, the testbed area at the NEOS building in Bremen covers $358.5\,\mathrm{m}^2$. Thus, this testbed area is significantly larger. The testbed area in Fig. 8.25 is surrounded by insulated windows on the top, right and bottom side that stretch from the floor to the ceiling, while the building extends to the left but is separated from this area by a wall. The ceiling height is approximately $3.2\,\mathrm{m}$ and the ceiling is covered with a ceiling heating system with metal elements. These physical properties lead to the conclusion that the testbed represents an industrial like environment. AGVs are restricted from the red areas because they contain equipment connection floor boxes with Ethernet and power outlets. Also, the restricted areas are used for other equipment such as APs and antenna racks.

The tracks in Fig. 8.25 indicate possible routes for AGVs where we conduct SNR measurements as presented in Fig. 8.26. Specifically in Fig. 8.26a AP0 experiences good channel conditions with high SNR in the upper area, while the area behind the stairwell suffers from poor SNR values in the bottom half. Similarly, AP1 measures a lower SNR in the top half of the testbed area. In contrast to Fig. 8.26a, the measurements in Fig. 8.26b show the results with a $2 \times 2$ antenna configuration at a single AP0. Both antennas exhibit the same SNR heatmap pattern with the lowest values in the lower left behind the stairwell. As expected, the single AP configuration does not provide spatial diversity to the extend the distributed AP setup does. However, we require further insight into the system to verify that the distributed AP setup provides better resilience against burst errors and thus is better able to support URLLC use cases.

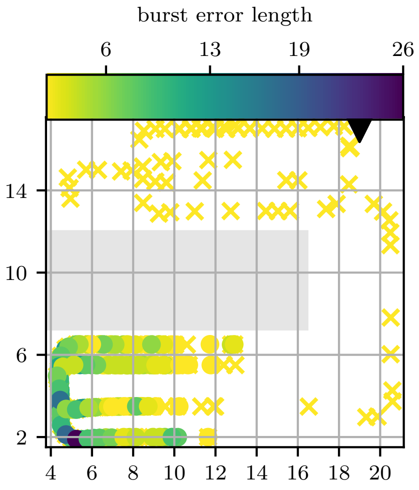


**(a)** Two distributed APs in one Cloud RAN system with a $1 \times 1$ antenna setup each that jointly serve the area.

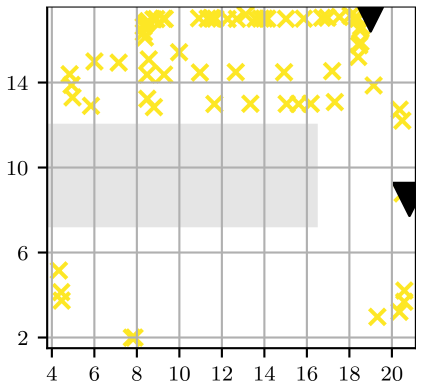


**(b)** One AP with a $2 \times 2$ antenna setup.

**Figure 8.26:** Testbed area SNR heatmaps. Black triangles indicate AP positions. Gray area indicates stairwell position. Axes units in m.



**(a)** One AP with a 2 × 2 antenna setup.



**(b)** Two distributed APs with a 1 × 1 antenna setup each.

**Figure 8.27:** Testbed area burst error length heatmaps. Crosses [×] mark single packet loss positions. Triangles [▼] indicate AP positions. Gray area indicates stairwell position. Axes units in m.

We continue with a lost packet heatmap in Fig. 8.27a with a 2 × 2 single AP configuration. A cross marks a location with a single packet error. In contrast, the color-coded points indicate burst errors. Therefore, the results in Fig. 8.27a make it obvious that a single AP is insufficient to prevent burst errors over the whole demo area.

Finally, we consider a lost packet heatmap in Fig. 8.27b that corresponds to the SNR measurements in Fig. 8.26a. In this case only single packet errors occur. Thus, burst errors are successfully mitigated with a distributed AP setup. Again, we conclude that our Cloud RAN approach with distributed APs is required to enable URLLC.

In order to gain more insight into the latency behavior of our system during an experiment, we periodically use multiple ICMP `ping` tests to measure the RTT of our system as shown in Fig. 8.28. It should be noted that these tests may interfere with the application communication and thus, they are prone to a larger variance. The presented mean, minimum, and maximum graphs show the respective cumulative results over a short time frame. These results are in agreement with the results presented in Fig. 8.20 with a $T_{\text{cycle}} = 320\,\mu\text{s}$ cycle duration that was configured during this measurement as well. Most importantly, our GNU Radio software implementation offers lower latencies than current LTE, and more remarkably 5G NR, systems [SFVS20]. In conjunction with our investigations on reliability in Fig. 8.27, we conclude that our implementation offers the capabilities to support URLLC in an industrial like environment.

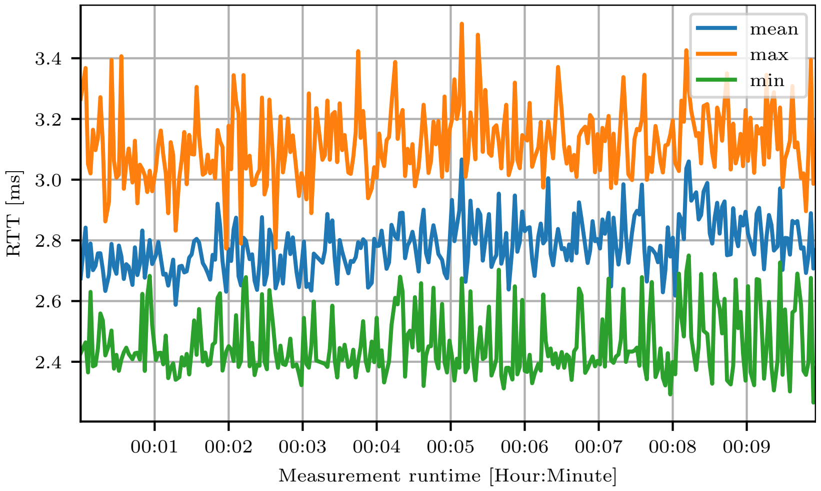


**Figure 8.28:** RTT measurement during a ~10 min SNR measurement run.

## 8.8 Summary

We presented a full communication system demonstrator implemented in software and integrated into GNU Radio. We started this chapter with a discussion of our testbed concept, the software environment, and the used software components along with their configuration. This includes a discussion on our own software as well as important dependencies such as GNU Radio, VOLK, and FFTW. Afterwards, we presented the available hardware and their integration. We conducted multiple experiments to validate system performance regarding reliability and latency constraints. Here, we can conclude that the presented system is able to deliver reliable low latency communication for future URLLC systems with periodic deterministic communication behavior.

### 8.8.1 Contribution

Firstly, we contribute an open-source software communication system implementation in GNU Radio [Dem22a, DRKK22, DG22, Dem22b, Dem22c, DL22]. With distributed APs to improve reliability through spatial diversity, the chosen approach shows a significant improvement to counter fading induced communication outages. Thus, distributed APs provide an important option to minimize burst errors in URLLC communication systems. Our latency measurements confirm that low latency communication systems are achievable and fully implementable in software. The GNU Radio OTA software implementation is able to provide lower latency than current LTE and 5G NR systems [SFVS20]. We show that DSP algorithms alone are only a partial contributor to system latency. The whole system needs to be tightly integrated and optimized in a future work in order to deliver the highest possible performance and consequently lowest latency. Thus, we contribute detailed insight into a full software implementation of a communication system along with valuable pointers where and when to optimize further.

# Chapter 9

# Summary

New use cases for wireless communication that require Ultra Reliable Low Latency Communication (URLLC) systems open a new trove of challenges. In this thesis we address multiple aspects to achieve lower latency and higher reliability. Furthermore, we present a thorough analysis of PHYsical layer (PHY) Digital Signal Processing (DSP) concepts and their impact on latency and reliability. These concepts are used in a Medium-Access-Control (MAC) layer Scheduling and Resource Allocation (S&RA) analysis to find suitable approaches for S&RA that minimize burst errors. Finally, our PHY and MAC analysis results in a demonstrator that verifies our approaches in a testbed. Thus, it is possible to meet the requirements of URLLC systems with a full software Cloud Radio Access Network (Cloud RAN) implementation with distributed Access Points (APs).

## 9.1 Contributions

In Chapter 2 we introduced the scenario we consider and our system model. This resulted in a description for our small and large scale fading channel model with considerations towards industrial channels. Most notably, we consider all fading channels to be Rayleigh distributed.

In Chapter 3, we showed that polar codes are good candidates to meet the requirements of low latency and high reliability for small packets. Based on these findings, we introduced polar codes in-depth with algorithmic improvements towards latency reduction and increased error correction performance. Thereafter, the trade-offs between latency and reliability and the impact of specific parameters were analyzed.

In Chapter 4, we introduced Bit-Interleaved Coded Modulation (BICM) signal processing techniques. The results show that interleaving as well as optimized soft-demappers exhibit very low DSP latency. However, these processing steps may serve as a starting point to better understand how DSP may be optimized in software on a Central Processing Unit (CPU).

In Chapter 5, we introduced Generalized Frequency Division Multiplexing (GFDM) and Orthogonal Frequency Division Multiplexing (OFDM) multicarrier modulation. Here, we devise advantages and disadvantages of these multicarrier modulations. While GFDM is more flexible, exhibits lower Out-Of-Band (OOB) emissions, and allows to shorten frame duration, it is a non-orthogonal modulation and may suffer from self-interference. Further, we demonstrate how Interference-Cancellation (IC) for GFDM demodulation has a large impact on DSP latency. All in all, we devise critical parameters that impact reliability and latency as well as trade-offs based on parameter selection.

In Chapter 6, we show how a Cloud RAN concept with distributed APs and a Functional Split (FS) within the PHY aides to meet reliability requirements via spatial diversity. Moreover, we demonstrate Joint Transmission (JT), and Joint Reception (JR) with Joint Decoding (JD) approaches that greatly improve resilience against deep fades and overall system reliability. In order to ease the fronthaul rate burden with multiple distributed APs, we introduced an offline quantizer design based on the Information Bottleneck Method (IBM). Here, this quantizer approach shows promising results to significantly reduce fronthaul data rates with minimal online computational effort and negligible error correction performance loss.

In Chapter 7, we discussed MAC layer improvements for URLLC systems, based on our Key Performance Indicator (KPI) burst error minimization. First, we identified Mutual Information Effective SNR Mapping (MIESM) as a suitable approach to Link Abstraction (LA). This includes a thorough analysis of derived parameters to optimize accuracy. Afterwards, we investigated S&RA algorithms to combat burst errors. Most importantly, Frequency-Division-Multiplex (FDM) based Resource Allocation (RA) approaches require more accurate Channel State Information (CSI) knowledge while Time-Division-Multiplex (TDM) based approaches perform better with imperfect CSI knowledge. Furthermore, we show that dynamic S&RA strategies are necessary to minimize burst errors unless an abundance of spectrum resources are available. In order to minimize burst errors, these S&RA strategies must take error state knowledge into account to meet URLLC requirements.

In Chapter 8, we fused our gathered software implementation experience into an Over-the-Air (OTA) demonstrator in GNU Radio (GNU Radio). This demonstrator is capable of low latency transmission with high reliability.

Here, we analyzed the impact of a software system on the overall latency in contrast to DSP latencies to identify further areas for future work. Moreover, we analyzed our distributed AP approach to improve reliability. Distributed APs are a feasible and implementable approach to meet URLLC requirements in a practical system.

## 9.2 Open research questions and future work

Naturally, we identified multiple areas for future work with open research questions.

- Besides DSP latencies, system integration latencies were revealed as a major contributor to the overall system latency. These latencies need to be addressed in a latency-optimized multi-threading environment.
- Data exchange between hardware frontends and and hosts that execute a Software-Defined Radio (SDR) system is another major latency contributor. Approaches to reduce these latencies are known in literature [BM20]. However, this is another area that would benefit from further research.
- Overall host Operating System (OS) configuration investigations as well as suitable hardware investigations are an open research topic. Our considerations focus on x86 64bit (x86) hardware but other CPU architectures like ARM (ARM) and RiscV (RiscV) become more prominent and may be better suited in the future.
- It may be beneficial to integrate hardware accelerators, e.g. Graphics Processing Units (GPUs), for specific tasks, e.g. Forward Error Correction (FEC), into an otherwise full software implementation. However, it is imperative to learn more about the latency impact of dedicated accelerators because data transfer latencies may outweigh their benefit. Furthermore, these accelerators need to be evaluated in terms of DSP latencies as well because often times such hardware is optimized for throughput while latency is worse than on a CPU.
- We investigated a set of algorithms for S&RA. However, integration of different optimization targets for different users or services as well as more complicated approaches may be of interest.
- Further polar decoder optimizations both for error correction performance as well as latency are an active area of research.

