FHV
Vorarlberg University
of Applied Sciences

Machine to Machine communication via cellular network

Implementation and evaluation of communication over a 5G standalone private network in an industrial use case focusing on the data exchange between two articulated robots

Master Thesis
submitted in fulfillment of the Degree

Master of Science in Engineering (MSc)

University of Applied Sciences Vorarlberg
Mechatronics

Submitted to
Prof. (FH) DI Dr. Robert Merz

Handed in by
Michael Sonnberger
Dornbirn, August 2023

Abstract

Implementation and evaluation of communication over a 5G standalone private network in an industrial use case focusing on the data exchange between two articulated robots

This thesis focuses on implementing and testing communication over a private 5G standalone network in an industrial environment, with a specific emphasis on communication between two articulated robots. The main objective is to examine machine-to-machine communication behavior in various test scenarios. Initially, the 5G core and radio access network components are described, along with their associated interfaces, to establish foundational knowledge. Subsequently, a use case involving two articulated robots is implemented, and essential metrics are defined for testing, including round-trip time, packet and inter-packet delay, and packet error rate.

The tests investigate the impact of 5G quality of service, packet size, and transmission interval on communication between the robots, focusing on the effects of network traffic. The results highlight the significance of prioritizing network resources based on the assigned quality of service identifier (5QI), demonstrate the influence of packet sizes on communication performance, and underscore the importance of transmission intervals for automation purposes.

Additionally, the study examines how network disturbances influence the movements of a robot controlled via 5G, establishing a direct relationship between network metrics and the resulting deviations in the robot's trajectory. The work concludes that while machine-to-machine communication can be successfully implemented with 5G SA, tradeoffs must be carefully considered, especially concerning packet error rate, and emphasizes the importance of understanding the required resources before implementation to ensure feasibility.

Future research directions include investigating network slicing, secure remote control of robots, and exploring the use of higher frequency bands. The study highlights the significance of aligning theoretical standards with practical implementation options in the evolving landscape of 5G Networks.

Kurzreferat

Implementierung und Evaluierung einer Kommunikation mittels privatem 5G-Standalone-Netz in einem industriellen Anwendungsfalls mit Fokus auf die Datenübertragung zwischen zwei Knickarmrobotern

Diese Arbeit konzentriert sich auf die Implementierung und Testung eines privaten 5G-Standalone-Netzes in einem industriellen Umfeld. Der Schwerpunkt liegt auf der Maschine-zu-Maschine-Kommunikation und deren Verhalten in verschiedenen Testszenarien. Zunächst werden der 5G Core, das Radio Access Network und die zugehörigen Schnittstellen beschrieben, um grundlegende Kenntnisse aufzubauen. Ein Anwendungsfall wird mithilfe von zwei Knickarmrobotern implementiert, und die relevanten Metriken werden definiert, um darauf basierend die Round-Trip-Zeit, die Paket- und Inter-Paket-Verzögerungsänderung sowie die Paketfehlerrate in einer praxisnahen Umgebung zu messen.

Dabei werden die Auswirkungen von 5G quality of services, Paketgröße und Übertragungsintervall auf die Kommunikation zwischen den Robotern untersucht, wobei besonderes Augenmerk auf den Einfluss des Netzwerkverkehrs gelegt wird. Die Ergebnisse betonen die Bedeutung der Priorisierung von Netzwerkressourcen anhand der zugewiesenen Quality-of-Service-Identifier (5QI), beschreiben den Einfluss der Paketgröße auf die Kommunikation und verdeutlichen die Wichtigkeit des Übertragungsintervalls für die Automatisierung.

Darüber hinaus werden die Auswirkungen von Netzwerkstörungen auf die Bewegungen eines über 5G gesteuerten Roboters untersucht und der direkte Zusammenhang zwischen Netzwerkmetriken und daraus resultierenden Abweichungen der Robotertrajektorie beschrieben. Die Arbeit schlussfolgert, dass die Maschine-zu-Maschine Kommunikation zwar mit 5G-SA umgesetzt werden kann, jedoch Kompromisse eingegangen werden müssen, insbesondere hinsichtlich der Paketfehlerrate. Zudem wird auf die Bedeutung einer umfassenden Ressourcenanalyse vor der Implementierung zur sicherstellung der Machbarkeit hingewiesen.

Zukünftige Forschungsrichtungen umfassen die Untersuchung von Netzwerk-Slicing, die sichere Fernsteuerung von Robotern und die Nutzung höherer Frequenzbänder. Die Arbeit unterstreicht zudem die Notwendigkeit, theoretische Standards mit praktischen Implementierungsoptionen in der sich entwickelnden Landschaft der 5G-Netze abzugleichen.

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	Third Generation Partnership Project
5G NR	Radio Access Network
5G NR	5G New Radio
5GC	5G Core
5GS	5G System
5QI	5G Quality-of-Service Identifier
AF	Application Function
AMF	Access and Mobility Management Function
AS	Access Stratum
AUSF	Authentication Server Function
CDF	Cumulative Distribution Function
CP	Control Plane
CUPS	Control and User Plane Separation
DN	Data Network
DNN	Data Network Name
ECDF	Empirical Cumulative Distribution Function
EPC	Evolved Packet Core
F1AP	5G NR Layer 1 (L1) Application Protocol
FDD	Frequency Division Duplex
FDM	Frequency Division Multiplexing

FFT	Fast Fourier Transform
GBR	Guaranteed Bit Rate
gNB	Next-Generation NodeB (base station)
gNB.DU	gNB Distributed Unit
gNB-CU	gNB Centralized Unit
GTP	GPRS Tunneling Protocol
GTP-U	GTP User Plane
ICI	Inter-Channel Interference
IFFT	Inverse Fast Fourier Transform
IPDV	Inter-Packet Delay Variation
IQR	Interquartile Range
ISI	Intersymbol Interference
LTE	Long-Term Evolution
MAC	Medium Access Control
MIB	Master Information Block
MIMO	Multiple-Input Multiple-Output
mMIMO	Massive MIMO
MSE	Mean Squared Error
NAS	Non-Access Stratum
NEF	Network Exposure Function
NF	Network Function
NRF	Network Repository Function
NSA	Non-Standalone Access
NSSF	Network Slice Selection Function

OFDM	Orthogonal Frequency Division Multiplexing
PCF	Policy Control Function
PDCP	Packet Data Convergence Protocol
PDSCH	Physical Downlink Shared Channel
PDU	Protocol Data Unit
PDV	Packet Delay Variation
PER	Packet Error Rate
PLMN	Public Land Mobile Network
QAM	Quadrature Amplitude Modulation
RAN	Radio Access Network
RLC	Radio Link Control
RRC	Radio Resource Control
RTDE	Real-Time Data Exchange
RTT	Round Trip Time
SA	Standalone Access
SBA	Service-Based Architecture
SCTP	Stream Control Transmission Protocol
SDAP	Service Data Adaptation Protocol
SIB	System Information Block
SMF	Session Management Function
S-NSSAI	Single Network Slice Selection Assistance Information
SSC	Security and Support Capability
SST	Service Set Type
SUCI	Subscription Concealed Identifier

SUPI	Subscription Permanent Identifier
TDD	Time Division Duplex
UDM	Unified Data Management
UDR	Unified Data Repository
UE	User Equipment
UP	User Plane
UPF	User Plane Function

1 Introduction

Since 1998 the 3rd Generation Partnership Project (3GPP), an association of different standardization institutions Worldwide, is the driving force when it comes to specifications for cellular telecommunication. They started during the era of GSM with the so-called Release 99 and grew in importance during the standardization of LTE, with Release 8 beginning in 2005. They are now crucially involved in the characterization and normalization of mobile communication. At the end of 2018, 3GPP released with its Release 15, the first version of the then-new cellular standard called 5G. Since then, 3GPP has been working on further improvement and development of 5G, and according to their Work Plan, Release 18 will be finished in mid-2024. [1]

But why was a technology like 5G needed? What was the driving idea behind that?

Technologies like augmented or virtual reality need large bandwidth to guarantee the user experience. The Internet of Things is growing fast and needs a backbone communication Network that can shoulder a massive number of devices connected to it and critical applications like remote surgeries need to be very reliable and need a very low End-to-End Latency. All these different application requirements led to the conclusion that if there should be one system that would fulfill all the criteria above, a completely new approach was needed. Away from the existing Evolved Packet Core (EPC) of LTE, which was developed for voice services and mobile internet and was built on dedicated Hardware, and towards a highly flexible virtualized architecture that could be scaled and configured to specific use cases. These new requirements made it also necessary that the Radio Access Network (RAN) needed an overhaul, and that lead to the development of the 5G Core (5GC) System and the 5G New Radio (5G NR) Access Network.

This new approach created a constantly evolving cellular network that steadily grows in complexity and capability. One of the goals of 5G is to provide network resources according to specific use cases, i.e., ultra-reliable low-latency, or massive machine-type communication. The network configuration allows multiple subscribers to use the same medium but under different conditions. These newly acquired configuration capabilities need to be tested in real setups. Particularly, application-oriented tests are necessary to determine the feasibility and quality of a potential implementation. This thesis aims to build upon and attempt to showcase the current possibilities.

1.1 Context of Work

At the beginning of this thesis, the architecture, components, and procedures of 5G are in the focus. The 5G standard is continuously maintained by 3GPP. However, since these technical specifications describe how such a system functions in detail, it was necessary to abstract and summarize essential knowledge for this work. An attempt was made to generate this knowledge directly from the standard, but literature was also used, which had already abstracted it to some extent. In particular, references were made to [1] and [2].

To create a test environment that is both flexible enough to encompass multiple test scenarios and as close to reality as possible, a use case was developed and implemented. For this purpose, two articulated robots were used to perform synchronous movements. The first robot initiates a motion and sends its axis data over a network to the second robot, which follows the movement based on the received information. To integrate this with the 5G network, the robots were connected to the network using 5G modems. This allows us to identify which and how data is sent in a machine-to-machine communication setup. This information was used to implement different network tests and test scenarios to observe the network behavior under certain conditions.

The metrics focused on include the round-trip time, which is the time a packet takes to travel from a sender to a receiver and back. The variation in this time and the number of lost packets also play a crucial role. The results are evaluated using statistical methods, presented graphically, and explained.

The last part of this thesis presents the outcome and summarizes the results.

In [3], similar approaches were used to test a 5G Network. The behavior of the network when using different Packet sizes and transmission intervals were the main focus of this study and laid part of the foundation for this thesis. Additionally, the setup in [4] bears similarities to the one used for this thesis, therefore the results are comparable. Both studies investigated the behavior of data packets in a 5G network. They used the packet delay between sender and receiver, as well as changes in delay and packet loss, as crucial metrics. The connection between robot movement and network metrics was introduced in [5], where the test setup was similar but no detailed analysis on the influence of traffic was made.

This thesis adds the investigation of different Quality of Services and the behavior of connections under network congestion. Additionally, the influence of network disturbances on the trajectory of a remotely controlled robot will be investigated using the aforementioned use case. This adds an industrial application element that was either missing in previous studies or not examined in such detail and results in the following research question:

"How do round-trip time, delay variation, and error rate behave in a 5G standalone private network in a realistic environment where the network is operating close to its full capacity, and how are these metrics affected in a machine-to-machine communication when different qualities of service, packet sizes, and transmission intervals are used?"

2 5G Basics and Functionality

This Chapter describes the main features introduced in 5G concerning LTE, explains what a 5G core architecture looks like, the aerial interface works, communication is established, and data travels from one endpoint to the other. Furthermore, this chapter outlines some key technologies that make 5G superior to its predecessors and provides an overview of this communication standard.

The information in this chapter refers only to the 3GPP Release 15 of 5G because the system used during this thesis is based mostly on this release. Newer release versions not yet available for commercial use will be implemented via a software update when provided. Therefore, this thesis points out when technologies in this setup refer to a different Release than Version 15.

The 5G Network can be split into two main Domains:

- The Radio Access Network (RAN), also called 5G New Radio (5G NR), consists of multiple access point cells called gNodeB and is responsible for connecting the user equipment (UE) with the 5G network. Further details are provided in Chapter 2.2,
- The 5G Core (5GC) is a virtualized environment that offers different network functions (NFs). The 5GC is responsible, among other things, for routing data to specific endpoints, handling control and user data from and to the RAN, and providing different possibilities depending on the NFs available. Further details are provided in Chapters 2.1 and 2.3.

This split was necessary because 5G was not developed to replace 4G (LTE) from the beginning solely but to coexist as a first step. Therefore, flexible standardization and implementation were needed that allowed the mobile network providers to use already existing infrastructure to implement 5G gradually. The solution was to launch two different setups called None Stand Alone (NSA), which is a combined setup with a 4G System, and Stand Alone (SA), which is set up with only 5G components:

- None Stand Alone (NSA): The idea was to make the first implementation easier and quicker. Therefore, the 5G NR works alongside the LTE RAN and uses that as a so-called communications anchor. There are different options for how an NSA mode could be executed, but the most used version for already existing networks is 3X, shown in Fig. 2.1 on the left side, where the 5G and 4G RAN connect to an evolved packet core (EPC) used in 4G. The solid lines represent data paths, and the dotted lines show the control signal path. This is easy to implement, and the update to a 5GC is possible in the future. For newly built NSA networks, the version 7X shown in Fig. 2.1 on the right side, where the 5G and 4G RAN connect to a 5GC, makes the most sense.

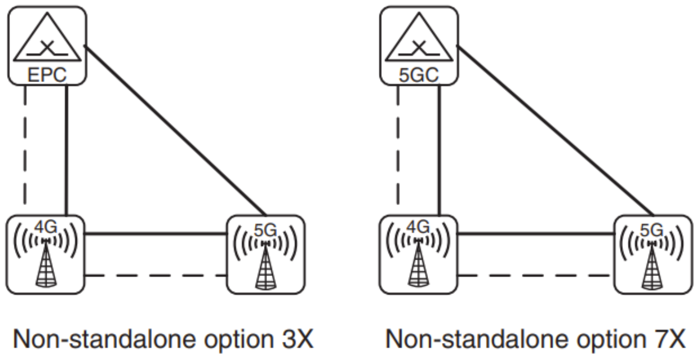
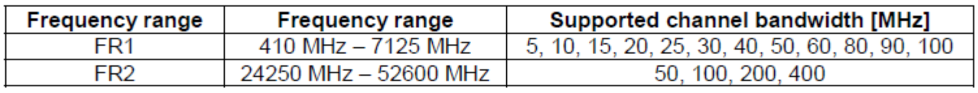


Figure 2.1: NSA Versions 3X where the network relies on an EPC and Version 7X where a 5GC is in use [1, S. 29]

- Stand Alone (SA): This implementation relies only on the 5G technologies and consists of a 5GC and a 5G RAN. This setup allows optimized use of all functionalities that 5G has to offer since backward compatibility does not have to be considered. If no backward compatibility is needed, a 5G SA architecture should be preferred since the capabilities of 5G can only be reached in this setup. However, comprehensive deployment of SA in public networks is not possible due to the many non-5G capable devices still in use. Therefore, in an industrial environment in which a new 5G system is to be implemented and only 5G will be used, nothing speaks against a SA setup. [1] [2]

Since the system under test is a 5G stand-alone system, this thesis will focus only on the 5G SA setup. That means all the following chapters always refer to a SA topology unless otherwise stated, and for more information about NSA, please refer to [1] [2]

Another difference between 5G and its predecessors is the frequency spectrum, which consists of a sub-6-GHz band and a mmWave band, as shown by the two frequency ranges in Fig. 2.3. This allows higher data rates in FR2 and ensures the mobile coverage of lower frequencies with FR1, the band used by LTE systems. [6]



Frequency range	Frequency range	Supported channel bandwidth [MHz]
FR1	410 MHz – 7125 MHz	5, 10, 15, 20, 25, 30, 40, 50, 60, 80, 90, 100
FR2	24250 MHz – 52600 MHz	50, 100, 200, 400

Figure 2.2: 5G frequency range FR1 and FR2 with supported channel bandwidth [6, S. 25]

Furthermore, the numerology of 5G allows a flexible frame structure that enables communication with lower latency. That is described in detail in Chapter 2.2

2.1 System Architecture

Fig. 2.3 shows what a simplified 5G system architecture looks like. The UE connects to the RAN, which consists of multiple radio cells. The RAN is controlled by the 5G Core, in particular from the control plane (CP) and its functions. The focus of the CP is to provide a centralized maintenance point and flexibility in terms availability of different NFs. On the other hand, the user plane (UP) goal is to provide the bandwidth and latency for data connections between users or via an external data network (DN). Therefore, the UP can be located closer to where communication services are provided and reduce transport costs. In addition, the control/user plane separation (CUPS) makes sense because of their different responsibilities, which were already used in LTE. [1]

One of the main differences between 5G and previous technologies is the step away from dedicated hardware and towards using standard hardware components and running a virtualized core as software on those components. The CP was realized as a service-based architecture (SBA) and allows implementing or updating NFs independently from the rest of the system. Furthermore, SBA allows the use of all provided functions by every other NF, the multi-existence of NFs, and

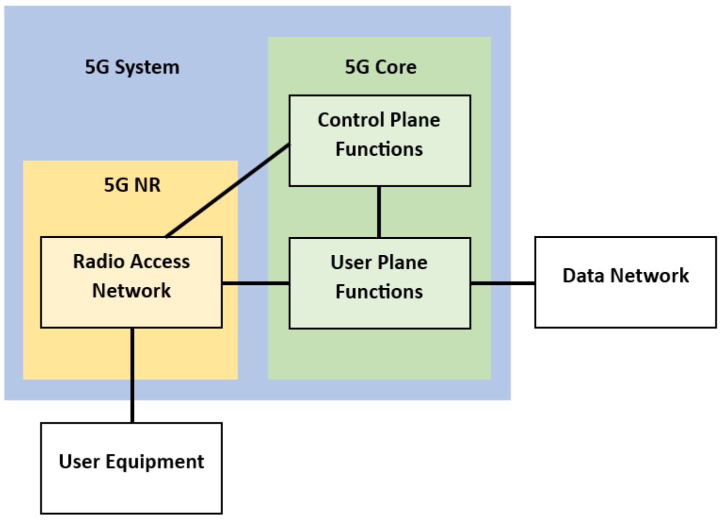


Figure 2.3: 5G system overview with UE and DN, separation of the RAN and the core as well as CP and UP (Adapted from [7, S. 22])

the independence of every NF, which means that no other service should impact this NF.