# Appendix A

# Specialized node implementation in C++

**Listing A.1:** C++ implementation of a Single Parity Check (SPC) node.

```cpp
#include <limits>
#include <vector>
#include <immintrin.h>

static const __m256 SIGN_MASK = _mm256_set1_ps(-0.0f);

inline __m256 _mm256_abs_ps(const __m256 values) {
return _mm256_andnot_ps(SIGN_MASK, values);
}

inline __m256 _mm256_cmplt_ps(const __m256 a, const __m256 b)
    {
return _mm256_cmp_ps(a, b, _CMP_GT_OQ);
}

inline __m256 _mm256_argabsmin_ps(__m256& minindices, const
    __m256 indices,
const __m256 minvalues, const __m256 values) {
const __m256 abs = _mm256_abs_ps(values);
const __m256 mask = _mm256_cmplt_ps(abs, minvalues);
minindices = _mm256_blendv_ps(indices, minindices, mask);
return _mm256_blendv_ps(abs, minvalues, mask);
}

inline unsigned _mm256_argmin_ps(const __m256 x) {
const __m256 fourMin = _mm256_min_ps(x, _mm256_permute2f128_ps
    (x, x, 0b00000001));
const __m256 twoMin = _mm256_min_ps(fourMin, _mm256_permute_ps
    (fourMin, 0b01001110));
const __m256 oneMin = _mm256_min_ps(twoMin, _mm256_permute_ps(
    twoMin, 0b10110001));
const __m256 mask = _mm256_cmp_ps(x, oneMin, _CMP_EQ_OQ);
const int movemask = _mm256_movemask_ps(mask);
const unsigned idx = __tzcnt_u32(movemask);
return idx & 0x7;
}

inline float reduce_xor_ps(__m256 x) {
/* ( x3+x7, x2+x6, x1+x5, x0+x4 ) */
const __m128 x128 =
_mm_xor_ps(_mm256_extractf128_ps(x, 1), _mm256_castps256_ps128
    (x));
/* ( -, -, x1+x3+x5+x7, x0+x2+x4+x6 ) */
const __m128 x64 = _mm_xor_ps(x128, _mm_movehl_ps(x128, x128))
    ;
/* ( -, -, -, x0+x1+x2+x3+x4+x5+x6+x7 ) */
const __m128 x32 = _mm_xor_ps(x64, _mm_shuffle_ps(x64, x64, 0
    x55));
/* Conversion to float is a no-op on x86-64 */
return _mm_cvtss_f32(x32);
}

template<unsigned size>
void calculate_spc(float* bits, const float* llrs) {
__m256 parity = _mm256_setzero_ps();
__m256 minvalues = _mm256_set1_ps(std::numeric_limits<float>::
    max());
__m256 minindices = _mm256_setzero_ps();
__m256 indices = {0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0};
const __m256 step = _mm256_set1_ps(8.0);
for (unsigned i = 0; i < size; i += 8) {
__m256 part = _mm256_loadu_ps(llrs + i);
_mm256_storeu_ps(bits + i, part);
parity = _mm256_xor_ps(parity, part);
minvalues = _mm256_argabsmin_ps(minindices, indices, minvalues
    , part);
indices = _mm256_add_ps(indices, step);
}

const unsigned p = _mm256_argmin_ps(minvalues);
const unsigned minidx = unsigned(minindices[p]);

union { float fParity; unsigned int iParity; };
fParity = reduce_xor_ps(parity);
iParity &= 0x80000000;

reinterpret_cast<unsigned int*>(bits)[minidx] ^= iParity;
}
```

# Appendix B

# Mutual information derivations

Here, we present a more detailed approach to mutual information computations in accordance with [Boc12, Jäc05].

## B.1 Entropy

First, we reiterate over the entropy

$$H_{\mathrm{e}}\left(\mathrm{d}\right) = -E\left\{\log_2 p\left(d\right)\right\} = -\sum_{i=0}^{2^M-1} P\left(d_i\right) \cdot \log_2 P\left(d_i\right) \tag{B.1}$$

with a discrete alphabet $\mathcal{A}_{\mathrm{C}}$ with $|\mathcal{A}_{\mathrm{C}}| = 2^M$. And furthermore, we recall differential entropy

$$h_{\mathrm{e}}\left(\mathrm{d}\right) = -E\left\{\log_2 p_{\mathrm{d}}\left(d\right)\right\} = -\int_{-\infty}^{\infty} p_{\mathrm{d}}\left(\zeta\right) \cdot \log_2 p_{\mathrm{d}}\left(\zeta\right) \mathrm{d}\zeta \tag{B.2}$$

## B.2 Channel capacity

Now, we want to compute channel capacity for a Gaussian alphabet $d \in \mathcal{A}_{\mathrm{C}}$ transmitted over an Additive White Gaussian Noise (AWGN) channel with $\mathrm{n} \sim \mathcal{N}\left(0, \sigma_{\mathrm{n}}^2\right)$ because it serves as an upper bound and thus as our reference.

We can compute mutual information

$$h_{\mathrm{e}}\left(\mathrm{d};\tilde{\mathrm{d}}\right)=E\left\{\log_2\frac{p_{\mathrm{d},\tilde{\mathrm{d}}}\left(d,\tilde{d}\right)}{p_{\mathrm{d}}\left(d\right)\cdot p_{\tilde{\mathrm{d}}}\left(\tilde{d}\right)}\right\}=h_{\mathrm{e}}\left(\tilde{\mathrm{d}}\right)-h_{\mathrm{e}}\left(\mathrm{n}\right)\tag{B.3}$$

for Gaussian variables under the AWGN channel assumption $\tilde{d}=d+n$. If we find the maximum mutual information, we compute channel capacity

$$C=h_{\mathrm{e}}\left(\tilde{\mathrm{d}}\right)-h_{\mathrm{e}}\left(\mathrm{n}\right)=\log_2\left(2\pi\mathrm{e}\sigma_{\mathrm{d}}^2\right)-\log_2\left(2\pi\mathrm{e}\sigma_{\mathrm{n}}^2\right)=\log_2\left(1-\frac{\sigma_{\mathrm{d}}^2}{\sigma_{\mathrm{n}}^2}\right)\tag{B.4}$$

for complex Gaussian alphabets [RU08].

## B.3 Mutual information for BICM

Instead of a theoretical bound, we want to compute mutual information for specific, practical constellations $\mathcal{A}_{\mathrm{C}}$ with $|\mathcal{A}_{\mathrm{C}}|=2^M$, e.g. QPSK or 16QAM. This task is split into computations over bit layers. For one layer, mutual information is calculated as

$$C_l=1-E\left\{\log_2\frac{\sum\limits_{\delta\in\mathcal{A}_{\mathrm{C}}}p_{\tilde{\mathrm{d}}|\mathrm{d}}\left(\tilde{d}|\delta\right)}{\sum\limits_{\varrho\in\mathcal{A}_{\mathrm{C}}{}_l^{(b)}}p_{\tilde{\mathrm{d}}|\mathrm{d}}\left(\tilde{d}|\varrho\right)}\right\}\tag{B.5}$$

where

$$\mathcal{A}_{\mathrm{C}}{}_l^{(b)}=\left\{d_l=\mathcal{M}\left(\boldsymbol{b}_{\mathrm{lb}}|b_{\mathrm{lb},l}=b\right)\right\}\tag{B.6}$$

denotes the set of symbols where the bit pattern at layer $l$ has the value $b$ and

$$p_{\mathrm{y}|\mathrm{x}}\left(y|x\right)=\frac{1}{\pi\sigma_{\mathrm{n}}}e^{-\frac{1}{\sigma_{\mathrm{n}}^2}|y-x|^2},\ x,y\in\mathbb{C}\tag{B.7}$$

denotes the complex conditional probability density function. In (B.5) we substitute $x=\sqrt{p}\alpha$ and $\sigma_{\mathrm{n}}^2=1$ and reformulate the per layer mutual information

$$C_l=1-\frac{1}{2\cdot 2^M\pi}\sum_{b=0}^{1}\sum_{\alpha\in\mathcal{A}_{\mathrm{C}}}\int\limits_{\mathbb{C}}e^{-|\zeta-\sqrt{p}\alpha|^2}\log_2\frac{\sum\limits_{\delta\in\mathcal{A}_{\mathrm{C}}}e^{-|\zeta-\sqrt{p}\delta|^2}}{\sum\limits_{\varrho\in\mathcal{A}_{\mathrm{C}}{}_l^{(b)}}e^{-|\zeta-\sqrt{p}\varrho|^2}}\mathrm{d}\zeta\tag{B.8}$$

to apply the Gauss-Hermite-Quadrature approximation and use symmetry arguments to simplify it. Colors in (B.8) indicate how different parts of the equation are connected. With the Gauss-Hermite-Quadrature

$$y = \int_{-\infty}^{\infty} e^{-v^2} f(v) \mathrm{d}v \approx \sum_{\rho=0}^{N-1} w_\rho f(v_\rho) \tag{B.9}$$

we are able to solve (B.8) numerically. We may compute the weights $W(v_\rho) = w_\rho = e^{-v_\rho^2}$ for an $N$-point Gauss-Hermite-Quadrature or rely on software toolkits such as `numpy.polynomial.hermite.hermgauss` that yield the correct evaluation points $v$. This strategy may be extended to multivariate Gauss Hermite Quadrature to solve this mutual information calculation for complex values [Jäc05, Wil21].

We need to substitute $\Delta_1 = \zeta - \sqrt{p}\alpha$ and thus $\zeta = \Delta_1 + \sqrt{p}\alpha$

$$C_l = 1 - \frac{1}{2 \cdot 2^M \pi} \sum_{b=0}^{1} \sum_{\alpha \in \mathcal{A}_\mathrm{C}} \int_{\mathbb{C}} e^{-|\Delta_1|^2} \log_2 \frac{\sum\limits_{\alpha \in \mathcal{A}_\mathrm{C}} e^{-|\Delta_1 + \overbrace{\sqrt{p}(\alpha - \delta)}^{\Delta_2}|^2}}{\sum\limits_{\varrho \in \mathcal{A}_{\mathrm{C}_l}^{(b)}} e^{-|\Delta_1 + \underbrace{\sqrt{p}(\alpha - \varrho)}_{\Delta_3}|^2}} \mathrm{d}\Delta_1 \tag{B.10}$$

and identify the weights $w = e^{-|\Delta_1|^2}$ and the function

$$f_l^b(\Delta_1) = \log_2 \frac{\sum\limits_{\Delta_2} e^{-|\Delta_1 + \Delta_2|^2}}{\sum\limits_{\Delta_3} e^{-|\Delta_1 + \Delta_3|^2}} = f_l^b(\Delta_1^\mathrm{I}, \Delta_1^\mathrm{Q}) \tag{B.11}$$

and observe that we want to solve a two-dimensional Gauss-Hermite-Quadrature equation with $\Delta_1 = \Delta_1^\mathrm{I} + j\Delta_1^\mathrm{Q}$. We rewrite

$$w = e^{-|\Delta_1|^2} = e^{-\Delta_1^{\mathrm{I}\,2} - \Delta_1^{\mathrm{Q}\,2}} = e^{-\Delta_1^{\mathrm{I}\,2}} \cdot e^{-\Delta_1^{\mathrm{Q}\,2}} = w^\mathrm{I} \cdot w^\mathrm{Q} \tag{B.12}$$

and arrive at our final per layer mutual information equation

$$C_l = 1 - \frac{1}{2 \cdot 2^M \pi} \sum_{b=0}^{1} \sum_{\mathcal{A}_\mathrm{C}} \sum_{\rho} w_\rho^\mathrm{I} \sum_{\tau} w_\tau^\mathrm{Q} f_l^b(\Delta_{1,\rho}^\mathrm{I}, \Delta_{1\ ,\tau}^\mathrm{Q}) \,. \tag{B.13}$$

that we can use to calculate reliabilities for specific constellations that we use. Finally, we can define the mutual information function

$$\mathrm{f}(x) = \sum_{l=0}^{M-1} C_l \tag{B.14}$$

for a constellation $\mathcal{A}_\mathrm{C}$ with $|\mathcal{A}_\mathrm{C}| = 2^M$ and thus $M$ bit layers.

# Appendix C

# Log-likelihood ratio calculation

We need to calculate Log-Likelihood Ratios (LLRs) $\tilde{\boldsymbol{c}}$ from complex received symbols $\tilde{\boldsymbol{d}}$ before decoding to leverage soft decision error correction performance gains [Pro95]. A soft demapper yields soft bit values, or LLRs, whose sign bit indicates the equivalent hard decision bit and the absolute value is a measure of reliability [TB02, ALF04, MAXC16].

This calculation is performed under the assumption of an equivalent AWGN channel with noise power $\sigma_{\mathrm{n}}^2$ on a per symbol basis. We assume transmit symbols $d \in \mathcal{A}_{\mathrm{C}}$, $|\mathcal{A}_{\mathrm{C}}| = 2^M$ from a given constellation, e.g. Quadrature Phase Shift Keying (QPSK) or 16Quadrature Amplitude Modulation (QAM).

## C.1 LLR definition

We recall our LLR definition from (3.4) for Binary Phase Shift Keying (BPSK) or Non Return to Zero (NRZ) mapped bits

$$\begin{aligned} \tilde{c} = \quad \ln \frac{p\left(c=0|\tilde{d}\right)}{p\left(c=1|\tilde{d}\right)} = & \quad \ln \frac{p\left(\tilde{d}|c=0\right)\cdot p\left(c=0\right)}{p\left(\tilde{d}|c=1\right)\cdot p\left(c=1\right)} \\ = & \quad \ln \frac{p\left(\tilde{d}|c=0\right)}{p\left(\tilde{d}|c=1\right)} + \ln \frac{p\left(c=0\right)}{p\left(c=1\right)} \end{aligned} \tag{C.1}$$

and apply Bayes' theorem

$$p\left(c=0|\tilde{d}\right) = \frac{p\left(\tilde{d}|c=0\right) \cdot p\left(c=0\right)}{p\left(\tilde{d}\right)} \tag{C.2}$$

where we note that $p\left(\tilde{d}\right)$ cancels out in (C.1). We note that the a-priori LLR $\ln \frac{p(c=0)}{p(c=1)} = 0$ under the assumption of equiprobable bits. We may either make this assumption because we lack knowledge about the source statistics, or assume that source coding removed all redundancy and thus, all bits should be equiprobable. Thus, we focus on the a-posteriori probabilities $p\left(\tilde{d}|c\right)$ now.

## C.2 Likelihood

We compute a likelihood

$$p\left(\tilde{d}|d\right) = \frac{1}{\pi\sigma_{\mathrm{n}}^2} \cdot e^{-\frac{|\tilde{d}-d|^2}{\sigma_{\mathrm{n}}^2}} \quad d \in \mathcal{A}_{\mathrm{C}} \tag{C.3}$$

for a transmit symbol $d$ when $\tilde{d}$ is received under our AWGN assumption. We move to the ln domain

$$\ln p\left(\tilde{d}|d_m\right) = -\frac{|\tilde{d}-d_m|^2}{\sigma_{\mathrm{n}}^2} - \ln\frac{1}{\pi\sigma_{\mathrm{n}}^2} \ . \tag{C.4}$$

and note that this computation simplifies to computing the euclidean distance between a complex received symbol $\tilde{d}$ and an assumed transmit symbol $d$ and multiplication with a scaling factor.