Another advantage is the connection of the different NFs over one common network via HTTP/2 and JSON. That enables all NFs with a 3GPP-compliant service-based interface to connect seamlessly to the core network and makes it easier to develop and manage those. [8]

2.2 Radio Access Network

The job of the RAN is to connect the UE with the 5GC and the DN. Therefore, the RAN contains one or multiple gNBs and operates three major interfaces:

- the aerial interface which connects UEs with gNBs via access stratum (AS) and UEs with the 5GC via non-access stratum (NAS),
- the NG interface which connects the gNBs with a 5GC,
- the Xn interface which connects a gNB with a neighbor gNB.

Communication that is restricted exclusively between a gNB and a UE is carried out with the access stratum (AS) layer. That includes establishing and maintaining radio channels. Communication between the UE and the 5GC is carried out with the non-access stratum (NAS) layer and contains access and connection control, mobility, and session management.

In Fig. 2.4 the architecture of a NG-RAN is shown, whereas the single gNB can be further split up into central Unit (gNB-CU) and one or more distributed units (gNB-DU) connected via the F1 interface.
u

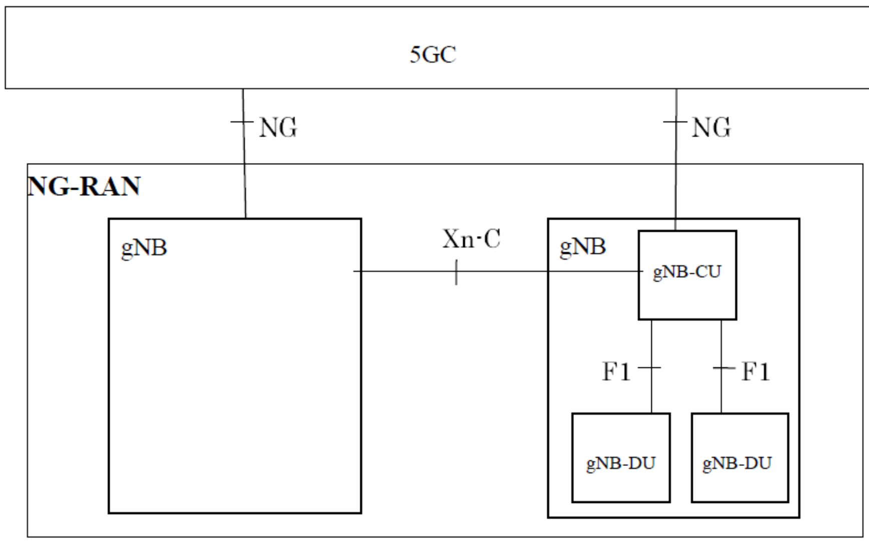


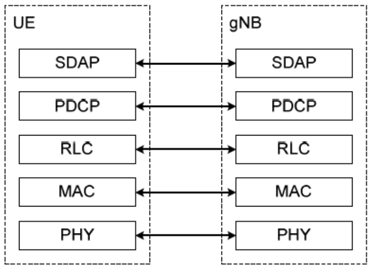
Figure 2.4: RAN architecture with the connection interfaces internal and to the 5GC [9, S. 10]

It’s essential to notice that the protocol stack of the NR RAN is also divided into user plane protocol stack and control plane protocol stack where layers one and two are similar and layer three only refers to the control plane protocol stack. In brief, the protocol stack looks like this:

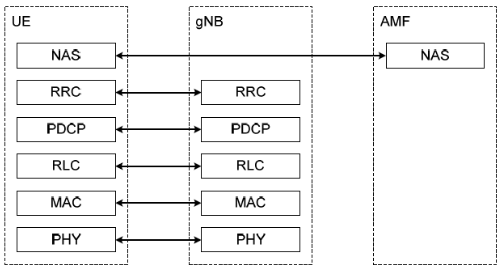
Layer 1 is the physical layer which contains modulation and demodulation of physical channels, synchronization beamforming, and error detection.

Layer 2 includes:

- Medium access control (MAC) responsible for priority handling, logical and transport channel mapping, and re-adjustment of physical beams,
- Radio link control (RLC) is responsible for the transfer of PDUs and error detection,



(a) UP protocol stack between UE and gNB [11, S. 16]



(b) CP protocol stack between UE and gNB [11, S. 17]

Figure 2.5: Protocol Stacks show that on the UP side, the UE connects only to the gNB while the CP connects the UE to the gNB and via NAS to the 5GC

- Packet data convergence Protocol (PDCP) responsible for user data forwarding and connection management,
- Service data adaptation protocol (SDAP) (only UP) responsible for forwarding and ciphering control plane data.

Layer 3 The radio resource control (RRC) (only CP) layer sends NAS and AS information and is responsible for RRC connection and radio bearers, key management, UE measurement data, and NAS messaging to UE. [10] [11]

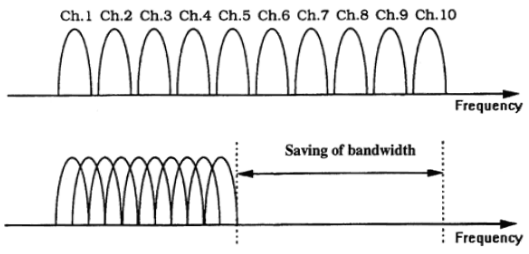
This is visualized in Fig. 2.5a and Fig. 2.5b with the overlaying NAS protocol on the control plane protocol stack.

2.2.1 Aerial Interface

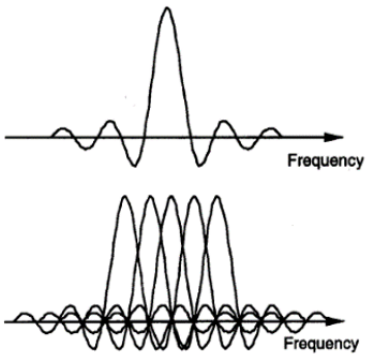
The aerial interface of a 5G-RAN is more complex than in other cellular technologies because of two major differences:

- The frequency spectrum of 5G is very broad since FR1 is from 410 to 7125 MHz and FR2 is from 24250 - 52600 MHz. That means the aerial interface must be able to provide the technical capabilities to cover that range.
- The goal of 5G is to provide profiles with different communication characteristics for specific use cases. Therefore, the aerial interface must be flexible and configurable.

Therefore, looking at the implemented technologies in more detail is necessary to understand how that flexibility was created.



(a) OFDM bandwidth reduction in comparison to conventional multi-carrier systems [13, S. 22]



(b) OFDM subchannel and signal in frequency domain to show the necessary orthogonality [13, S. 23]

Figure 2.6: Illustration of the advantages of OFDM and the orthogonality required to take advantage of them.

First, in 5G time division duplex (TDD) and frequency division duplex (FDD) is used. In TDD the uplink and downlink are transmitted on the same carrier frequency but divided into timeslots. In FDD two different frequencies are used to transmit uplink and downlink. In FDD, small frequency shifts between subcarriers lead to intercarrier interference (ICI). Therefore it is common in 5G to use a TDD signal between the uplink and downlink channel of an FDD signal for efficient use of the spectral resources. [12] [1]

As modulation method is quadrature amplitude modulation (QAM) in use. The specification of 5G ranges from 4-QAM up to 256-QAM, dependent on the possible signal quality between UE and gNB. [11]

5G uses frequency-division multiplexing (FDM) to reach high data rates by given bandwidths. Therefore, a given channel bandwidth is divided into sub-channels with sub-carriers. A specific variant of FDM, orthogonal frequency-division multiplexing (OFDM), is used, where these sub-carriers must be orthogonal functions, so they don't interfere with their neighbor sub-channels as shown in Fig. 2.6. That allows the reduction of the sub-carrier space and to transmit more sub-carriers in a given bandwidth as shown in Fig. 2.6a

Fig. 2.7 shows how the data processing for transmission and reception is done. In the beginning, the binary data gets parallelized and QAM encoded. The parallelized data can be viewed as if it were in the frequency domain and will then be transformed mathematically to the time domain by an inverse fast Fourier transformation (IFFT). A cyclic prefix is added, where a part of the end of every transmission symbol is copied into the guard interval at the beginning of each symbol to preserve orthogonality and reduce Inter Symbol Interference. After the signal is serialized, it is transmitted through a DAC. On the receiving side, an ADC converts the data back and the prefix is removed. fast Fourier transformation (FFT) converts the combined signal into the different sub-signals which are then filtered to reduce inter symbol (ISI) and inter carrier interference (ICI). The last step is to decode the QAM-modulated signal into the original data. [14] [15]

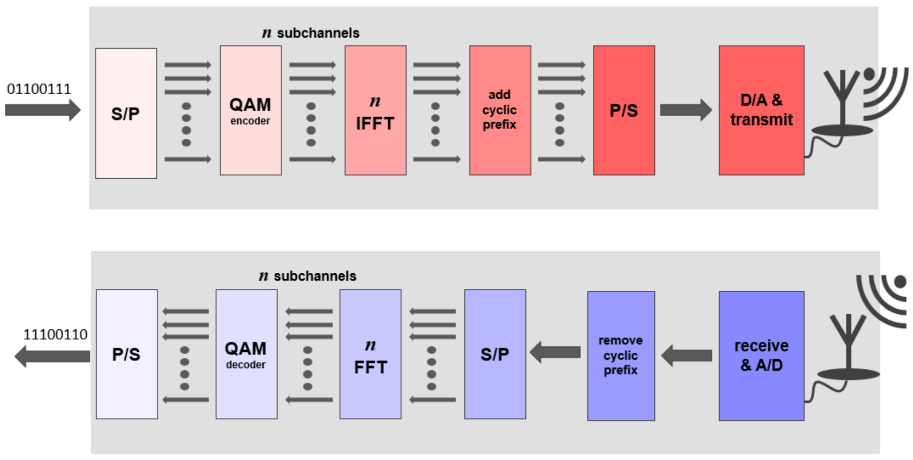


Figure 2.7: Decoding and encoding of a binary signal using OFDM (adapted from [15, S. 121])

The difference between LTE to 5G is that, while in LTE the space between the sub-carriers was fixed to 15 kHz, in 5G the sub-carrier spacing is flexible. The sub-carrier channel bandwidth can be changed between 15, 30, 60, 120, or 240 kHz which enables faster data rates with the trade-off for higher bandwidth consumed. Those different sub-spaces are written down in the so-called "Numerology" of 5G. [11]

Since the 15 kHz channel bandwidth in LTE was used to minimize ISI which occurs because of time delay spread in multipath channels [16], the higher frequencies in 5G make it possible to change sub-channel bandwidth without increasing ISI. [2] [17]

Another advantage is that in 5G the broadcast and synchronization information do not have to be at a specific position on a channel but can be at any position. This allows it to use different Numerologies and therefore different QoS profiles on the same channel. [2]

Multiple input multiple output (MIMO) was already used in LTE to send i.e. two data streams on the same channel from two antennas on the eNB to two antennas on the UE to increase the data rate or improve connection quality. A maximum of eight data streams was possible in LTE but typically four were used. [1] In 5G the term massive MIMO (mMIMO) is used, whereas "massive" stands for more than eight but typically are phased-array-antennas with 32 or 64 elements. These antenna arrays can be used the same way as in LTE but in addition, they allow beamforming, which enables the manipulation of beam characteristics to concentrate the transmitting power on certain points in an area. Fig. 2.8 shows how such beams could look for an array with 64 elements. [12] [16]

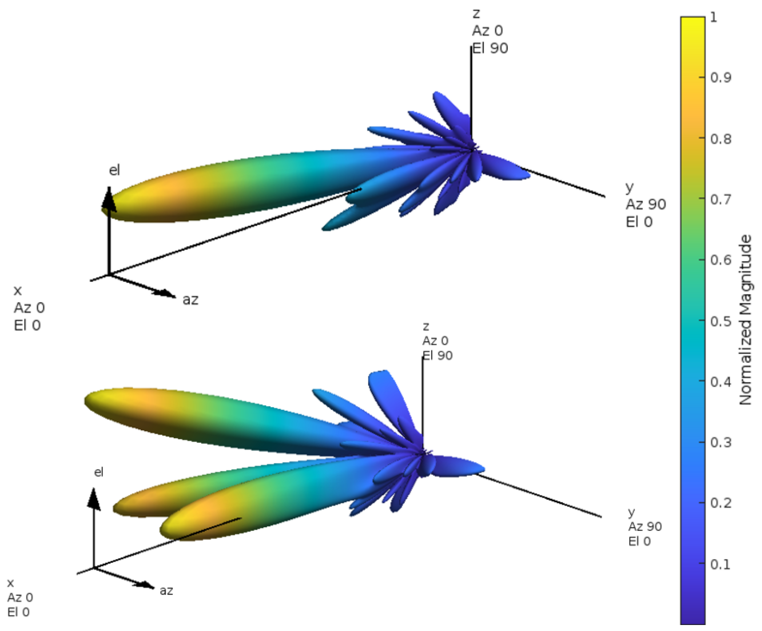


Figure 2.8: Beamforming with a phased array transmitter showing one beam and three beams; simulated in Matlab [18]

The readjustment of a beam is done on the MAC layer after the first signaling is done on the RRC layer. This allows quick adjusting via communication between UE and RAN whereas the UE analyzes the incoming signal and sends the needed adjustments. [2]

2.2.2 NG Interface

The NG interface works as a logical point-to-point interface and separates the RAN and the 5GC. It supports CUPS and therefore is divided into CP and UP. The CP side transfers NAS signaling between UE and the Access Management Function (AMF, described in chapter 2.3) of the 5GC and control and configuration messages between RAN and 5GC, i.e. mobility management procedures. The NGAP protocol is used on top of an IP and Stream Control Transmission Protocol (SCTP) [19] layer and some elementary procedures of NGAP are:

- RAN configuration updates,
- Handover management,
- Initial context and Packet Data Unit (PDU) setup and modification,
- Paging.

The UP side uses UDP packets with the GPRS Tunneling Protocol (GTP-U) to send user data between the RAN and the User Plane Function (UPF, described in chapter 2.3) of the 5GC. [20]

2.2.3 Xn Interface

The Xn interface is between two gNBs and has a similar protocol structure to the NG interface. It also supports CUPS and therefore can be separated into control and user plane side. The Xn interfaces tasks are mobility and connectivity management between gNB on the CP side and flow control and user data forwarding on the UP side. [11]

2.2.4 F1 Interface

The F1 interface is between gNB-CU and the gNB-DU and moves data from the RLC to the PDCP sublayer in layer 2. Its task is to converge data and to split radio network and transport network for i.e. higher compatibility regarding different vendors and therefore uses the F1 application protocol (F1AP). [21]

2.3 Core Functions

In this chapter, the core functions of the 5GC will be explained, regarding their tasks, their communication with each other, and how they enable a 5G System to transport data from one endpoint to another. It must be mentioned that not all core functions will be explained in detail since it is not necessary within the framework of this thesis. Functions that are only described superficially will be referenced for more detailed information.

The 5G Core contains several Network Functions (NFs) that work together to create a scalable, highly flexible, and compatible communication environment for control and user data. The UP contains the user plane function and is therefore the main element for user data forwarding.

The CP contains several NFs that have their own tasks in the overall picture of the system. In Fig. 2.9 is the core architecture shown, with the separation between UP, CP, RAN, and external DN and the connecting interface N1-N4, N6, and N9. The

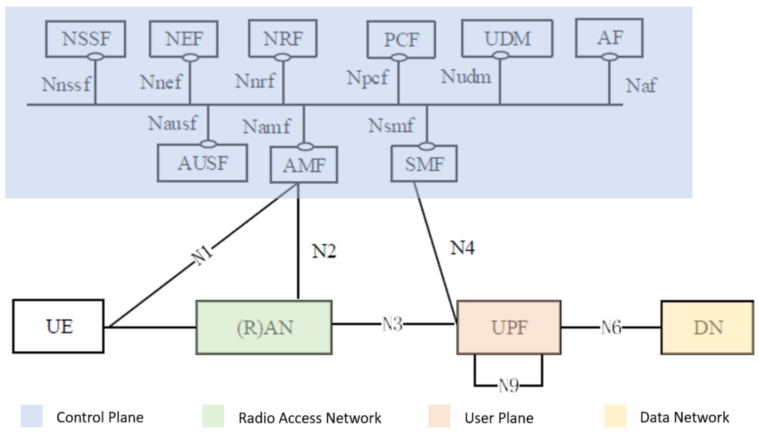


Figure 2.9: 5G non-roaming System architecture [7, S. 22]

NFs shown in Fig. 2.9 are the basic set of a 5GC and will be explained in detail.