## C.3 Calculation

With all necessary prerequisites in place, we can now compute a LLR

$$\tilde{c} = \ln\frac{p\left(c=0|\tilde{d}\right)}{p\left(c=1|\tilde{d}\right)} = \ln\frac{p\left(d_1|\tilde{d}\right)}{p\left(d_0|\tilde{d}\right)} = \frac{|\tilde{d}-d_1|^2}{\sigma_{\mathrm{n}}^2} - \frac{|\tilde{d}-d_0|^2}{\sigma_{\mathrm{n}}^2} \tag{C.5}$$

for our BPSK case. Finally, we want to extend this calculation to constellations of size $M$. We calculate an LLR

$$\tilde{c}_l = \ln \frac{p\left(c_l = 0|\tilde{d}\right)}{p\left(c_l = 1|\tilde{d}\right)} = \ln \frac{\sum\limits_{\forall d \in \mathcal{A}_{\mathrm{C}}{}_l^{(0)}} p\left(\tilde{d}|d\right)}{\sum\limits_{\forall d \in \mathcal{A}_{\mathrm{C}}{}_l^{(1)}} p\left(\tilde{d}|d\right)} \tag{C.6}$$

for layer $l$ by summing up the probabilities of $d \in \mathcal{A}_{\mathrm{C}}{}_l^{(c)}$. The set $\mathcal{A}_{\mathrm{C}}{}_l^{(c)} \subset \mathcal{A}_{\mathrm{C}}$ contains all symbols whose labels hold $c$, 0 or 1, at layer $l$. In order to carry out all operations in the ln domain, we employ the LogSumExp (LSE) approximation

$$\ln \sum_j e^{z_j} \approx \max_j z_j \tag{C.7}$$

[TB02]. Under the AWGN assumption and (C.3) together with the approximation (C.7), we approximate (C.6) and arrive at a computationally more lightweight LLR calculation

$$\begin{aligned} \tilde{c}_l = &\ \ln \frac{\sum\limits_{\forall d \in \mathcal{A}_{\mathrm{C}}{}_l^{(0)}} p\left(\tilde{d}|d\right)}{\sum\limits_{\forall d \in \mathcal{A}_{\mathrm{C}}{}_l^{(1)}} p\left(\tilde{d}|d\right)} \\ \approx &\ \frac{1}{\sigma_{\mathrm{n}}^2} \left( \min_{d \in \mathcal{A}_{\mathrm{C}}{}_l^{(0)}} |\tilde{d} - d|^2 - \min_{d \in \mathcal{A}_{\mathrm{C}}{}_l^{(1)}} |\tilde{d} - d|^2 \right) \end{aligned} \tag{C.8}$$

that we use for general constellations.

## C.4 Specialized calculations

Specialized demappers might help tremendously to improve soft demapper latency and throughput while the impact on error rate performance is negligible [TB02, ALF04, MAXC16]. Specialization, and thus optimization, is executed under the assumption of fixed constellations [ETS18a] and an independent and identically distributed (i.i.d.) assumption for inphase and quadrature components. Here we present the formulas for BPSK, QPSK, 16QAM, 64QAM, and 256QAM.

**BPSK**

$$\tilde{c}_0 = \frac{2}{\sigma_{\mathrm{n}}^2} \mathfrak{Re}\left\{\tilde{d}\right\} \tag{C.9}$$

**QPSK**

$$\begin{aligned}\tilde{c}_0 &= \frac{\sqrt{2}}{\sigma_\mathrm{n}^2}\mathfrak{Re}\left\{\tilde{d}\right\} \\ \tilde{c}_1 &= \frac{\sqrt{2}}{\sigma_\mathrm{n}^2}\mathfrak{Im}\left\{\tilde{d}\right\}\end{aligned} \tag{C.10}$$

**16QAM**

$$\begin{aligned}\tilde{c}_0 &= \frac{2}{\sqrt{10}\cdot\sigma_\mathrm{n}^2}\mathfrak{Re}\left\{\tilde{d}\right\} \\ \tilde{c}_1 &= \frac{2}{\sqrt{10}\cdot\sigma_\mathrm{n}^2}\mathfrak{Im}\left\{\tilde{d}\right\} \\ \tilde{c}_2 &= \frac{2}{\sqrt{10}\cdot\sigma_\mathrm{n}^2}\left(\frac{2}{\sqrt{10}} - \left|\mathfrak{Re}\left\{\tilde{d}\right\}\right|\right) \\ \tilde{c}_3 &= \frac{2}{\sqrt{10}\cdot\sigma_\mathrm{n}^2}\left(\frac{2}{\sqrt{10}} - \left|\mathfrak{Im}\left\{\tilde{d}\right\}\right|\right)\end{aligned} \tag{C.11}$$

**64QAM**

$$\begin{aligned}\tilde{c}_0 &= \frac{2}{\sqrt{42}\cdot\sigma_\mathrm{n}^2}\mathfrak{Re}\left\{\tilde{d}\right\} \\ \tilde{c}_1 &= \frac{2}{\sqrt{42}\cdot\sigma_\mathrm{n}^2}\mathfrak{Im}\left\{\tilde{d}\right\} \\ \tilde{c}_2 &= \frac{2}{\sqrt{42}\cdot\sigma_\mathrm{n}^2}\left(\frac{4}{\sqrt{42}} - \left|\mathfrak{Re}\left\{\tilde{d}\right\}\right|\right) \\ \tilde{c}_3 &= \frac{2}{\sqrt{42}\cdot\sigma_\mathrm{n}^2}\left(\frac{4}{\sqrt{42}} - \left|\mathfrak{Im}\left\{\tilde{d}\right\}\right|\right) \\ \tilde{c}_4 &= \frac{2}{\sqrt{42}\cdot\sigma_\mathrm{n}^2}\left(\frac{2}{\sqrt{42}} - \left|\left|\mathfrak{Re}\left\{\tilde{d}\right\}\right| - \frac{4}{\sqrt{42}}\right|\right) \\ \tilde{c}_5 &= \frac{2}{\sqrt{42}\cdot\sigma_\mathrm{n}^2}\left(\frac{2}{\sqrt{42}} - \left|\left|\mathfrak{Im}\left\{\tilde{d}\right\}\right| - \frac{4}{\sqrt{42}}\right|\right)\end{aligned} \tag{C.12}$$

**256QAM**

$$\begin{aligned}
\tilde{c}_0 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\Re\mathfrak{e}\left\{\tilde{d}\right\}\\
\tilde{c}_1 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\Im\mathfrak{m}\left\{\tilde{d}\right\}\\
\tilde{c}_2 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\left(\frac{8}{\sqrt{170}}-\left|\Re\mathfrak{e}\left\{\tilde{d}\right\}\right|\right)\\
\tilde{c}_3 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\left(\frac{8}{\sqrt{170}}-\left|\Im\mathfrak{m}\left\{\tilde{d}\right\}\right|\right)\\
\tilde{c}_4 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\left(\frac{4}{\sqrt{170}}-\left|\left|\Re\mathfrak{e}\left\{\tilde{d}\right\}\right|-\frac{8}{\sqrt{170}}\right|\right)\\
\tilde{c}_5 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\left(\frac{4}{\sqrt{170}}-\left|\left|\Im\mathfrak{m}\left\{\tilde{d}\right\}\right|-\frac{8}{\sqrt{170}}\right|\right)\\
\tilde{c}_6 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\left(\frac{2}{\sqrt{170}}-\left|\left|\left|\Re\mathfrak{e}\left\{\tilde{d}\right\}\right|-\frac{8}{\sqrt{170}}\right|-\frac{4}{\sqrt{170}}\right|\right)\\
\tilde{c}_7 &= \frac{2}{\sqrt{170}\cdot\sigma_{\mathrm{n}}^2}\left(\frac{2}{\sqrt{170}}-\left|\left|\left|\Im\mathfrak{m}\left\{\tilde{d}\right\}\right|-\frac{8}{\sqrt{170}}\right|-\frac{4}{\sqrt{170}}\right|\right)
\end{aligned} \tag{C.13}$$

# Acronyms

**3GPP** 3rd Generation Partnership Project. 205

**4G** 4th Generation. 23, 66, 205

**5G** 5th Generation. 43, 46, 47, 205

**5G NR** 5th Generation New Radio. 2, 5, 6, 17, 18, 23, 43, 53, 55, 57, 65, 66, 118, 136, 148, 151, 157, 184, 185, 205

**ADC** Analog-to-Digital Converter. 96, 205

**AES** Advanced Encryption Standard. 40, 205

**AESM** Average Effective SNR Mapping. 116, 122, 124, 126, 132, 144, 145, 205

**AGC** Automatic-Gain-Control. 205

**AGV** Automated Guided Vehicle. 9–11, 19, 94, 148, 159, 168–170, 176–178, 180, 181, 205

**AMC** Adaptive Modulation and Coding. 117, 205

**AMD** Advanced Micro Devices. 168, 205

**AP** Access Point. 3, 5, 7, 94, 113, 141, 142, 145, 147–149, 168, 169, 172, 173, 176–183, 185, 187–189, 205

**API** Application Programming Interface. 4, 53, 156, 160, 164–167, 205

**ARM** ARM. 156, 168, 189, 205

**ASIC** Application Specific Integrated Circuit. 4, 152, 205

**ASK** Amplitude Shift Keying. 71, 101, 205

**AVX** Advanced Vector Extensions. 34, 35, 156, 168, 205

**AWGN** Additive White Gaussian Noise. 13, 14, 18, 19, 21, 23, 28, 29, 57, 61, 78, 88, 101, 102, 106–108, 110, 112, 121–125, 129, 133, 195, 196, 199–201, 205

**B210** Ettus USRP B210. 151, 168, 170, 174, 205

**BB** Bhattacharyya Bound. 29, 30, 43, 44, 46, 47, 49, 50, 205

**BE** $\beta$-Expansion. 30, 43, 46, 47, 126, 205

**BEC** Binary Erasure Channel. 28, 29, 205

**BER** Bit-Error-Rate. 28, 49, 205

**BICM** Bit-Interleaved Coded Modulation. 6, 12, 53, 57–59, 62, 116, 165, 188, 205

**BLT** Balian-Low Theorem. 205

**BP** Belief Propagation. 31, 205

**BPSK** Binary Phase Shift Keying. 31, 56, 199, 201, 205

**BSC** Binary Symmetric Channel. 28, 205

**CA** Channel Aware. 131, 132, 137, 205

**CA-SCL** CRC-Aided Successive Cancellation List. 23–25, 31, 38, 40, 42, 44, 45, 48, 52, 205

**CADS** Channel Aware Delay Sensitive. 131, 132, 137–144, 205

**CCDF** Complementary Cumulative Distribution Function. 136, 205

**CCSDS** Consultative Committee for Space Data Systems. 23, 24, 205

**CDD** Cyclic Delay Diversity. 88, 93, 94, 98, 113, 205

**CDF** Cumulative Distribution Function. 205

**CEP** Consecutive Erroneous Packet. 2, 3, 205

**CFO** Carrier-Frequency-Offset. 163, 205

**CLC** Closed-Loop-Control. 2, 115, 120, 205

**cloud platform** cloud platform. 7, 93–97, 99–101, 205

**Cloud RAN** Cloud Radio Access Network. 2–5, 7–9, 13, 22, 24, 53, 54, 63, 67, 93–97, 99, 105–108, 110, 112, 113, 147, 149, 152, 154, 159, 168, 169, 181–183, 187, 188, 205

**CMAC** Cipher-based Message Authentification Code. 21, 40, 48, 49, 205

**CNR** Carrier-to-Noise-Ratio. 17, 61, 78, 89, 101, 108, 109, 118, 122, 126, 132, 162, 163, 165, 166, 205

**Core i7-4770S** Intel Core i7-4770S. 42, 205

**COTS** Commercial-Of-The-Shelf. 3, 4, 67, 205

**CP** Cyclic Prefix. 16, 66, 67, 69, 72, 76–79, 87, 88, 98, 151, 163, 165, 176, 205

**CPU** Central Processing Unit. 7, 59–61, 63, 151, 155, 157, 158, 168, 172–174, 177, 188, 189, 205

**CRC** Cyclic Redundancy Check. 11, 22, 25, 26, 38–42, 44, 48–50, 52, 99, 150, 205

**CS** Cyclic Suffix. 66, 77–79, 87, 151, 163, 165, 205

**CSI** Channel State Information. 12, 118–121, 126, 131–134, 138–140, 142–145, 188, 205

**CU** Centralized Unit. 95, 96, 205

**DAB** Digital Audio Broadcasting. 66, 205

**DC** Direct Current. 68, 69, 205

**DE** Density Evolution. 29, 43, 46, 205

**DFT** Discrete Fourier Transform. 16, 17, 70, 74, 76, 77, 205

**DMC** Discrete Memoryless Channel. 24, 28, 205

**DPDK** Data Plane Development Kit. 157, 205

**DQ** Deterministic Quantizer. 101, 205

**DREP** TypeI Double Repetition. 37, 205

**DS** Delay Sensitive. 131, 132, 137, 205

**dSNR** Design SNR. 30, 47–50, 205

**DSP** Digital Signal Processing. 5, 8, 65, 66, 90, 91, 95–97, 100, 148, 149, 156, 158, 159, 162, 163, 165, 166, 170–173, 175, 185, 187–189, 205

**DSPC** TypeIII Double SPC. 37, 205

**DU** Distributed Unit. 95, 96, 205

**DVB** Digital Video Broadcasting. 66, 205

**EESM** Exponential Effective SNR Mapping. 116, 122, 123, 125–127, 129, 130, 144, 145, 205

**eMBB** enhanced Mobile Broadband. 1, 2, 205

**ESM** Effective SNR Mapping. 116, 122–124, 126, 205

**Ethernet** IEEE 802.3 Ethernet. 157, 168, 174, 181, 205

**FA** Factory Automation. 1, 39, 205

**FAR** False Alarm Rate. 39, 205

**Fast-SSC** Fast Simplified Successive Cancellation. 25, 26, 36, 37, 205

**FBMC** Filter-Bank Multi-Carrier. 66, 205

**FDD** Frequency-Division-Duplex. 18, 118, 205

**FDE** Frequency-Domain Equalization. 70, 75, 77, 205

**FDM** Frequency-Division-Multiplex. 119, 134–140, 142, 145, 188, 205

**FEC** Forward Error Correction. 4, 6, 12, 22, 23, 40, 41, 52–54, 56, 57, 66, 92, 93, 97, 151, 162, 166, 189, 205

**FER** Frame-Error-Rate. 3, 6, 24, 28, 30, 49, 52, 62, 86, 93, 94, 100, 106–113, 115, 116, 121–124, 129, 133, 135, 137, 138, 145, 205