2.3.1 User Plane Function

The UPFs main task is to handle user traffic. Therefore, one or more UPFs can be deployed and then controlled and configured by the CP via the N4 interface which connects the UPF with the Session Management Function (SMF). In LTE one had the option of three predefined capability sets that were not changeable. Now in 5G, the UPFs are configurable by the CP to increase flexibility and give the System much more possibilities in terms of how user data is transported from one end-point to the other, whether this means sending user data from one UE to the other on the same UPF, a different UPF but in the same 5G system though the N9 interface or sending data through the N6 gateway to another network. [1]

To ensure this, the UPF has some or all of the following functionalities implemented:

- it is the anchor point for communication mobility, which means data from a UE flows through at least one UPF before exiting on another UPF or entering the DN,
- routing data from one endpoint over the network to the other endpoint based on the configuration given by the SMF,
- handles the QoS flow of the UP and ensures the UL/DL rates as well as traffic latency,
- monitoring available resources and events and reporting them to the CP,
- guaranteeing data integrity during handover,
- has implemented functions to respond to ARP or NDP requests. [1] [7]

2.3.2 Access and Mobility Management Function

The AMF is connected to a UE via the N1 interface and to the RAN via N2. The N1 interface is directly routed to a UE over NAS signaling for procedures that relate to a specific UE and the N2 interface is used to send information with NGAP for procedures related to the RAN. Therefore the main task of the AMF is controlling the communication between RAN and UE and providing several access and mobility-related functionalities that include the following:

- managing UE authentication, authorization, and registration together with the AUSF and therefore taking care of mobility management notifications, like handovers,
- managing the UE signaling connection, i.e changing a UE from idle to connected mode, via NAS messaging,
- transport SMS, Public Warning System, and Location Service messages to the UE,
- providing UE access security management. [1] [7]

2.3.3 Session Management Function

The SMF is the controlling instance of one or more UPFs and therefore connected to them via the N4 interface. That means the SMF controls how data is sent through the network, establishing tunnels for user data between the RAN and UPFs and modifying and releasing the end-to-end connection. The following functionalities are supported:

- IP address management and DHCP functions,
- configuration of UPF regarding traffic steering and routing,
- enforces QoS according to the given configuration for the specified UEs,
- collecting data for charging functions,
- session and service continuity (SSC) mode selection, ([7, S. 90-92])
- roaming functions. [1] [7]

2.3.4 Further Network Functions

To enable the multiple functions implemented in a 5GC further NFs are needed. These provide additional functionalities in terms of usability, security and compatibility and the most important of them are:

- **Network Slice Selection Function (NSSF)**
  The NSSF is the central instance of a public land mobile network (PLMN) regarding allowed and configured network slice information, mapping network slices of different UEs, and determining the AMF instance for a specific UE. [7]

- **Network Exposure Function (NEF)**
  The NEF securely receives and provides information from AFs or internal NFs according to the network policy and stores the information as structured data in the UDR. [7]

- **Authentication Server Function (AUSF)**
  The AUSF provides security keys and authentication procedures between a UE and the network. In addition, the AUSF communicates with the UDM about subscriber authentication and handles the subscription permanent identifier (SUPI) and subscription concealed identifier (SUCI) provision to PLMNs. [1] [22]

- **Network Repository Function (NRF)**
  The NRF manages the information of available NFs, their instances, addresses, supported services, and attributes and provides them to all NFs in the Network. [7]

- **Policy Control Function (PCF)**
  The PCF provides or pushes policies to NFs. The communication with the SMF relates to PDU forwarding, QoS, charging, and local traffic. The communication with the AMF relates to network selection and restrictions in terms of available networks in certain areas. [1] [23]

- **Unified Data Management (UDM)**
  The UDM handles the UE identification and subscription data, i.e. the SUPI, roaming credentials, serving AMFs and SMFs, and is always located in the UEs home PLMN. [7] [1]

- **Application Function (AF)**
  AFs are not part of the 5GC but are external applications concerning policy control or the influence of certain applications on data traffic. The AF can either communicate to 5GC NFs directly if allowed or via NEF. [7]

The communication between the CP functions is shown in Fig. 2.9 as one common network with which all CP NFs are connected. The NFs provide certain specified services via their service-based interface (i.e., Namf stands for the network service-based interface of the AMF) for other NFs to consume. The communication is done via TCP over HTTP/2 with JSON for serialization to ensure high flexibility and easy integration. [7] [1]

2.4 Network Procedures and data flow

When talking about establishing data connection in 5G the term Protocol Data Unit Session (PDU Session) is used, where the PDU includes user data and all the protocol overhead needed for the specific network layer where the data is transmitted. In this chapter, the connection setup between a UE and a 5G Network, different communication procedures and the transmission of PDUs from endpoint to endpoint are considered. Since not every procedure is relevant in the context of this work, the focus is placed on procedures that are relevant to this thesis; therefore, roaming will not be discussed.

2.4.1 Connection establishment

The first step of connection establishment is using the correct PLMN and data network name (DNN) of the network the UE wants to connect. To make the 5G network visible to the UE the RAN sends out periodical broadcast signals where one of which is called the master information block (MIB). The MIB is sent every 40 ms and contains basic information about the network and how the following signal information blocks (SIB) 1 to 26 are transmitted. In contrast, not every SIB has to be transmitted depending on the network. The SIBs after SIB 1 are not sent as a broadcast anymore and are transmitted via a physical downlink shared channel (PDSCH). These SIBs contain all the information needed for a UE to establish a registration request via NAS signaling with the designated gNB and the 5GC. A suited AMF is selected, sends an identity request, and gets a response from the UE in the form of the SUCI. After that, the AMF starts the authentication procedure with the AUSF which checks if the subscriber data stored in the UDM is valid. The AMF and UDM exchange UE context data and subscriber data. If the UE can connect to the network, the AMF finishes the registration procedure shown in Fig. 2.10 with a registration acceptance. [1] [24]

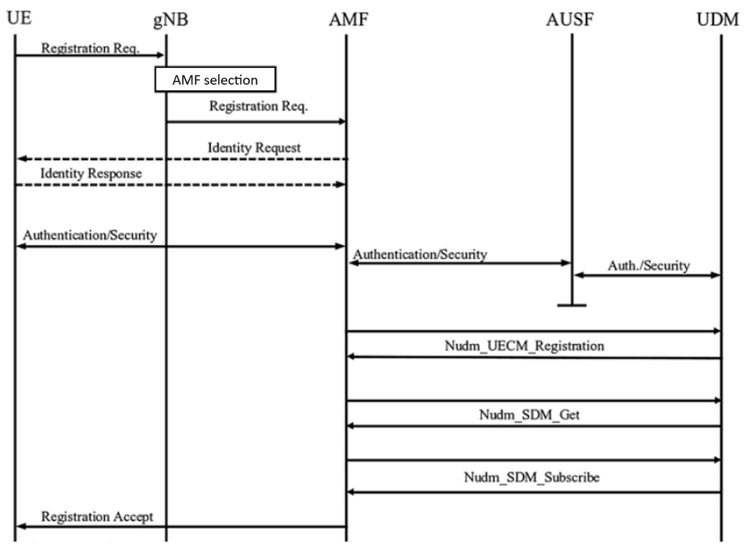


Figure 2.10: Registration procedure of a UE with a 5G network [2, S. 182]

After the UE is registered it requests a PDU Session as illustrated in Fig. 2.11 Establishing a PDU session means creating an IP tunnel where user data can be sent between two endpoints. However, the IP tunnel endpoint is not the UE but the gNB and the responsible UPF to which the UE is connected. That allows an easier change of the data flow when the UE changes its location. The SMF controls the establishment of such data sessions, and the UPF carries out the data forwarding. Therefore, the UE sends a session establishment request and the AMF then chooses a suitable SMF for the requested session. The chosen SMF retrieves the subscriber data from the UDM, responds to the AMF that the session can be handled, and selects a matching UPF concerning the session's purpose. The SMF instructs the UPF to establish an N4 session and the AMF to establish the N1 and N2 interfaces for messaging. An acknowledgment is sent from the SMF via AMF to the gNB which forwards a NAS message to the UE about the acceptance and establishes a new radio bearer connection with the UE. After that, the gNB responds and the AMF, SMF, and UPF update the session information. Now the tunnel is established, and the UE can send and receive data. [24] [2]

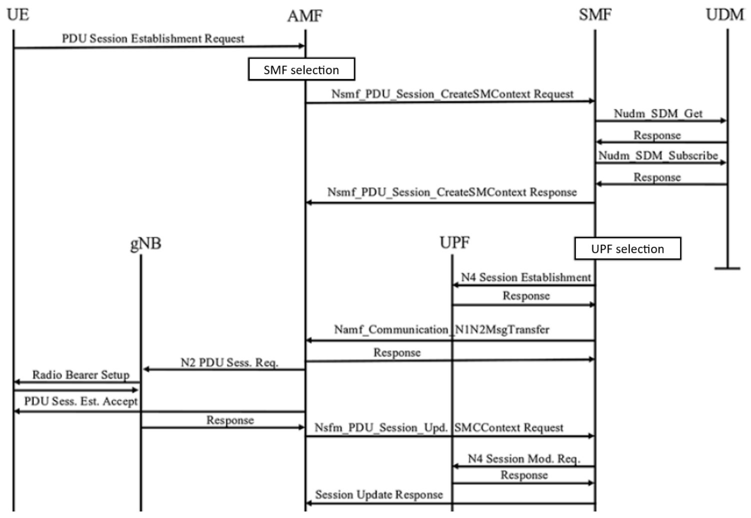


Figure 2.11: Session establishment of a UE with the 5G Network [2, S. 186]

For simplicity, the communication with the PCF is not explained in this chapter as is the connection of the UPF via the N6 interface with the data network. Further information regarding the connection establishment can be found in [24].

2.4.2 Session modification and release

A PDU session modification is necessary when one or more QoS parameters need to be changed either initiated by the UE or the 5GC. The SMF and the PCF initiate the update of the QoS parameters after checking that the changes are permitted. If the changes only regard the UPF then the SMF sends a modification request via the N4 interface, which occurs immediately. If the modification concerns UE or RAN, the AMF informs them through the connected interfaces and waits for a session modification acknowledgment. After that, the SMF requests the UPF to update the PDU session accordingly and initiates a policy modification with the PCF. [24]

When releasing a PDU session, unrelated to who initiated it, the SMF gets a release request from the initiator. The SMF then releases the PDU session-related IP address and UPF resources. Now there are two possibilities:

- UE unreachable:
  If the UE is unreachable the SMF notifies the AMF that the PDU session is released. The AMF then releases all associations with this PDU session.

- UE reachable:
  If the UE is reachable the SMF creates a release request that is sent to the UE and RAN via the N1 and N2 interface. The UE then acknowledges the release command as a NAS message via the AMF to the SMF. The further procedure is the same as if the UE is unreachable. [24]

2.4.3 Handover between neighboring gNBs

A UE constantly measures the connection quality to the currently connected gNB, also called the serving cell. Measurands for this are the received signal power, quality, and strength indication. If there is more than one gNB in the connection radius a decision must be made whether to stay connected with the current gNB or switch to a new one. If the connection to a new gNB is better, the UE sends a measurement report to its serving cell and starts a handover procedure as shown in Fig. 2.12. The serving cell sends the Handover request over the Xn interface to its neighbor gNB which will be the new destination for the PDU session endpoint. At first, all UE data will be routed through the Xn interface and the new gNB sends a path switch request to the AMF. The AMF communicates with the SMF and the UPF to modify the session and then change the IP address of the tunnel end point from the old to the new gNB. After that the old and the new gNB will be informed about the executed switch, the PDU session is routed over the UPF to the new serving cell, and the resources of the old gNB are released. [2]
Further procedure explanations and details can be found in 3GPP TS 23.502: "Procedures for the 5G System (5GS)". [24]

2.4.4 Data flow

For further explanation of how data is transported over a 5G Network the following three scenarios should visualize the possible data flows. Since roaming is not

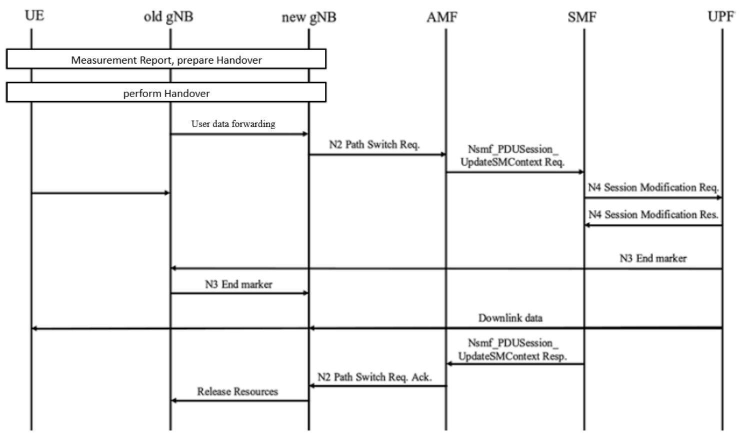

UE
old gNB
new gNB
AMF
SMF
UPF
Measurement Report, prepare Handover
perform Handover
User data forwarding
N2 Path Switch Req.
Nsmf_PDUSession_ UpdateSMContext Req.
N4 Session Modification Req.
N4 Session Modification Res.
N3 End marker
N3 End marker
Downlink data
Nsmf_PDUSession_ UpdateSMContext Resp.
N2 Path Switch Req. Ack.
Release Resources

Figure 2.12: Handover between neighboring gNBs through the Xn interface [2, S. 188]

in the scope of this thesis the following scenarios focus on connections where the UEs are in their Home PLMN.

In the first scenario in Fig. 2.13 shows a PDU session between a UE and the Data Network i.e., the internet. The UE sends the data utilizing the UP-protocol stack shown in 2.5a via the Radio interface to the gNB. The gNB wraps the packet in an IP layer and sends it via GPRS tunneling protocol (GTP) over the N3 interface to the UPF. The UPF changes the third layer and therefore the IP address to the corresponding N6 address and sends the packet to the DN over standard IP protocol. Incoming messages are similarly handled in the other direction.

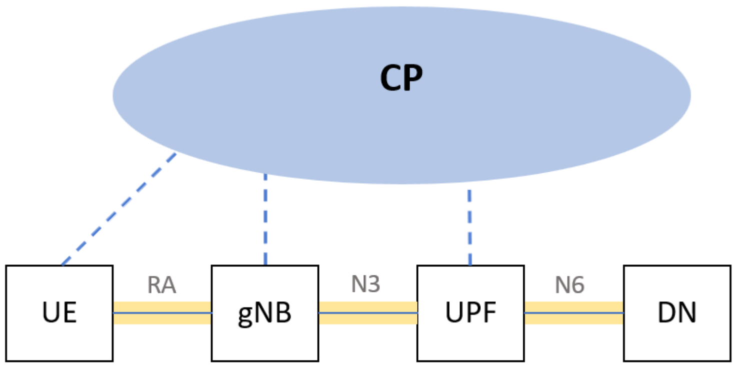
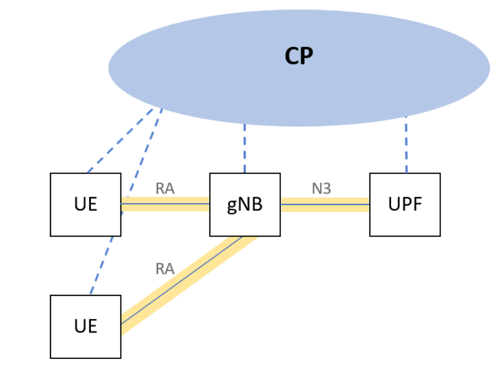
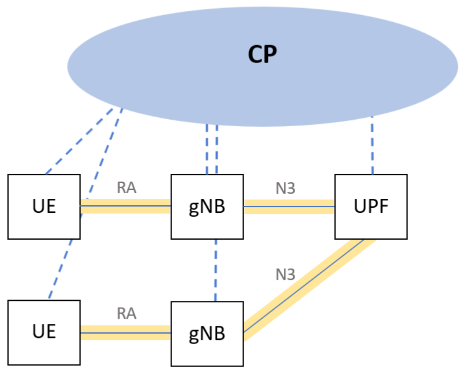


Figure 2.13: Dataflow between the UE and a Data Network with the corresponding interfaces

The second scenario explains a PDU session between two UE with the same QoS



(a) Dataflow between two UEs with the same UPF and serving cell



(b) Dataflow between two UEs with the same UPF but a different serving cell

Figure 2.14: Dataflow between two UEs with similar QoS and location

demands and similar spatial environments so that the same UPF can supply them. The PDU session is established between the two UEs and the RAN whereas a radio bearer for each UE is set up. The UPF must be able to handle the necessary QoS flow and needs free resources to establish the session. The source UE then sends the data to the responsible gNB where it gets routed to the UPF. The UPF realizes that it handles both the receiver and transmitter, sending the data packet to the gNB connected to the destination UE. This scenario works for a setup where both UEs are connected to the same gNB as shown in Fig. 2.14a or a setup where two different gNBs like in Fig. 2.14b shown, are in use.

The third scenario, shown in Fig. 2.15 displays the connection between two UEs whereas a different UPF is needed to connect them. Possible reasons for that scenario could be that the first UPF has insufficient resources left to handle another UE, the location between the UEs is more significant than the connection radius of the deployed UPF or the QoS demands of the two UEs are different. The data flow is similar to scenario two except the source UPF routes the data packets to the destination UPF via N9 interfaces as an IP packet.

2.5 Network-slicing and QoS

The specifications assume a broad spectrum of use cases can be implemented with 5G. From handling a massive amount of UEs that send only a few data packets every second up to fast connection with low error rates, the possibilities diverge very strongly. In addition, a 5G network should be able to serve these differ-

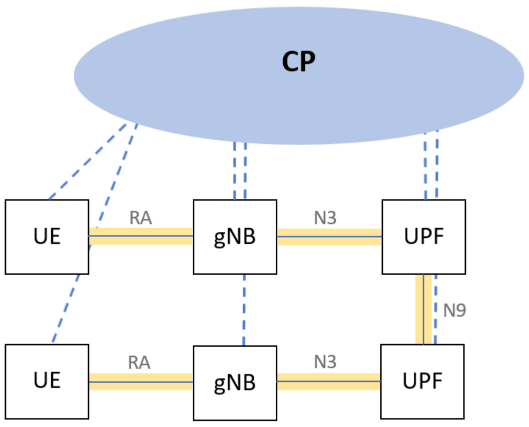


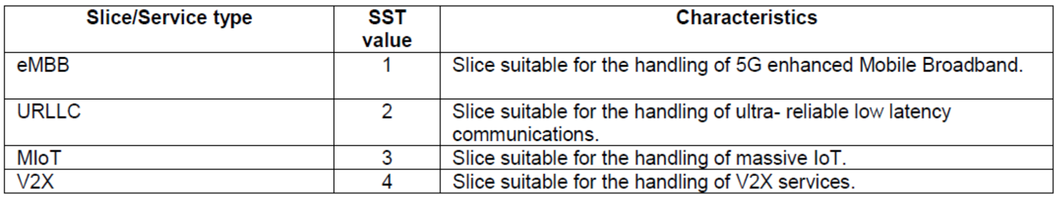
Figure 2.15: Dataflow between two UEs with a different UPF

ent use cases simultaneously without interfering with each other. A key enabler to make this possible is network slicing.