**FFT** Fast Fourier Transform. 27, 66, 70, 72, 74, 75, 79, 81, 82, 84, 90, 91, 205

**FFTW** Fastest Fourier Transform in The West. 81, 82, 156, 164, 184, 205

**FOU** Foo over UDP. 155, 160, 205

**fp-32** floating point. 34, 44, 48, 50, 51, 94, 106–109, 111, 113, 151, 205

**FPGA** Field Programmable Gate Array. 4, 67, 152, 205

**Frozen-QUP** Frozen Quasi Uniform Puncturing. 39, 51, 52, 205

**FS** Functional Split. 7, 93, 94, 96, 97, 100, 105, 107, 108, 110, 112, 113, 188, 205

**GA** Gaussian Approximation. 29, 30, 43, 46, 205

**GCC** GNU Compiler Collection. 59, 205

**GFDM** Generalized Frequency Division Multiplexing. 7, 12, 16, 65–67, 69–75, 77–81, 83–87, 90–92, 105–112, 149, 151, 162, 164, 165, 176, 188, 205

**GNU Radio** GNU Radio. 5, 6, 8, 53, 63, 65, 147–150, 153–160, 163–167, 170, 171, 173–175, 177, 183–185, 188, 205

**GPLv3+** GNU General Public License v3.0 or later. 156, 164, 205

**GPP** General Purpose Processor. 3, 91, 96, 154, 156, 157, 168, 205

**GPU** Graphics Processing Unit. 4, 168, 189, 205

**GRC** GNU Radio Companion. 159, 205

**GUI** Graphical User Interface. 159, 205

**Götting KATE** Götting KG Kinetic Automat for Transport Enhancement. 148, 168, 170, 205

**HDL** Hardware Description Language. 205

**HF** High Frequency. 177, 205

**HiFlecs** Innovative Wireless Technologies for Industrial Automation. 2, 147, 205

**high-PHY** high-PHYsical layer. 97, 99, 100, 205

**HMAC** Keyed-hash Message Authentification Code. 22, 40, 205

**i.i.d.** independent and identically distributed. 14, 57, 58, 100, 101, 110, 112, 118, 201, 205

**I4.0** Industry 4.0. 2, 54, 65–67, 79, 91, 93, 113, 118, 120, 205

**I&Q** Inphase and Quadrature component. 94, 97, 205

**IB** Information Bottleneck. 205

**IBM** Information Bottleneck Method. 7, 94, 97, 100, 101, 105–113, 188, 205

**IC** Interference-Cancellation. 66, 75, 76, 85–87, 92, 171, 172, 188, 205

**ICI** Inter-Carrier Interference. 75, 205

**ICMP** Internet Control Message Protocol. 150, 172, 183, 205

**ID** IDentifier. 150, 160, 205

**int8** 8bit integer. 50, 51, 151, 205

**Intel X710** Intel Ethernet Converged Network Adapter X710-DA2. 168, 205

**intra-PHY** intra-PHYsical layer. 93, 97, 99, 205

**IoT** Internet of Things. 1, 2, 205

**IP** Internet Protocol. 150, 155, 176, 205

**IRLG** Industrial Radio Lab Germany. 147, 205

**ISA** Instruction Set Architecture. 168, 205

**ISI** Inter-Symbol Interference. 67, 69, 77, 205

**JD** Joint Decoding. 94, 97–100, 110, 112, 113, 188, 205

**JES1500** Jauch Quartz Power Station JES1500WHA. 168, 205

**JR** Joint Reception. 93, 94, 96, 99–101, 105, 113, 188, 205

**JT** Joint Transmission. 88, 93, 94, 96, 98, 105, 112, 113, 188, 205

**KPI** Key Performance Indicator. 2, 3, 7, 115, 116, 188, 205

**LA** Link Abstraction. 5, 7, 57, 116–118, 120–123, 125, 126, 129, 133, 135, 144, 145, 188, 205

**LDPC** Low Density Parity Check. 22–24, 205

**LLR** Log-Likelihood Ratio. 7, 17, 26, 31, 32, 36–38, 53, 54, 56, 61, 78, 89, 97, 100, 103–106, 109, 154, 158, 162, 166, 199–201, 205

**LLRq** LLR quantization. 105–109, 205

**LO** Local Oscillator. 69, 205

**LOS** Line-Of-Sight. 18, 187, 205

**low-PHY** low-PHYsical layer. 97, 99, 100, 205

**LSB** Least Significant Bit. 60, 205

**LSE** LogSumExp. 122, 201, 205

**LTE** Long Term Evolution. 2, 5, 23–25, 53, 55, 65, 66, 148, 157, 184, 185, 205

**LUT** Look-Up-Table. 49, 56, 97, 100, 101, 104, 205

**M2M** Machine to Machine. 21, 205

**MAC** Medium-Access-Control. 2, 3, 7, 9, 40, 97, 115, 116, 144, 150, 151, 160, 176, 187, 188, 205

**MACode** Message Authentification Code. 11, 22, 40, 41, 52, 205

**MC** Motion Control. 1, 120, 205

**MCS** Modulation and Coding Scheme. 116–118, 121–123, 125, 128–130, 132–136, 139, 144, 145, 151, 205

**MF** Matched-Filter. 70, 71, 75, 205

**MIESM** Mutual Information Effective SNR Mapping. 116, 122, 123, 125–127, 129, 130, 144, 145, 188, 205

**MIMO** Multiple Input Multiple Output. 70, 181, 205

**ML** Maximum Likelihood. 38, 205

**MMSE** Minimum Mean Square Error. 70, 71, 78, 126, 205

**mMTC** massive Machine Type Communication. 1, 2, 205

**MSB** Most Significant Bit. 34, 205

**MSE** Mean Square Error. 116, 123, 124, 205

**MTC** Machine-type-Communication. 205

**MTTF** Mean Time To Failure. 2, 115, 117, 205

**MTU** Maximum Transmission Unit. 150, 205

**N310** Ettus USRP N310. 168, 169, 173–175, 205

**NEON** NEON. 156, 168, 205

**NIC** Network Interface Controller. 174, 205

**NIST** National Institute of Standards and Technology. 94, 205

**NLOS** Non-Line-Of-Sight. 18, 19, 136, 181, 205

**NR** New Radio. 205

**NRZ** Non Return to Zero. 14, 56, 199, 205

**O-RAN** O-RAN Alliance. 4, 205

**OAI** OpenAirInterface. 5, 205

**Octoclock-G** Ettus Octoclock-G. 168, 169, 205

**OFDM** Orthogonal Frequency Division Multiplexing. 7, 12, 16, 65–73, 75, 77, 78, 80, 81, 86, 88, 90, 91, 105–108, 116, 118, 121, 124, 126, 165, 188, 205

**OO** Object Oriented. 154, 205

**OOB** Out-Of-Band. 66, 67, 73, 74, 77, 91, 188, 205

**OOT** Out-Of-Tree. 63, 147, 155, 156, 159, 165–167, 205

**OS** Operating System. 155, 167, 189, 205

**OTA** Over-the-Air. 6, 8, 78, 147–150, 153, 158, 159, 164, 170, 171, 173, 185, 188, 205

**PDP** Power Delay Profile. 15, 205

**PDU** Protocol Data Unit. 160, 162, 205

**PHY** PHYsical layer. 2, 5, 7, 9, 11, 96, 97, 112, 116, 117, 151, 160, 162, 171, 176, 187, 188, 205

**PSD** Power Spectral Density. 73, 205

**PSK** Phase Shift Keying. 71, 205

**QAM** Quadrature Amplitude Modulation. 12, 54–56, 58, 59, 71, 102, 121, 127, 130, 166, 196, 199, 201–203, 205

**QoS** Quality-of-Service. 115, 117, 145, 205

**QPSK** Quadrature Phase Shift Keying. 55, 56, 100, 102, 104–107, 109, 111, 112, 124, 130, 134, 151, 166, 176, 196, 199, 201, 202, 205

**QUP** Quasi Uniform Puncturing. 39, 205

**RA** Resource Allocation. 116, 117, 119, 120, 130–135, 137–145, 188, 205

**Radeon RX 550** AMD Radeon RX 550. 168, 205

**RAM** Random Access Memory. 168, 169, 205

**RAN** Radio Access Network. 3–5, 11, 93–96, 148, 152, 205

**RAP** Radio Access Point. 9, 93, 94, 96–101, 105, 108, 110–113, 149, 205

**RC** Raised-Cosine. 77, 205

**REP** Repetition. 36, 47, 205

**REP-One** TypeIV Repetion One. 37, 205

**RF** Radio Frequency. 96, 157, 205

**RiscV** RiscV. 168, 189, 205

**RR** Round Robin. 131, 132, 137, 205

**RRC** Root-Raised-Cosine. 72, 75, 205

**RRH** Remote Radio Head. 96, 205

**RRU** Remote Radio Unit. 96, 205

**RTT** Round Trip Time. 171–175, 183, 184, 205

**RU** Radio Unit. 95, 96, 205

**Ryzen 5900X** AMD Ryzen 9 5900X. 168, 172–174, 205

**S&RA** Scheduling and Resource Allocation. 5, 7, 17, 116–121, 123, 125–128, 130, 144, 145, 187–189, 205

**SC** Successive Cancellation. 23, 24, 29, 31, 33, 35, 37, 38, 42, 44, 45, 50, 172, 205

**SCAN** Soft CANcellation. 25, 31, 37, 205

**SCFlip** Successive Cancellation Flip. 25, 31, 37, 205

**Schmidl&Cox** Schmidl&Cox. 78, 163, 205

**SCL** Successive Cancellation List. 25, 31, 37, 38, 42, 44, 171, 172, 205

**SCS** Successive Cancellation Stack. 31, 205

**SDQ** Sequential Deterministic Quantizer. 101, 205

**SDR** Software-Defined Radio. 4, 5, 10, 13, 41, 42, 53, 54, 59, 63, 65, 67, 91, 95, 148, 155, 157, 175, 189, 205

**SER** Symbol-Error-Rate. 86, 205

**SIMD** Single-Instruction-Multiple-Data. 33, 34, 49, 156, 168, 205

**SINR** Signal-to-Interference-plus-Noise-Ratio. 205

**SNR** Signal-to-Noise-Ratio. 7, 14, 19, 30, 58, 61, 71, 73, 75, 78, 89, 90, 94, 100, 101, 103–105, 107, 113, 121–126, 129, 130, 133, 134, 152, 159, 162, 163, 165, 166, 176, 178–184, 205

**SotA** State-of-the-Art. 5, 40, 41, 66, 116, 145, 154, 155, 157, 205

**SPC** Single Parity Check. 37, 47, 191, 205

**SR** Sum-Rate. 131, 132, 137, 205

**SR** Software Radio. 4, 205

**srsLTE** srsLTE. 5, 205

**srsRAN** srsRAN. 5, 205

**SSC** Simplified Successive Cancellation. 35, 36, 205

**SSE** Streaming SIMD Extensions. 34, 156, 205

**STL** Standard Template Library. 205

**TACNET 4.0** TACtile interNET 4.0. 2, 147, 176, 205

**Tal&Vardy** Tal & Vardy. 205

**TDD** Time-Division-Duplex. 18, 118, 121, 136, 151, 205

**TDM** Time-Division-Multiplex. 119, 134–145, 151, 188, 205

**TFL** Time-Frequency-Localization. 69, 70, 72, 205

**TREP** TypeII Triple Repetition. 37, 205

**TRX3970X** AMD Ryzen Threadripper 3970X. 43, 59, 79, 164, 168, 169, 172–175, 205

**TTI** Transmission Time Interval. 117, 118, 205

**TypeV** TypeV. 37, 205

**UDN** Ultra Dense Network. 3, 5, 93, 205

**UDP** User Datagram Protocol. 150, 155, 160, 176, 205

**UE** User Equipment. 151, 152, 172–175, 205

# Bibliography

[3GP17] 3GPP, "Study on new radio access technology: Radio access architecture and interfaces", Technical Report 38.801 V14.0.0, 3rd Generation Partnership Project, Valbonne, France, March 2017.

[3GP19a] 3GPP, "Service requirements for cyber-physical control applications in vertical domains; Stage 1", Technical Specification 22.104 V16.1.0, 3rd Generation Partnership Project, Valbonne, France, March 2019.

[3GP19b] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz", Technical Report 38.901 - V16.0.0, 3rd Generation Partnership Project, Valbonne, France, October 2019.

[3GP20] 3GPP, "Study on Communication for Automation in Vertical Domains", Technical Report 22.804 V16.3.0, 3rd Generation Partnership Project, Valbonne, France, July 2020.

[ADE$^{+}$19] M. Aleksy, F. Dai, N. Enayati, P. Rost, and G. Pocovi, "Utilizing 5G in Industrial Robotic Applications", International Conference on Future Internet of Things and Cloud (FiCloud), volume 7, pp. 278–284, IEEE, Istanbul, Turkey, August 2019. DOI:10.1109/FiCloud.2019.00046.

[AKE08] A. B. Awoyesila, C. Kasparis, and B. G. Evans, "Improved preamble-aided timing estimation for OFDM systems", IEEE Communications Letters, volume 12, no. 11, pp. 825 – 827, November 2008. DOI:10.1109/LCOMM.2008.081054.

[ALF04] S. Allpress, C. Luschi, and S. Felix, "Exact and Approximated Expressions of the Log-Likelihood Ratio for 16-QAM Signals", Asilomar Conference on Signals, Systems and Computers, volume 38, IEEE, Pacific Grove, CA, USA, November 2004. DOI:10.1109/ACSSC.2004.1399245.

[Arı09] E. Arıkan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels", IEEE Transactions on Information Theory, volume 55, no. 7, pp. 3051–3073, July 2009. DOI:10.1109/TIT.2009.2021379.

[Arı11] E. Arıkan, “Systematic polar coding”, IEEE Communications Letters, volume 15, no. 8, pp. 860–862, August 2011. DOI:10.1109/LCOMM.2011.061611.110862.

[Ars15] K. Arshad, “LTE system level performance in the presence of CQI feedback uplink delay and mobility”, International Conference on Communications, Signal Processing, and their Applications (ICCSPA), pp. 1–5, IEEE, Sharjah, United Arab Emirates, February 2015. DOI:10.1109/ICCSPA.2015.7081294.

[AVK19] A. M. Alba, J. H. G. Velásquez, and W. Kellerer, “An adaptive functional split in 5G networks”, Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 410–416, IEEE, Paris, France, April 2019. DOI:10.1109/INFCOMW.2019.8845147.

[AYK11] A. Alamdar-Yazdi and F. R. Kschischang, “A Simplified Successive-Cancellation Decoder for Polar Codes”, IEEE Communications Letters, volume 15, no. 12, pp. 1378–1380, December 2011. DOI:10.1109/LCOMM.2011.101811.111480.

[AZB$^+$21] R. Ali, Y. B. Zikria, A. K. Bashir, S. Garg, and H. S. Kim, “URLLC for 5G and Beyond: Requirements, Enabling Incumbent Technologies and Network Intelligence”, IEEE Access, volume 9, pp. 67064–67095, April 2021. DOI:10.1109/ACCESS.2021.3073806.

[Bah21] D. Bahr, “CRC++”, March 2021.

[BAL$^+$19] J. H. Bae, A. Abotabl, H.-P. Lin, K.-B. Song, and J. Lee, “An overview of channel coding for 5G NR cellular communications”, APSIPA Transactions on Signal and Information Processing, volume 8, no. 1, p. 17, June 2019. DOI:10.1017/ATSIP.2019.10.

[BAS$^+$05] K. Brueninghaus, D. Astely, T. Salzer, S. Visuri, A. Alexiou, S. Karger, and G. Seraji, “Link Performance Models for System Level Simulations of Broadband Radio Access Systems”, International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), volume 16, IEEE, Berlin, Germany, September 2005. DOI:10.1109/PIMRC.2005.1651855.

[BBK22] B. Brik, K. Boutiba, and A. Ksentini, “Deep Learning for B5G Open Radio Access Network: Evolution, Survey, Case Studies, and Challenges”, IEEE Open Journal of the Communications Society, volume 3, pp. 228–250, January 2022. DOI:10.1109/OJCOMS.2022.3146618.

[BCL21] V. Bioglio, C. Condo, and I. Land, “Design of Polar Codes in 5G New Radio”, IEEE Communications Surveys Tutorials, volume 23, no. 1, pp. 29–40, January 2021. DOI:10.1109/COMST.2020.2967127.

[BD87] G. E. P. Box and N. R. Draper, Empirical Model-Building and Response Surfaces, Wiley, New York, NY, USA, 1987, ISBN 978-0-471-81033-9.

[BDG+17] C. Bockelmann, A. Dekorsy, A. Gnad, L. Rauchhaupt, A. Neumann, D. Block, U. Meier, J. Rust, S. Paul, F. Mackenthun, A. Weinand, H. Schotten, J. Siemons, T. Neugebauer, and M. Ehlich, "Hi-Flecs: Innovative Technologies for Low-Latency Wireless Closed-Loop Industrial Automation Systemsi", Mobilkommunikation - Technologien und Anwendungen, p. 6, VDE, Osnabrück, Germany, May 2017, ISBN 978-3-8007-4408-4.

[BGL17] V. Bioglio, F. Gabry, and I. Land, "Low-Complexity Puncturing and Shortening of Polar Codes", Wireless Communications and Networking Conference Workshops (WCNCW), pp. 0–5, IEEE, San Francisco, CA, USA, March 2017. DOI:10.1109/WCNCW.2017.7919040.

[Blo21] B. Bloessl, "gr-sched repository", November 2021.

[BM06] G. Bauch and J. S. Malik, "Cyclic delay diversity with bit-interleaved coded modulation in orthogonal frequency division multiple access", IEEE Transactions on Wireless Communications, volume 5, no. 8, pp. 2092–2100, August 2006. DOI:10.1109/TWC.2006.1687724.

[BM20] D. Brennan and V. Marojevic, "UHD-DPDK Performance Analysis for Advanced Software Radio Communications", arXiv, volume 2011.06355, p. 7, November 2020.

[BMH19] B. Bloessl, M. Müller, and M. Hollick, "Benchmarking and Profiling the GNU Radio Scheduler", GNU Radio Conference (GRCon), volume 9, p. 8, GNU Radio Foundation, Huntsville, AL, USA, September 2019.

[Boc12] C. Bockelmann, Robust Link Adaption in Coded OFDM Systems, number 23 in Forschungsberichte aus dem Arbeitsbereich Nachrichtentechnik der Universität Bremen, Shaker Verlag, Aachen, Germany, April 2012, ISBN 978-3-8440-0913-2.

[BRW+15] J. Bartelt, P. Rost, D. Wübben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and backhaul requirements of flexibly centralized radio access networks", IEEE Wireless Communications, volume 22, no. 5, pp. 105–111, October 2015. DOI:10.1109/MWC.2015.7306544.

[BSPB15] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-Based Successive Cancellation List Decoding of Polar Codes", IEEE Transactions on Signal Processing, volume 63, no. 19, pp. 5165–5179, October 2015. DOI:10.1109/TSP.2015.2439211.

[CB14] L. Chen and M. A. Babar, "Towards an Evidence-Based Understanding of Emergence of Architecture through Continuous Refactoring in Agile Software Development", Conference on Software Architecture, pp. 195–204, IEEE, Sydney, NSW, Australia, April 2014. DOI:10.1109/WICSA.2014.45.

[CBL18] C. Condo, V. Bioglio, and I. Land, "Generalized Fast Decoding of Polar Codes", Global Communications Conference (GLOBECOM), IEEE, Abu Dhabi, United Arab Emirates, December 2018. DOI:10.1109/GLOCOM.2018.8648105.

[CCS15] CCSDS, "Short Block Length LDPC Codes for TC Synchronization and Channel Coding", Experimental Specification CCSDS 231.1-0-1, Consultative Committee for Space Data Systems, Washington, DC, USA, April 2015.

[CCY$^+$15] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks - A Technology Overview", IEEE Communications Surveys & Tutorials, volume 17, no. 1, pp. 405–426, 2015. DOI:10.1109/COMST.2014.2355255.

[CDG$^+$21] P. Carbone, G. Dan, J. Gross, B. Goeransson, and M. Petrova, "NeuroRAN: Rethinking Virtualization for AI-native Radio Access Networks in 6G", arXiv, volume 2104.08111, April 2021. DOI:10.48550/arXiv.2104.08111. ArXiv: 2104.08111.

[CHL$^+$19] A. Cassagne, O. Hartmann, M. Léonardon, K. He, C. Leroux, R. Tajan, O. Aumage, D. Barthou, T. Tonnellier, V. Pignoly, B. Le Gal, and C. Jégo, "AFF3CT: A Fast Forward Error Correction Toolbox!", SoftwareX, volume 10, p. 100345, July 2019. DOI:10.1016/j.softx.2019.100345.

[CLT$^+$21] A. Cassagne, M. Léonardon, R. Tajan, Vyacheslav, T. Tonnellier, B. PETIT, CGigi, V. Pignoly, E. JANIN, C. Leroux, O. Hartmann, and J. Demel, "A Fast Forward Error Correction Toolbox (AFF3CT)", June 2021. DOI:10.5281/zenodo.595356.

[Cor14] J. Corbet, "Foo over UDP", October 2014.

[cpp21a] cppreference.com community, "C++ Date and time utilities", March 2021.

[cpp21b] cppreference.com community, "C++ reference", March 2021.

[CSD16] L. Chandesris, V. Savin, and D. Declercq, "An improved SCFlip decoder for polar codes", Global Communications Conference (GLOBECOM), IEEE, Washington, DC, USA, December 2016. DOI:10.1109/GLOCOM.2016.7841594.

[CSD17] L. Chandesris, V. Savin, and D. Declercq, "On Puncturing Strategies for Polar Codes", International Conference on Communications Workshops (ICC Workshops), IEEE, Paris, France, May 2017. DOI:10.1109/ICCW.2017.7962751.

[CSD18] L. Chandesris, V. Savin, and D. Declercq, "Dynamic-SCFlip Decoding of Polar Codes", IEEE Transactions on Communications, volume 66, no. 6, pp. 2333–2345, June 2018. DOI:10.1109/TCOMM.2018.2793887.

[CSW14] Y. Chen, F. Schaich, and T. Wild, “Multiple access and waveforms for 5G: IDMA and universal filtered multi-carrier”, Vehicular Technology Conference (VTC Spring), volume 79, IEEE, Seoul, Korea (South), May 2014. DOI:10.1109/VTCSpring.2014.7022995.

[CTB98] G. Caire, G. Taricco, and E. Biglieri, “Bit-Interleaved Coded Modulation”, IEEE Transactions on Information Theory, volume 44, no. 3, pp. 927–946, May 1998. DOI:10.1109/18.669123.

[CWC17] Y. Chen, X. Wang, and L. Cai, “Head-of-Line Access Delay-Based Scheduling Algorithm for Flow-Level Dynamics”, IEEE Transactions on Vehicular Technology, volume 66, no. 6, pp. 5387–5397, June 2017. DOI:10.1109/TVT.2016.2625326.

[DBD17a] J. Demel, C. Bockelmann, and A. Dekorsy, “Evaluation of a Software Defined GFDM Implementation for Industry 4.0 Applications”, International Conference on Industrial Technology (ICIT), IEEE, Toronto, Canada, March 2017. DOI:10.1109/ICIT.2017.7915548.

[DBD$^+$17b] J. Demel, C. Bockelmann, A. Dekorsy, A. Rode, S. Koslowski, and F. K. Jondral, “An optimized GFDM software implementation for future Cloud-RAN and field tests”, GNU Radio Conference (GRCon), volume 7, p. 9, GNU Radio Foundation, San Diego, CA, USA, September 2017.

[DBD19] J. Demel, C. Bockelmann, and A. Dekorsy, “Industrial Radio Link Abstraction Models for Short Packet Communication with Polar Codes”, International ITG Conference on Systems, Communications and Coding (SCC), volume 12, pp. 257–262, VDE, Rostock, Germany, February 2019. DOI:10.30420/454862044.

[DBD20] J. Demel, C. Bockelmann, and A. Dekorsy, “Burst error analysis of scheduling algorithms for 5G NR URLLC periodic deterministic communication”, Vehicular Technology Conference (VTC Spring), volume 91, IEEE, Antwerp, Belgium, May 2020. DOI:10.1109/VTC2020-Spring48590.2020.9129493.

[DBD21] J. Demel, C. Bockelmann, and A. Dekorsy, “Method for a sending entity and method for a receiving entity in a network environment”, Luxembourg, Patent LU101567B1, June 2021.

[DBD23] J. Demel, C. Bockelmann, and A. Dekorsy, “An ultra reliable low latency Cloud RAN implementation in GNU Radio for automated guided vehicles”, Workshop on Smart Antennas (WSA) & Conference on Systems, Communications, and Coding (SCC), VDE, Braunschweig, Germany, February 2023, ISBN 978-3-8007-6050-3.

[DDA$^+$22] J. Demel, M. Dickens, D. Anderson, B. Ashton, P. Balister, D. Behar, S. Behnke, A. Bekhit, A. Bhowmick, E. Blossom, J. Blum, A. M. Bottoms, E. Briggs, J. Cardoso, P. Cercueil, J. Corgan, N. Corgan, L. Cruz, R. Economos, B. P. Enochs, C. Fernandez, M. Fischer,

N. Foster, D. Geiger, P. Giard, G. Goavec-Merou, B. Hilburn, A. Holguin, J. Iwamoto, M. Kaesberger, M. Lichtman, K. A. Logue, M. Lundmark, S. Markgraf, C. Mayer, N. McCarthy, N. McCarthy, D. Miralles, S. Munaut, M. Müller, G. Nieboer, T. O'Shea, J. Olivain, S. Oltmanns, J. Pinkava, M. Piscopo, J. M. H. Quiceno, M. Rene, F. Ritterhoff, D. Robertson, F. L. L. Rocca, A. Rode, A. Rodionov, T. Rondeau, T. Sekine, K. Semich, V. Sergeev, A. Slokva, C. Smith, A. Stigo, A. Thompson, R. Thompson, V. Velichkov, R. Volz, A. Walls, D. Ward, N. West, B. M. Wiedemann, S. Wunsch, V. Zapodovnikov, J. Škarvada, Aang23, AlexandreRouma, Andrew, Zlika, luz.paz, and rear1019, "Vector-Optimized Library of Kernels (VOLK)", February 2022. DOI:10.5281/zenodo.6052858.

[Dem22a] J. Demel, "GNU Radio Symbolmapping (gr-symbolmapping)", December 2022. DOI:10.5281/zenodo.5997647.

[Dem22b] J. Demel, "gr-latency", December 2022. DOI:10.5281/zenodo.6338010.

[Dem22c] J. Demel, "gr-polarwrap", December 2022. DOI:10.5281/zenodo.6325778.

[Dem22d] J. Demel, "py-channelmodel", November 2022. DOI:10.5281/zenodo.7382872.

[Dem23] J. Demel, "gr-tacmac", January 2023. DOI:10.5281/zenodo.7525406.

[DG22] J. Demel and L. Göhrs, "XFDMSync", December 2022. DOI:10.5281/zenodo.6514722.