In 5G network slicing creates isolated virtual networks within a PLMN with independent functions, configurations, policies, and security domains to fit the network slices to a given use case. Controlled are the network slices by the NSSF, NEF, and NRF which are also the only NFs aware of slicing in the first place. Every other NF acts according to the given circumstances and based on the available data stored in the UDM. The NEF and NRF are the internal and external access points to specific NFs and must know which slice to match with which entity of the needed NF. The NSSF interacts with the AMF to select a possible network slice instance identified by the single network slice selection assistance information (S-NSSAI) for the pending PDU session during the connection establishment of a UE. Each network slice operates its own SMF, UPF, and PCF and is operated by an AMF where one AMF can manage several slices. That implies several PDU sessions with completely different UPF configurations can coexist in the same network without influencing each other. [7] [1]

To identify the single network slices the S-NSSAI contains a slice/service type (SST) and a slice differentiator. The slice/service type informs about slice performance and behavior whereas the differentiator is a number to identify multiple entities of the same slice type. For common slice types standardized SST values, as shown in Fig. 2.16, are used to guarantee interoperability. [7]

One significant advantage brought by network slicing is that a QoS flow, which specifies the framework in which a PDU session must operate, is not only applied



Slice/Service type	SST value	Characteristics
eMBB	1	Slice suitable for the handling of 5G enhanced Mobile Broadband.
URLLC	2	Slice suitable for the handling of ultra- reliable low latency communications.
MIoT	3	Slice suitable for the handling of massive IoT.
V2X	4	Slice suitable for the handling of V2X services.

Figure 2.16: Standardized SST values for network slicing [7, S. 199]

in the core system but from end to end of a GTP tunnel. Therefore a QoS flow can provide a guaranteed bit rate (GBR) and is defined by a QoS profile that contains a 5G QoS identifier (5QI), a priority level, a guaranteed flow bit rate, and a maximum packet loss rate. For standardization, the QoS flows were associated with different use cases and can be identified with a 5QI. The 5QI contains the following values:

- Resource type:
  It contains information if the QoS flow operates with a guaranteed flow bit rate or not. In addition, a delay critical GBR is available, which further restricts whether a packet is discarded or not.

- Priority level:
  If there are two or more PDU sessions established with different QoS flows, a hierarchy is defined by this priority level. This means that the lower the priority level, the higher the packet's priority from a particular QoS flow is in the core.

- Packet delay budget:
  Defines the maximum time a packet may be delayed between the UE and the N6 interface of the UPF. In the case of delay, critical GBR packets that exceed this value are automatically added to the packet error rate.

- Packet error rate:
  Sets the upper boundary of sent packages that did not arrive or in the case of delay critical GBR did not arrive in time at the receiver.

- Averaging window:
  Time over which the guaranteed flow bit rate is calculated.

- Maximum data burst volume:
  Maximum data must be transported within the packet delay budget.

Some of the standardized identifiers are listed in tab. 2.1. [7]

5QI Value	Resource Type	Default Priority Level	Packet Delay Budget	Packet Error Rate	Default Maximum Data Burst Volume	Default Averaging Window	Example Service
1	GBR	20	100 ms	10^(-2)		2000 ms	Conversational Voice
2	GBR	40	150 ms	10^(-3)		2000 ms	Conversational Video
4	GBR	50	300 ms	10^(-6)		2000 ms	Non-Conversational Video
5	Non-GBR	10	100 ms	10^(-6)		N/A	IMS Signalling Video
6	Non-GBR	60	300 ms	10^(-6)		N/A	TCP-based communication (i.e. www, e-mail, chat)
7	Non-GBR	70	100 ms	10^(-3)		N/A	Voice and Video Live-streaming interactive gaming
8	Non-GBR	80	300 ms	10^(-6)		N/A	Video Buffered-streaming TCP-based communication
79	Non-GBR	65	50 ms	10^(-2)		N/A	V2X messages
80	Non-GBR	68	10 ms	10^(-6)		N/A	Low Latency eMBB applications and Augmented Reality
82	delay-critical GBR	19	10 ms	10^(-4)	255 bytes	2000 ms	Discrete Automation
83	delay-critical GBR	22	10 ms	10^(-4)	1354 bytes	2000 ms	Discrete Automation, V2X messaging
86	delay-critical GBR	18	5 ms	10^(-4)	1354 bytes	2000 ms	V2X messages

Table 2.1: Standardized 5QI Values [7, S. 109-110]

3 Hardware and Network Architecture

In this Chapter, the Hardware used in this thesis is defined and the outlines of the testing environment are set. Therefore, the 5G System and its components are described and how the 5GC and RAN can be configured is explained. The hardware used to implement the use case, which will be explained in Chapter 4, is documented, and the premises in which the tests take place are specified.

3.1 5G Core and RAN Network

The 5GC used in this thesis is provided by the company Athonet [25] and is set up on a Dell virtual edge platform 1485N. This universal customer premises equipment contains all NFs of the core as well as the whole SBA. When this thesis was written, the core had most of the functionalities the 3GPP release 15 provides and will be updated soon to newer releases. Connect to the core via fiber is a Mikrotik CRS305 switch [26]. This switch connects the core with the RAN, which contains two 5G NR gNB femtocells of the type BTI nCELL-F2240 [27]. These gNBs have a maximum transmission power of 250 mW and can switch between 64 and 256 QAM, among other setting options. This setup is shown in Fig. 3.1. Fig. 3.1 also shows that the interfaces N1, N2, and N3 are implemented as one physical interface for the three virtualized N interfaces. This means that all data related to the CP and the UP flows through the switch and to the core via one fiber cable. This port on the switch is mirrored to an empty port to connect a host running a packet sniffer for testing and troubleshooting.

The configuration of the Core is possible via GUI provided by Athonet. This GUI allows Basic management of the system as well as the configuration of network slices and quality of service identifiers. However, this restricts the core configuration to the implemented elements in the user interface. A critical function yet to be implemented is the possibility of applying network slices containing GBR and delay critical GBR 5QI values. In addition, the interface between RAN and 5GC is only conditionally capable of establishing network slicing as a GTP end-to-end tunnel. Therefore, this thesis focuses on the non-GBR 5QI values and evaluates

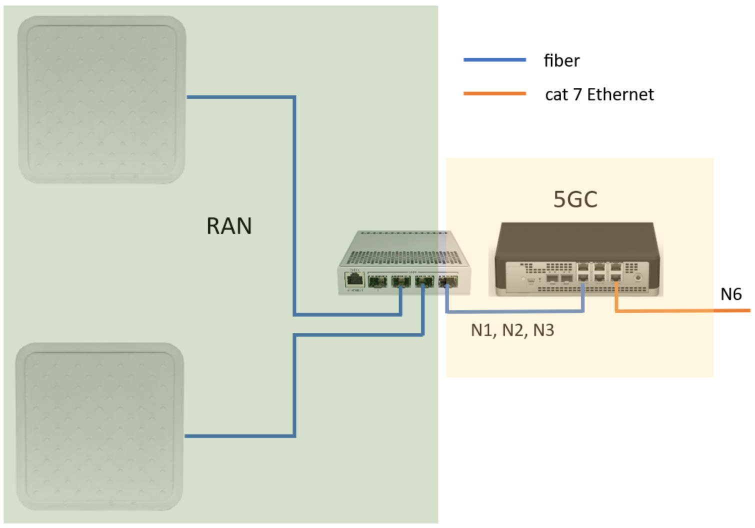


Figure 3.1: Physical setup of the 5G system that is used in this thesis with the connecting interfaces

the possibilities of using those to alter communication channels for industrial purposes.

The RAN can be configured directly at the femtocells, but this is primarily prohibited by the vendor, as the configuration of the base frequency and the permitted bandwidth are restricted due to regulation by the telecommunications authority.

The system is implemented in the infrastructure of the Digital Factory Vorarlberg and the spatial conditions are shown in Fig.3.2. The factory environment which acts as a test set including one of the femtocells in yellow is shown. For the testing only this femtocell was used.