[DHC$^+$19] M. Düngen, T. Hansen, R. Croonenbroeck, R. Kays, B. Holfeld, D. Wieruch, P. W. Berenguer, V. Jungnickel, D. Block, and U. Meier, "Channel measurement campaigns for wireless industrial automation", at - Automatisierungstechnik, volume 67, no. 1, pp. 7–28, January 2019. DOI:10.1515/auto-2018-0052.

[DKJ15] J. Demel, S. Koslowski, and F. K. Jondral, "A LTE receiver framework using GNU Radio", Journal of Signal Processing Systems, volume 78, no. 3, pp. 313–320, January 2015. DOI:10.1007/s11265-014-0959-z.

[DKP$^+$17] S. Dräxler, H. Karl, M. Peuster, H. R. Kouchaksaraei, M. Bredel, J. Lessmann, T. Soenen, W. Tavernier, S. Mendel-Brin, and G. Xilouris, "SONATA: Service programming and orchestration for virtualized software networks", International Conference on Communications Workshops (ICC Workshops), pp. 973–978, IEEE, Paris, France, May 2017. DOI:10.1109/ICCW.2017.7962785. ISSN: 2474-9133.

[DL22] J. Demel and F. Lotze, "polar-codes", December 2022. DOI:10.5281/zenodo.6325747.

[DMB$^{+}$20] J. Demel, T. Monsees, C. Bockelmann, D. Wübben, and A. Dekorsy, “Cloud-RAN Fronthaul Rate Reduction via IBM-based Quantization for Multicarrier Systems”, International ITG Workshop on Smart Antennas (WSA), volume 24, pp. 1–6, VDE, Hamburg, Germany, February 2020, ISBN 978-3-8007-5200-3.

[DMG$^{+}$15] M. Danneberg, N. Michailow, I. Gaspar, M. Matthé, D. Zhang, L. L. Mendes, and G. Fettweis, “Implementation of a 2 by 2 MIMO-GFDM Transceiver for Robust 5G Networks”, International Symposium on Wireless Communication Systems (ISWCS), IEEE, Brussels, Belgium, August 2015. DOI:10.1109/ISWCS.2015.7454336.

[DMW$^{+}$17] S. Dietrich, G. May, O. Wetter, H. Heeren, and G. Fohler, “Performance indicators and use case analysis for wireless networks in factory automation”, International Conference on Emerging Technologies and Factory Automation (ETFA), volume 22, IEEE, Limassol, Cyprus, September 2017. DOI:10.1109/ETFA.2017.8247605.

[DNS$^{+}$17] J. Dai, K. Niu, Z. Si, C. Dong, and J. Lin, “Does Gaussian Approximation Work Well for the Long-Length Polar Code Construction?”, IEEE Access, volume 5, pp. 7950–7963, April 2017. DOI:10.1109/ACCESS.2017.2692241.

[DPL15] A. Dyck, R. Penners, and H. Lichter, “Towards Definitions for Release Engineering and DevOps”, International Workshop on Release Engineering, pp. 3–3, IEEE, Florence, Italy, May 2015. DOI:10.1109/RELENG.2015.10.

[DPS18] E. Dahlman, S. Parkvall, and J. Sköld, 5G NR: The Next Generation Wireless Access Technology, Academic Press, London [England] ; San Diego, CA, 2018. DOI:10.1016/C2017-0-01347-2.

[DRKK22] J. Demel, A. Rode, A. Kielstein, and S. Koslowski, “GNU Radio GFDM Modulator & Demodulator (gr-gfdm)”, December 2022. DOI:10.5281/zenodo.6532410.

[Du08] J. Du, Pulse Shape Adaptation and Channel Estimation in Generalised Frequency Division Multiplexing Systems, PhD Thesis, KTH, Stockholm, Sweden, December 2008. ISBN: 978-91-7415-187-9.

[Dwo01] M. Dworkin, “Recommendation for Block Cipher Modes of Operation, Methods and Techniques”, Special Publication 800-38A, National Institute of Standards and Technology, Gaithersburg, MD, USA, December 2001.

[Dwo16] M. Dworkin, “Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication”, Special Publication 800-38B, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2016. DOI:10.6028/NIST.SP.800-38B.

[Dyk82] Dykstra, Edsger W., “Why numbering should start at zero”, Manuscript 831, University of Texas, Plataanstraat 5 5671 AL NUENEN The Netherlands, August 1982.

[ETS18a] ETSI, “5G NR: Physical layer procedures for control”, Technical Specification 138.213 V15.3.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, October 2018.

[ETS18b] ETSI, “5G NR: Physical layer procedures for data”, Technical Specification 138.214 V15.3.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, October 2018.

[ETS18c] ETSI, “5G NR: Radio Resource Control (RRC)”, Technical Specification 138.331 V15.2.1, European Telecommunications Standards Institute, Sophia-Antipolis, France, June 2018.

[ETS18d] ETSI, “5G Study on New Radio (NR) access technology”, Technical Report 138.912 V15.0.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, September 2018.

[ETS19a] ETSI, “5G NR: User Equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone”, Technical Specification 138.101 V15.5.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, May 2019.

[ETS19b] ETSI, “LTE: Physical channels and modulation”, Technical Specification 136.211 V15.7.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, October 2019.

[ETS20a] ETSI, “5G NR: Multiplexing and channel coding”, Technical Specification 138.212 V15.8.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, January 2020.

[ETS20b] ETSI, “5G: Study on channel model for frequencies from 0.5 to 100 GHz”, Technical Report 138.901 V16.1.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, November 2020.

[ETS20c] ETSI, “LTE: Multiplexing and channel coding”, Technical Specification 136.212 V16.2.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, July 2020.

[ETS21] ETSI, “5G NR: Base Station (BS) radio transmission and reception”, Technical Specification 138.104 V15.15.0, European Telecommunications Standards Institute, Sophia-Antipolis, France, October 2021.

[Ett21] Ettus Research, a National Instruments Brand, “USRP Hardware Driver”, December 2021, original-date: 2013-03-27T19:52:19Z.

[FB14] U. U. Fayyaz and J. R. Barry, “Low-Complexity Soft-Output Decoding of Polar Codes”, IEEE Journal on Selected Areas in Communications, volume 32, no. 5, pp. 958–966, May 2014. DOI:10.1109/JSAC.2014.140515.

[FDG$^+$18] T. Fehrenbach, R. Datta, B. Göktepe, T. Wirth, and C. Hellge, “URLLC Services in 5G Low Latency Enhancements for LTE”, Vehicular Technology Conference (VTC Fall), volume 88, pp. 1–6, IEEE, Chicago, IL, USA, August 2018. DOI:10.1109/VTCFall.2018.8690663.

[Fet14] G. P. Fettweis, “The tactile internet: Applications and challenges”, IEEE Vehicular Technology Magazine, volume 9, no. 1, pp. 64–70, March 2014. DOI:10.1109/MVT.2013.2295069.

[FJ05] M. Frigo and S. G. Johnson, “The design and implementation of FFTW3”, Proceedings of the IEEE, volume 93, no. 2, pp. 216–231, February 2005. DOI:10.1109/JPROC.2004.840301.

[FLS$^+$21] M. Feathers, J. Lacoste, E. Sommerlade, B. Lepilleur, B. Bakker, and S. Robbins, “CppUnit - The C++ Unit Test Library”, March 2021.

[Fre07] Free Software Foundation, “GNU General Public License v3.0 or later”, June 2007.

[Fre14] L. E. Frenzel, Principles of electronic communication systems, McGraw-Hill, New York, NY, USA, fourth edition edition, 2014, ISBN 978-0-07-337385-0.

[Gal08] R. G. Gallager, Principles of digital communication, Cambridge University Press, Cambridge, NY, USA, 1st edition, 2008. DOI:10.1017/CBO9780511813498. OCLC: ocn166382261.

[Gia16] P. Giard, High-Speed Decoders for Polar Codes, PhD Thesis, McGill University, Montreal, Canada, September 2016. DOI:10.1007/978-3-319-59782-9.

[GLJ15] B. L. Gal, C. Leroux, and C. Jego, “Multi-Gb/s Software Decoding of Polar Codes”, IEEE Transactions on Signal Processing, volume 63, no. 2, pp. 349–359, January 2015. DOI:10.1109/TSP.2014.2371781.

[GM13] I. Gómez-Miguelez, Radio and computing resource management in SDR clouds, Doctoral thesis, Universitat Politècnica de Catalunya, Catalunya, Spain, December 2013.

[GMGSS$^+$16] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, “srsLTE: an open-source platform for LTE evolution and experimentation”, International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization (WiNTECH), volume 10, pp. 25–32, Association for Computing Machinery, New York, NY, USA, October 2016. DOI:10.1145/2980159.2980163.

[GMMF14] I. S. Gaspar, L. L. Mendes, N. Michailow, and G. Fettweis, “A synchronization technique for generalized frequency division multiplexing”, EURASIP Journal on Advances in Signal Processing, volume 67, no. 1, pp. 1687–6180, May 2014. DOI:10.1186/1687-6180-2014-67.

[GMN+13] I. Gaspar, N. Michailow, A. Navarro, E. Ohlmer, S. Krone, and G. Fettweis, “Low complexity GFDM receiver based on sparse frequency domain processing”, Vehicular Technology Conference (VTC Spring), volume 77, IEEE, Dresden, Germany, June 2013. DOI:10.1109/VTCSpring.2013.6692619.

[Gol05] A. Goldsmith, Wireless communications, Cambridge University Press, Cambridge, UK, 2005. DOI:10.1017/CBO9780511841224. OCLC: 320129180.

[Goo21] Google, “Benchmark”, March 2021.

[Gra13] E. Grayver, Implementing Software Defined Radio, Springer, New York, NY, USA, 2013. DOI:10.1007/978-1-4419-9332-8.

[Gra19] E. Grayver, “Scaling the Fast x86 DVB-S2 Decoder to 1 Gbps”, Aerospace Conference, pp. 1–9, IEEE, Big Sky, MT, USA, March 2019. DOI:10.1109/AERO.2019.8742225.

[GRLa17] H. Gamage, N. Rajatheva, and M. Latva-aho, “Channel coding for enhanced mobile broadband communication in 5G systems”, European Conference on Networks and Communications (EuCNC), pp. 1–6, IEEE, Oulu, Finland, June 2017. DOI:10.1109/EuCNC.2017.7980697.

[GSL+18] P. Giard, G. Sarkis, C. Leroux, C. Thibeault, and W. J. Gross, “Low-Latency Software Polar Decoders”, Journal of Signal Processing Systems, volume 90, no. 5, pp. 761–775, May 2018. DOI:10.1007/s11265-016-1157-y.

[GSS+18] M. Gundall, J. Schneider, H. D. Schotten, M. Aleksy, D. Schulz, N. Franchi, N. Schwarzenberg, C. Markwart, R. Halfmann, P. Rost, D. Wübben, A. Neumann, M. Düngen, T. Neugebauer, R. Blunk, M. Kus, and J. Grießbach, “5G as Enabler for Industrie 4.0 Use Cases: Challenges and Concepts”, International Conference on Emerging Technologies and Factory Automation (ETFA), volume 23, pp. 1401–1408, IEEE, Turin, Italy, September 2018. DOI:10.1109/ETFA.2018.8502649.

[GSS+21] M. Gundall, M. Strufe, H. D. Schotten, P. Rost, C. Markwart, R. Blunk, A. Neumann, J. Grießbach, M. Aleksy, and D. Wübben, “Introduction of a 5G-Enabled Architecture for the Realization of Industry 4.0 Use Cases”, IEEE Access, volume 9, pp. 25508–25521, February 2021. DOI:10.1109/ACCESS.2021.3057675.

[HA17] M. Hanif and M. Ardakani, “Fast Successive-Cancellation Decoding of Polar Codes: Identification and Decoding of New Nodes”, IEEE Communications Letters, volume 21, no. 11, pp. 2360–2363, November 2017. DOI:10.1109/LCOMM.2017.2740305.

[HBL$^+$17] G. He, J. Belfiore, I. Land, G. Yang, X. Liu, Y. Chen, R. Li, J. Wang, Y. Ge, R. Zhang, and W. Tong, "Beta-Expansion: A Theoretical Framework for Fast and Recursive Construction of Polar Codes", Global Communications Conference (GLOBECOM), IEEE, Singapore, December 2017. DOI:10.1109/GLOCOM.2017.8254146.

[HCSM21] X. He, K. Cai, W. Song, and Z. Mei, "Dynamic Programming for Sequential Deterministic Quantization of Discrete Memoryless Channels", IEEE Transactions on Communications, volume 69, no. 6, pp. 3638–3651, June 2021. DOI:10.1109/TCOMM.2021.3062838.

[Hin13] P. Hintjens, ZeroMQ : Messaging for Many Applications, O'Reilly Media, Sebastopol, CA, USA, 1st edition, April 2013, ISBN 978-1-4493-3445-1.

[HMW$^+$20] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy", Nature, volume 585, no. 7825, pp. 357–362, September 2020. DOI:10.1038/s41586-020-2649-2.

[HP14] J. L. Hennessy and D. A. Patterson, Computer Organization and Design: The Hardware/Software Interface, Morgan Kauffmann, San Francisco, CA, USA, 1st edition, May 2014, ISBN 978-1-4832-2118-2.

[HWD20] S. Hassanpour, D. Wübben, and A. Dekorsy, "Generalized Distributed Information Bottleneck for Fronthaul Rate Reduction at the Cloud-RANs Uplink", Global Communications Conference (GLOBECOM), pp. 1–6, IEEE, Taipei, Taiwan, December 2020. DOI:10.1109/GLOBECOM42002.2020.9322494.

[HZA$^+$11] R. He, Z. Zhong, B. Ai, L. Xiong, and J. Ding, "The effect of reference distance on path loss prediction based on the measurements in high-speed railway viaduct scenarios", International ICST Conference on Communications and Networking in China (CHINACOM), pp. 1201–1205, IEEE, Harbin, China, August 2011. DOI:10.1109/ChinaCom.2011.6158340.

[IEE12] IEEE, "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", Technical Specification 802.11-2012, IEEE, Piscataway, NJ, USA, March 2012. DOI:10.1109/IEEESTD.2012.6178212.

[IEE19] IEEE, "IEEE Standard for Floating-Point Arithmetic", Technical Specification 754-2019, IEEE, New York, NY, USA, July 2019. DOI:10.1109/IEEESTD.2019.8766229.

[Int21] Intel Corporation, "Intel Intrinsics Guide", March 2021.

[ISLP06] T. Iwata, J. Song, J. Lee, and R. Poovendran, "The AES-CMAC Algorithm", Request for Comments 4493, Internet Engineering Task Force (IETF), Wilmington, DE, USA, June 2006. DOI:10.17487/rfc4493.

[ISO17] ISO, "ISO/IEC 14882:2017 — Programming languages — C++", Standard 14882:2017, International Organization for Standardization, Geneva, Switzerland, December 2017.

[Jäc05] P. Jäckel, "A note on multivariate Gauss-Hermite quadrature", p. 5, May 2005.

[JRS+22] W. Jakob, J. Rhinelander, H. Schreiner, D. Moldovan, I. Smirnov, R. W. Grosse-Kunstleve, Y. Jadoul, A. Gokaslan, B. Staletic, A. Huebl, E. Cousineau, B. Merry, A. Lee, S. Corlay, S. Izmailov, L. A. Burns, Dan, D. Spicuzza, S. Lyskov, T. Houliston, bennorth, jbarlow83, P. Schellart, B. Pritchard, R. Haschke, B. Schäling, tmiasko, and J. Maitin-Shepard, "pybind11", February 2022. DOI:10.5281/zenodo.5807779.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Request for Comments 2104, Internet Engineering Task Force (IETF), Wilmington, DE, USA, February 1997. DOI:10.17487/rfc2104.

[KC04] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks", International Conference on Dependable Systems and Networks, pp. 145–154, IEEE, Florence, Italy, June 2004. DOI:10.1109/DSN.2004.1311885.

[KD18] Kammeyer, Karl-Dirk and A. Dekorsy, Nachrichtenübertragung, Informationstechnik, Springer Vieweg, Wiesbaden, Germany, 6th edition, 2018, ISBN 978-3-658-17004-2.

[KEXMH20] Z. B. Kaykac Egilmez, L. Xiang, R. G. Maunder, and L. Hanzo, "The Development, Operation and Performance of the 5G Polar Codes", IEEE Communications Surveys and Tutorials, volume 22, no. 1, pp. 96–122, March 2020. DOI:10.1109/COMST.2019.2960746.

[KSKW19] F. Kaltenberger, G. d. Souza, R. Knopp, and H. Wang, "The OpenAirInterface 5G New Radio Implementation: Current Status and Roadmap", International ITG Workshop on Smart Antennas (WSA), volume 23, pp. 1–5, VDE, Vienna Austria, April 2019, ISBN 978-3-8007-4939-3.

[KY14] B. M. Kurkoski and H. Yagi, "Quantization of Binary-Input Discrete Memoryless Channels", IEEE Transactions on Information

Theory, volume 60, no. 8, pp. 4544–4552, August 2014. DOI:10.1109/TIT.2014.2327016.

[LA20] Li, Chai and Akman, Arda, "O-RAN Use Cases and Deployment Scenarios", February 2020.

[LCC19] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks", IEEE Communications Surveys Tutorials, volume 21, no. 1, pp. 146–172, January 2019. DOI:10.1109/COMST.2018.2868805.

[LCL+19] M. Léonardon, A. Cassagne, C. Leroux, C. Jégo, L.-P. Hamelin, and Y. Savaria, "Fast and Flexible Software Polar List Decoders", Journal of Signal Processing Systems, volume 91, no. 8, pp. 937–952, January 2019. DOI:10.1007/s11265-018-1430-3.

[Lem17] S. Lembo, Functions and Abstractions for Radio Access Network Softwarization, Doctoral thesis, Aalto University, Aalto, Finland, November 2017.

[LGJ20] B. Le Gal and C. Jego, "Low-latency and high-throughput software turbo decoders on multi-core architectures", Annals of Telecommunications, volume 75, pp. 27–42, February 2020. DOI:10.1007/s12243-019-00727-5.

[LKK12] I. Latif, F. Kaltenberger, and R. Knopp, "Link abstraction for multi-user MIMO in LTE using interference-aware receiver", Wireless Communications and Networking Conference (WCNC), IEEE, Paris, France, April 2012. DOI:10.1109/WCNC.2012.6214489.

[LMA+22] Long, Jeff, Morman, Josh, Abele, Jason, Aigner, Philipp, Alok, Aradhana, Anastasopoulos, Achilleas, Anderson, Douglas, Apitzsch, André, Arnold, Julian, Ashton, Brennan, Atreides, Paul, Baier, Gerald, Balister, Philip, Banti, Stefano, Barnhart, William, Bauer, Mark, Becker, Johannes K, Beeri, Gilad, Behar, Doron, Bell, Richard C., Berman, Michael, Berndt, Luke, Bhowmick, Abhishek, Bigot, Peter A., Bird, Alistair, Bloessl, Bastian, T. Blomberg, Blossom, Eric, Blot, Emmanuel, Blum, Josh, Boone, Jared, Bottoms, A. Maitland, Boven, Paul, Braun, Martin, N. Bruce, J. Brucker, S. Bruns, M. Byers, P. Cercueil, C. Chavez, J. Chiang, B. Clark, T. F. Collins, T. Conn, J. Corgan, N. Corgan, M. Cornelius, M. Cottrell, G. Cox, A. Csete, N. Cuervo, J. R. Cutler, P. David, A. Davis, Demel, Johannes, B. Diaconescu, Dickens, Michael, M. v. Dijke, C. Donohue, J. Drake, C. Duffy, B. Duggan, J. Dulmage, J. Dumps, J. Durst, M. DvH, R. Economos, B. P. Enochs, G. Eslinger, D. Estévez, M. Ettus, Z. Feng, M. Fischer, T. Flynn, N. Foster, F. Franzen, I. Freire, P. Garver, P. Gauthier, D. Geiger, A. Gelman, K. Gentile, D. Gerhard, F. Gerlits, A. Ghani, J. Gilbert, S. Glass, G. Goavec-Merou, N. Goergen, B. Gottula, R. Govostes, D. Grambihler, M. Griffiths, A. Gupta, A. Gupta, S. Gupta, T. Habets, S. Haynal, M. D. v. Heel,

J. Hein, H. t. Hennepe, U. Hermann, B. Hilburn, S. Hitefield, A. Horden, P. Horvath, M. Hostetter, B. Hsu, T. Huggins, B. Iannucci, B. Jackson, M. Jameson, E. Johnson, L. Joost, A. Kalør, J. Kamat, S. Kapoor, K. Kat, D. Kawamoto, B. Kerler, E. Kigwana, C. Koehler, T. Koehn, S. Koslowski, D. Kozel, O. Kravchuk, P. Krysik, J. Krämer, T. Kuester, C. Kuethe, L. Kuzmiak, L. LANGE, A. Lajovic, S. Larew, W. Ledbetter, J. Ledet-Pedersen, M. Leech, L. P. Lessard, M. Lichtman, T. Lindfors, P. Lobo, S. Ludwig, A. Løfaldli, N. M, J. Madeira, J. Madeira, K. Maier, S. Markgraf, M. Maroti, T. May, C. Mayer, C. Mayo, N. McCarthy, N. McCarthy, C. McCodeface, C. McDiarmid, B. McGwier, K. McQuiggin, A. Mendez, A. Michel, M. Mills, K. Mochalov, R. Molenkamp, T. Monahan-Mitchell, C. Morin, S. Munaut, B. Muzika, E. S. Muñoz, M. Müller, S. Müller, C. A. R. Naranjo, S. Negi, T. Newman, C.-T. C. Nguy?n, G. Nieboer, P. Niedermayer, M. D. Nil, S. Nowlan, G. Nychis, T. O'Shea, J. Olivain, S. Olsen, S. Oltmanns, J. Orlando, B. Orr, M. Ossmann, V. P, B. Padalino, K. Patel, C. Patulea, J. Pendlum, M. Pentler, D. Pi, J. Pinkava, J. Pinkava, A. Pisarenko, M. Piscopo, M. Plett, D. Porges, H. Pyle, J. Quaresma, B. Radulescu, S. Rajendran, M. Rasmussen, K. Reid, M. Rene, B. Reynwar, M. Ribero, G. Richardson, W. Roberts, D. Robertson, F. L. Rocca, A. Rode, M. Roe, T. Rondeau, S. Ross, H. Rui, J. Saari, B. Sabaghi, J. Sallay, R. Savoye, F. P. Schmidt, J. Schmitz, E. Schneider, V. Schroer, J. Schueler, R. Schutt, B. Seeber, C. Seguinot, K. Semich, R. Sharan, R. Sharma, T. Shepard, A. Singh, A. Sloane, C. Smith, C. Solano, D. Sorber, R. Spanbauer, Z. Spytz, I.-E. Srairi, B. Stapleton, E. Statzer, M. Stiefel, D. Stolnikov, A. Suciu, C. Swiger, J. Szymaniak, I. Tagunov, S. Talbert, S. Tan, N. Temple, A. Thakur, A. Thomas, R. Thompson, S. Torborg, E. Trewhitt, B. L. Trotter, V. Tsiligiannis, T. Tsou, E. Tuerkyilmaz, B. Tyers, J. Uher, R. Undheim, K. Unice, G. Vanhoy, V. Velichkov, J. Vierinen, R. Volz, S. N. Vutukuri, H. Vågsether, A. Walls, M. Walters, D. Ward, G. Warnes, D. Weber, E. Wecksell, N. West, P. Wicks, B. M. Wiedemann, A. Willecke, D. Winter, P. Witkowski, V. Wollesen, F. Wunsch, S. Wunsch, H. v. Wyk, H. Xu, L. Z, V. Zapodovnikov, S. Zehl, A. Zhao, K. Zheng, C. Zhu, J. Zy, D. H. Yang, N. Østergaard, J. Šafář, J. Škarvada, AlexandraTrifan, Antetokounpo, AsciiWolf, Callyan, Cate, Dhruvadityamittal, Federico, Flamewires, Head4che, Ipsit, JaredD, Lennart, MBoerschig, Moeller, Niki, Oliver, Pavon, RedStone002, Seeker, Shane, Unknown, Zero_Chaos, A-Andre, Aidan, Beitler, Beroset, Cmrincon, Cswiger, Danielnachun, Eb, Efardin, Ejk, Elms, Esrh, Gdt, Gmazilla, Gr-Sp, Hatsunearu, Ilovezfs, Japm48, Jb41997, Jcoy, Karel, Karel, Kc1212, Lenhart, Linwei, Luzpaz, Mark, Masw, Matt, Mi-A, Phansel, Rajb245, Rear1019, Schneider42, Shwhiti, Soggysec, Tttx, Vermillionsands, Wakass, and Wcampbell, "GNU Radio", September 2022. DOI:10.5281/zenodo.2704343.

[LSL$^+$16] J. Lin, J. Sha, L. Li, C. Xiong, Z. Yan, and Z. Wang, "A high throughput belief propagation decoder architecture for polar codes", International Symposium on Circuits and Systems (ISCAS), IEEE, Montreal, QC, Canada, May 2016. DOI:10.1109/ISCAS.2016.7527193.

[Mar08] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall PTR, USA, 1st edition, 2008, ISBN 978-0-13-235088-4.

[MAXC16] J. Mao, M. A. Abdullahi, P. Xiao, and A. Cao, "A low complexity 256QAM soft demapper for 5G mobile system", European Conference on Networks and Communications (EuCNC), pp. 16–21, IEEE, Athens, Greece, June 2016. DOI:10.1109/EuCNC.2016.7560996.

[MB17] F. Mackenthun and J. Berg, "Secure machine-to-machine communication", SmartCard Workshop, Fraunhofer, Darmstadt, Germany, February 2017, ISBN 978-3-9813317-1-4.

[MBF$^+$05] A. F. Molisch, K. Balakkrishnan, A. Fort, J. Karedal, J. Kunisch, H. Schantz, U. Schuster, and K. Siwiak, "IEEE 802.15.4a channel model - final report", Technical Report 802.15.4a, IEEE, Monterey, 2005.

[MG11] P. Mell and T. Grance, "The NIST Definition of Cloud Computing", Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg, MD, USA, September 2011.

[MGG$^+$12] V. Marojevic, I. Gomez, P. L. Gilabert, G. Montoro, and A. Gelonch, "Resource management implications and strategies for SDR clouds", Analog Integrated Circuits and Signal Processing, volume 73, no. 2, pp. 473–482, November 2012. DOI:10.1007/s10470-012-9963-z.

[Mit92] J. Mitola, "Software Radios Survey, Critical Evaluation and Future Directions", National Telesystems Conference (NTC), IEEE, Washington, DC, USA, May 1992. DOI:10.1109/NTC.1992.267870.

[MLM22] J. Morman, M. Lichtman, and M. Müller, "The Future of GNU Radio: Heterogeneous Computing, Distributed Processing, and Scheduler-as-a-Plugin", Military Communications Conference (MILCOM), pp. 180–185, IEEE, Rockville, MD, USA, November 2022. DOI:10.1109/MILCOM55135.2022.10017973. ISSN: 2155-7586.

[MMF14] M. Matthe, L. L. Mendes, and G. Fettweis, "Generalized frequency division multiplexing in a gabor transform setting", IEEE Communications Letters, volume 18, no. 8, pp. 1379–1382, July 2014. DOI:10.1109/LCOMM.2014.2332155.

[MMG$^+$14] N. Michailow, M. Matthe, I. S. Gaspar, A. N. Caldevilla, L. L. Mendes, A. Festag, and G. Fettweis, "Generalized frequency division multiplexing for 5th generation cellular networks", IEEE Transactions on Communications, volume 62, no. 9, pp. 3045 – 3061, September 2014. DOI:10.1109/TCOMM.2014.2345566.

[MMM19] G. Mountaser, M. Mahlouji, and T. Mahmoodi, "Latency Bounds of Packet-Based Fronthaul for Cloud-RAN with Functionality Split", International Conference on Communications (ICC), pp. 1–6, IEEE, Shanghai, China, May 2019. DOI:10.1109/ICC.2019.8761906.

[MWBD15] F. Monsees, M. Woltering, C. Bockelmann, and A. Dekorsy, "Compressive sensing multi-user detection for multicarrier systems in sporadic machine type communication", Vehicular Technology Conference (VTC Spring), volume 81, IEEE, Glasgow, UK, May 2015. DOI:10.1109/VTCSpring.2015.7145755.

[MWD21] T. Monsees, D. Wübben, and A. Dekorsy, "Relative Entropy based Message Combining for Exploiting Diversity in Information Optimized Processing", International ITG Workshop on Smart Antennas (WSA), volume 25, VDE, French Riviera, France, November 2021, ISBN 978-3-8007-5686-5.

[MZS$^+$16] M. Matthe, D. Zhang, F. Schaich, T. Wild, R. Ahmed, and G. Fettweis, "A reduced complexity time-domain transmitter for UF-OFDM", Vehicular Technology Conference (VTC Spring), volume 83, IEEE, Nanjing, China, May 2016. DOI:10.1109/VTCSpring.2016.7504101.

[NC21] K. Niu and K. Chen, "Stack decoding of polar codes", Electronics Letters, volume 48, no. 12, pp. 695 –697, June 2021. DOI:10.1049/el.2012.1459.

[NCL13] K. Niu, K. Chen, and J. R. Lin, "Beyond turbo codes: Rate-compatible punctured polar codes", International Conference on Communications (ICC), IEEE, Budapest, Hungary, June 2013. DOI:10.1109/ICC.2013.6655078.

[NIS08] NIST, "FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC)", Standard FIPS PUB 198-1, National Institute of Standards and Technology, Gaithersburg, MD, USA, July 2008.

[O-R21] O-RAN Alliance e.V., "O-RAN Alliance", March 2021.

[OBJ13] N. Otterbach, M. Braun, and F. K. Jondral, "Wireless Networks In-the-Loop: Creating a SDR Development Environment", International Symposium on Wireless Communication Systems (ISWCS), volume 10, VDE, Ilmenau, Germany, August 2013, ISBN 978-3-8007-3529-7.

[OMM16] A. Osseiran, J. F. Monserrat, and P. Marsch (Editors), 5G Mobile and Wireless Communications Technology, Cambridge

University Press, Cambridge, UK, 1st edition, June 2016. DOI:10.1017/CBO9781316417744.

[Ope21] OpenSSL Software Foundation, “OpenSSL”, March 2021.

[PCB21] C. Pillet, C. Condo, and V. Bioglio, “Fast-SCAN decoding of Polar Codes”, International Symposium on Topics in Coding (ISTC), volume 11, IEEE, Montreal, QC, Canada, August 2021. DOI:10.1109/ISTC49272.2021.9594208.

[PHV15] D. Pompili, A. Hajisami, and H. Viswanathan, “Dynamic provisioning and allocation in Cloud Radio Access Networks (C-RANs)”, Ad Hoc Networks, volume 30, pp. 128–143, July 2015. DOI:10.1016/j.adhoc.2015.02.006.

[PND$^+$11] P. Pawelczak, K. Nolan, L. Doyle, S. W. Oh, and D. Cabric, “Cognitive radio: Ten years of experimentation and development”, IEEE Communications Magazine, volume 49, no. 3, pp. 90–100, March 2011. DOI:10.1109/MCOM.2011.5723805. Conference Name: IEEE Communications Magazine.

[Pos80] J. Postel, “User Datagram Protocol”, Request for Comments 768, Internet Engineering Task Force, Wilmington, DE, USA, August 1980. DOI:10.17487/RFC0768.

[Pos81a] J. Postel, “Internet Control Message Protocol”, Request for Comments 792, Internet Engineering Task Force, Wilmington, DE, USA, September 1981. DOI:10.17487/RFC0792.

[Pos81b] J. Postel, “Internet Protocol”, Request for Comments 791, Internet Engineering Task Force, Wilmington, DE, USA, September 1981. DOI:10.17487/RFC0791.

[PPM18] G. Pocovi, K. I. Pedersen, and P. Mogensen, “Joint Link Adaptation and Scheduling for 5G Ultra-Reliable Low-Latency Communications”, IEEE Access, volume 6, pp. 28912–28922, May 2018. DOI:10.1109/ACCESS.2018.2838585.

[Pro95] J. G. Proakis, Digital Communications, McGraw-Hill, New York, NY, USA, 3rd edition, 1995, ISBN 978-0-07-051726-4.

[Pyt21] Python Software Foundation, “Python”, March 2021.

[Rap09] T. S. Rappaport, Wireless Communications, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2009, ISBN 978-0-13-042232-3.

[Ray03] E. S. Raymond, The Art of Unix Programming, volume 1, Addison-Wesley, Boston, MA, USA, September 2003, ISBN 0-13-142901-9.

[RBD$^+$14] P. Rost, C. J. Bernardos, A. D. Domenico, M. D. Girolamo, M. Lalam, A. Maeder, D. Sabella, and D. Wübben, “Cloud technologies for flexible 5G radio access networks”, IEEE Communications Magazine, volume 52, no. 5, pp. 68–76, May 2014. DOI:10.1109/MCOM.2014.6898939.

[RBM$^+$15] P. Rost, I. Berberana, A. Maeder, H. Paul, V. Suryaprakash, M. Valenti, D. Wübben, A. Dekorsy, and G. Fettweis, “Benefits and challenges of virtualization in 5G radio access networks”, IEEE Communications Magazine, volume 53, no. 12, pp. 75–82, December 2015. DOI:10.1109/MCOM.2015.7355588.

[RU08] T. Richardson and R. Urbanke, Modern Coding Theory, Cambridge University Press, Cambridge, UK, 2008, ISBN 978-0-511-38007-5.

[Sar16] G. Sarkis, Efficient Encoders and Decoders for Polar Codes: Algorithms and Implementations, PhD Thesis, McGill University, Montreal, Canada, 2016.

[SC97] T. Schmidl and D. Cox, “Robust frequency and timing synchronization for OFDM”, IEEE Transactions on Communications, volume 45, no. 12, pp. 1613–1621, December 1997. DOI:10.1109/26.650240.

[SFVS20] G. Soós, D. Ficzere, P. Varga, and Z. Szalay, “Practical 5G KPI Measurement Results on a Non-Standalone Architecture”, Network Operations and Management Symposium (NOMS), pp. 1–5, IEEE, Budapest, Hungary, April 2020. DOI:10.1109/NOMS47738.2020.9110457. ISSN: 2374-9709.

[SGA14] A. Sahin, I. Guvenc, and H. Arslan, “A survey on multicarrier communications: Prototype filters, lattice structures, and implementation aspects”, IEEE Communications Surveys and Tutorials, volume 16, no. 3, pp. 1312 – 1338, August 2014. DOI:10.1109/SURV.2013.121213.00263.

[SGV$^+$16] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, “Fast List Decoders for Polar Codes”, IEEE Journal on Selected Areas in Communications, volume 34, no. 2, pp. 318–328, February 2016. DOI:10.1109/JSAC.2015.2504299.

[Skl01] B. Sklar, Digital Communications: Fundamentals and Applications, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2001, ISBN 0-13-084788-7.

[SKMB21] D. Segura, E. J. Khatib, J. Munilla, and R. Barco, “5G Numerologies Assessment for URLLC in Industrial Communications”, Sensors, volume 21, no. 7, p. 2489, April 2021. DOI:10.3390/s21072489.

[Sta06] W. Staff, “GNU Radio Opens an Unseen World”, Wired, June 2006. Section: tags.

[STG$^+$16] G. Sarkis, I. Tal, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, “Flexible and low-complexity encoding and decoding of systematic polar codes”, IEEE Transactions on Communications, volume 64, no. 7, pp. 2732 – 2745, July 2016. DOI:10.1109/TCOMM.2016.2574996.

[SWD$^+$18] J. Sachs, G. Wikstrom, T. Dudda, R. Baldemair, and K. Kittichokechai, “5G Radio Network Design for Ultra-Reliable Low-Latency Communication”, IEEE Network, volume 32, no. 2, pp. 24–31, April 2018. DOI:10.1109/MNET.2018.1700232.

[SWJ$^+$16] M. Sybis, K. Wesolowski, K. Jayasinghe, V. Venkatasubramanian, and V. Vukadinovic, “Channel coding for ultra-reliable low-latency communication in 5G systems”, Vehicular Technology Conference (VTC Fall), volume 84, IEEE, Montreal, QC, Canada, September 2016. DOI:10.1109/VTCFall.2016.7880930.

[TB02] F. Tosato and P. Bisaglia, “Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2”, International Conference on Communications (ICC), volume 2, pp. 664–668, IEEE, New York, NY, USA, April 2002. DOI:10.1109/ICC.2002.996940.

[Tri12] P. Trifonov, “Efficient design and decoding of polar codes”, IEEE Transactions on Communications, volume 60, no. 11, pp. 3221–3227, November 2012. DOI:10.1109/TCOMM.2012.081512.110872.

[TV13] I. Tal and A. Vardy, “How to construct polar codes”, IEEE Transactions on Information Theory, volume 59, no. 10, pp. 6562–6582, October 2013. DOI:10.1109/TIT.2013.2272694.

[TV15] I. Tal and A. Vardy, “List Decoding of Polar Codes”, IEEE Transactions on Information Theory, volume 61, no. 5, pp. 2213–2226, May 2015. DOI:10.1109/TIT.2015.2410251.

[UW20] L. Underberg and S. Willmann, “Categorization of industrial communication requirements as key to developing application profiles”, IFAC World Congress, volume 53, pp. 8297–8302, IFAC, Berlin, Germany, July 2020. DOI:10.1016/j.ifacol.2020.12.1919.

[Ver98] Verdú, Sergio, Multiuser Detection, Cambridge University Press, Cambridge, UK, November 1998, ISBN 978-0-521-59373-1.

[VGO$^+$20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G.-L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra,

J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza, "SciPy 1.0: fundamental algorithms for scientific computing in Python", Nature Methods, volume 17, no. 3, pp. 261–272, March 2020. DOI:10.1038/s41592-019-0686-2.

[VHV16] H. Vangala, Y. Hong, and E. Viterbo, "Efficient algorithms for systematic polar encoding", IEEE Communications Letters, volume 20, no. 1, pp. 17–20, January 2016. DOI:10.1109/LCOMM.2015.2497220.

[VVH15] H. Vangala, E. Viterbo, and Y. Hong, "A Comparative Study of Polar Code Constructions for the AWGN Channel", arXiv, volume 1501.02473, p. 9, January 2015. DOI:10.48550/arXiv.1501.02473.

[VWS$^+$13] V. Vakilian, T. Wild, F. Schaich, S. Ten Brink, and J. F. Frigon, "Universal-filtered multi-carrier technique for wireless systems beyond LTE", Globecom Workshops (GC Wkshps), IEEE, Atlanta, GA, USA, December 2013. DOI:10.1109/GLOCOMW.2013.6824990.

[WE08] I. Wong and B. Evans, Resource Allocation In Multiuser Multicarrier Wireless Systems, Springer, Boston, MA, USA, 1st edition, 2008. DOI:10.1007/978-0-387-74945-7.

[Wel67] P. Welch, "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms", IEEE Transactions on Audio and Electroacoustics, volume 15, no. 2, pp. 70–73, June 1967. DOI:10.1109/TAU.1967.1161901.

[Wil21] M. van der Wilk, "A notebook detailing multivariate Gauss-Hermite quadrature", June 2021.

[WLS14] D. Wu, Y. Li, and Y. Sun, "Construction and block error rate analysis of polar codes over AWGN channel based on gaussian approximation", IEEE Communications Letters, volume 18, no. 7, pp. 1099–1102, July 2014. DOI:10.1109/LCOMM.2014.2325811.

[WO20] C. Walton and C. O'Dell, "Bridging Analog Land Mobile Radio to LTE Mission Critical Push-to-Talk Communications", National Institute of Standards and Technology, volume NISTIR 8338, no. 8338, p. 33, December 2020. DOI:10.6028/NIST.IR.8338.

[WP16] D. Wübben and H. Paul, "Analysis of virtualized Turbo-decoder implementation for Cloud-RAN systems", International Symposium on Turbo Codes and Iterative Information Processing (ISTC), volume 9, pp. 385–389, IEEE, Brest, France, September 2016. DOI:10.1109/ISTC.2016.7593142.

[WRB+14] D. Wübben, P. Rost, J. S. Bartelt, M. Lalam, V. Savin, M. Gorgoglione, A. Dekorsy, and G. Fettweis, “Benefits and Impact of Cloud Computing on 5G Signal Processing: Flexible centralization through cloud-RAN”, IEEE Signal Processing Magazine, volume 31, no. 6, pp. 35–44, November 2014. DOI:10.1109/MSP.2014.2334952.

[XCC15] J. Xu, T. Che, and G. Choi, “XJ-BP: Express journey belief propagation decoding for polar codes”, Global Communications Conference (GLOBECOM), IEEE, San Diego, CA, USA, December 2015. DOI:10.1109/GLOCOM.2014.7417316.

[XWXG19] Y. Xu, W. Wang, Z. Xu, and X. Gao, “AVX-512 Based Software Decoding for 5G LDPC Codes”, International Workshop on Signal Processing Systems (SiPS), pp. 54–59, IEEE, Nanjing, China, October 2019. DOI:10.1109/SiPS47522.2019.9020587.

[ZM09] M. Zivkovic and R. Mathar, “Preamble-Based SNR Estimation in Frequency Selective Channels for Wireless OFDM Systems”, Vehicular Technology Conference (VTC Spring), volume 69, pp. 1–5, IEEE, Barcelona, Spain, April 2009. DOI:10.1109/VETECS.2009.5073813.

[ZZW+14] L. Zhang, Z. Zhang, X. Wang, Q. Yu, and Y. Chen, “On the puncturing patterns for punctured polar codes”, International Symposium on Information Theory (ISIT), IEEE, Honolulu, HI, USA, July 2014. DOI:10.1109/ISIT.2014.6874807.