3.2 Additional hardware components

For the use case, two six-axis robot arms of the UR5e type from the manufacturer Universal Robots are used. These robots can be programmed on-site via a touch panel with the integrated software called "Polyscope" or remotely via different client interfaces. For remote access, the control box connected to the robot contains a Gigabit Ethernet interface for IP communication. The two client interfaces used in this thesis are:

- an interface for simple control commands called the "Dashboard Server". The connection is established via the robots IP address on port 29999 and

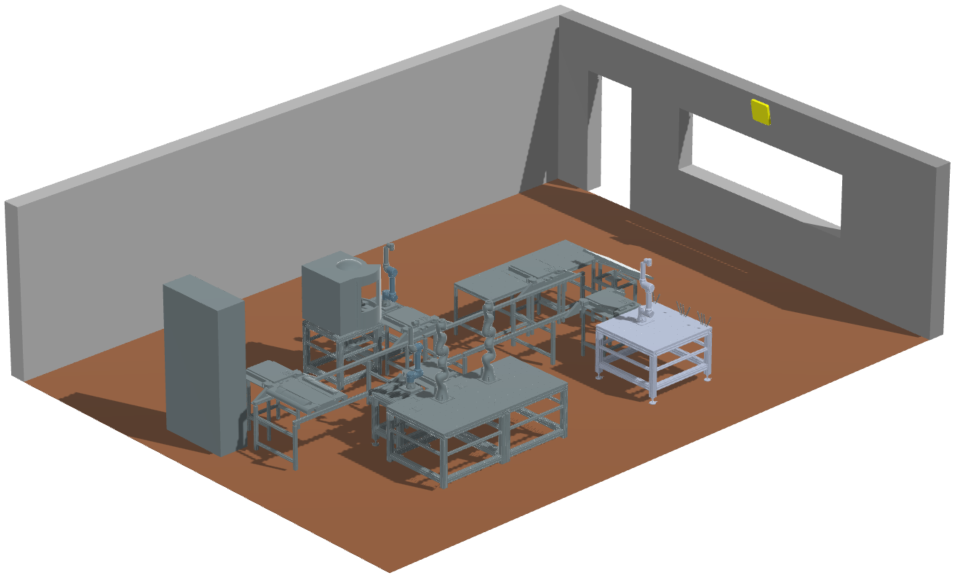


Figure 3.2: Spatial conditions of the test bed in the Digital Factory Vorarlberg

it is possible to power on/off the arm, load and start/stop programs, set the speed value with which the program is executed, and reads the actual robot status. [28]

- the Real-Time Data Exchange (RTDE) interface on port 30004. This interface allows synchronization and communication with the robot without breaking real-time properties. To do this, a setup procedure is first carried out to determine which data is received from the robot and which is sent to the robot. Once done, the data is transmitted in the specified order with a configurable interval. The RTDE interface is generally configured for a frequency of 125 Hz but can be changed up to 200 Hz. Please note that the real-time controller of the control box has a higher priority than the RTDE interface, and it can happen that data packets are not sent if the resources are not available. Therefore, increased frequency can lead to packet loss on the RTDE interface. [29]

One of those two robot arms is mounted on a workbench and is therefore not movable. The other however is mounted on a mobile platform of the type "RB-KAIROS+" from the manufacturer Robotnik [30]. This mobile platform enables the arm to move freely and allows testing to be performed while moving and with different local conditions. The advanced abilities of this mobile platform are not further discussed since they are not used in this thesis.

Industrial 5G modems from the manufacturer Teltonika type RUTX50 [31] with firmware version 07.03.04 are used to connect the robots via Gigabit Ethernet with the 5G network. The modems use 4x4 MIMO antennas to connect with the 5G Network and the maximum ratings for download speed is 2,1 Gbps and for upload speed 900 Mbps. They can also establish a WAN connection via Gigabit Ethernet which is further used for evaluation purposes.

3.3 Test bed

The environment used for this thesis will be a fundamental building block for further research in the areas of 5G, machine-type communication, and industrialization. This makes it inevitable to set up a test bed for upcoming research as configurable as possible to guarantee further use of the testing environment to analyze the communication between different devices and to evaluate the influencing factors of the 5G network.

To achieve this the two robots are connected to the industrial 5G modems mentioned above over Ethernet. The mobile robot has a built-in modem, and the stationary robot is connected to an external modem. These modems allow the connection of up to four devices in a wired LAN and can set up a WAN connection via 5G, Ethernet, or WiFi. Furthermore, on each modem, a laptop is connected for control or measuring purposes. This setup is shown in Fig. 3.3 and is the foundation on which the use-case is built, and the measurements will be executed. Further tests require additional modems to inject traffic into the system. For this

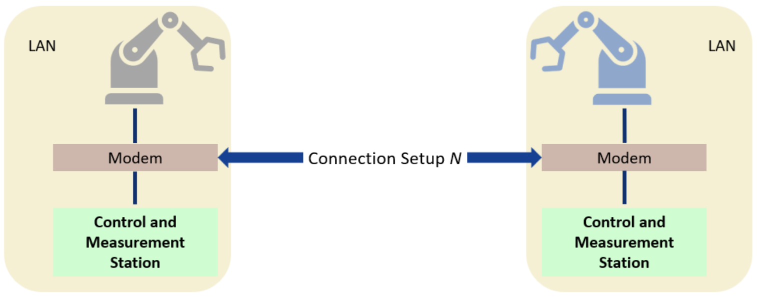
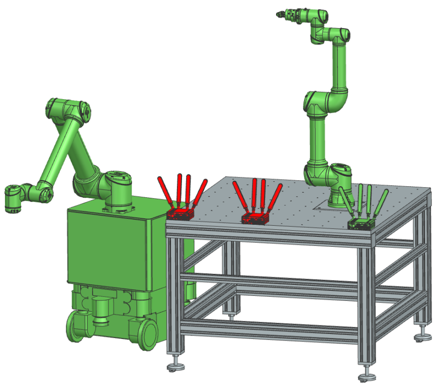

LAN
Modem
Control and Measurement Station
Connection Setup N
LAN
Modem
Control and Measurement Station

Figure 3.3: Testbed shows the connection of the two robots and the control and measurement stations. The connection setups between the robots are considered in more detail in Chapter 5

purpose, two modems of the same type are used and during testing, laptops are connected to these modems for traffic generation. The complete test bed is shown in Fig. 3.4, where the used devices for the connection under test, and the devices used for traffic injection are visible.


(a) CAD model, where the devices for the test link are marked in green and the devices used for traffic injection are marked in red.

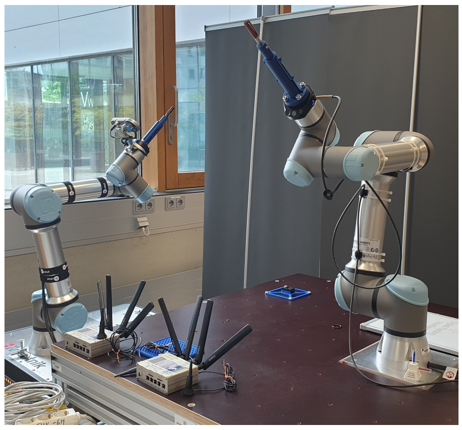

(b) Picture of the real setup

Figure 3.4: Detailed view of the devices used for the testbed

3.4 Management software and configuration interfaces

The 5GC management and configuration can be accomplished through a web interface provided by Athonet. The AMF, SMF, UPF, and unified data repository (UDR) are the key functions configured for further use.

The AMF contains the network name as well as a region ID and an AMF ID. The allowed network slices for the given network must be added to the associated PLMN.

The SMF adds a DNN to the corresponding network slices and assigns an IP address pool for every DNN. In addition, the SMF links every DNN to a UPF.

The UPF must be configured to route data from one virtual interface to the other according to the GTP tunnel established. Therefore, a mapping between network instances, DNNs, and virtual routing and forwarding links must be done. This means that a dedicated virtual interface must be associated with a DNN, and the core must capture this connection into its routing table.

Most configurations on the core are done in the UDR, where all the subscription and policy data is stored. For every type of UE, a provisioned data profile must be added, which contains access and session management data. The access management data contains the maximum Mbps for up- and down-link and the allowed network slice. The session management data contains most of the configurations used for further testing in this thesis and the following configurations can be done:

- add allowed network slices for this UE,
- link UE via DNN to a certain network slice,
- maximum up- and downlink bandwidth,
- enter 5G QoS identifier,
- configure preemption capability and vulnerability. This allows a UE to preempt a preemptable UE if the resources are needed,
- overwrite the default QoS priority level,
- configure a static IP and allow only certain IP types,
- select allowed SCC modes.

In addition to the provisioned data profile, a policy data profile must be added, which links the data policy of a UE to a DNN and a network slice. When the profiles are configured, they must be assigned to a SUPI which relates directly to the IMSI and therefore to a SIM card. If this SIM card is inserted into a device, it can communicate via the 5G connection according to the configured parameters.

The 5G modems were also configured via a web interface. The configuration of the mobile network and packet routing needed to be done and additional configurations regarding traffic logging and packet size where necessary.

3.5 Network configurations

An overview of the network architecture helps to understand, how the connection is established and where the packets are routed eventually. The 5G system creates a network with the ip 10.10.0.0/24 and the modems connect to that network. The LAN connection on each modem operates in a 192.168.X.0/24 network where X represents the network number for better comprehensibility. To communicate with a host in the individual LANs, port forwarding was used i.e., to connect to the RTDE server of a robot, the modem must be configured to route incoming traffic from the 5G side on port 30004 to the IP address of the robot in the LAN. This allows only restricted access on the LANs and enables a simple implementation to re-route traffic from the 5G network to single hosts in the LANs. The network is shown in Fig. 3.5, where the single sub-nets are highlighted as well as the used hardware components.

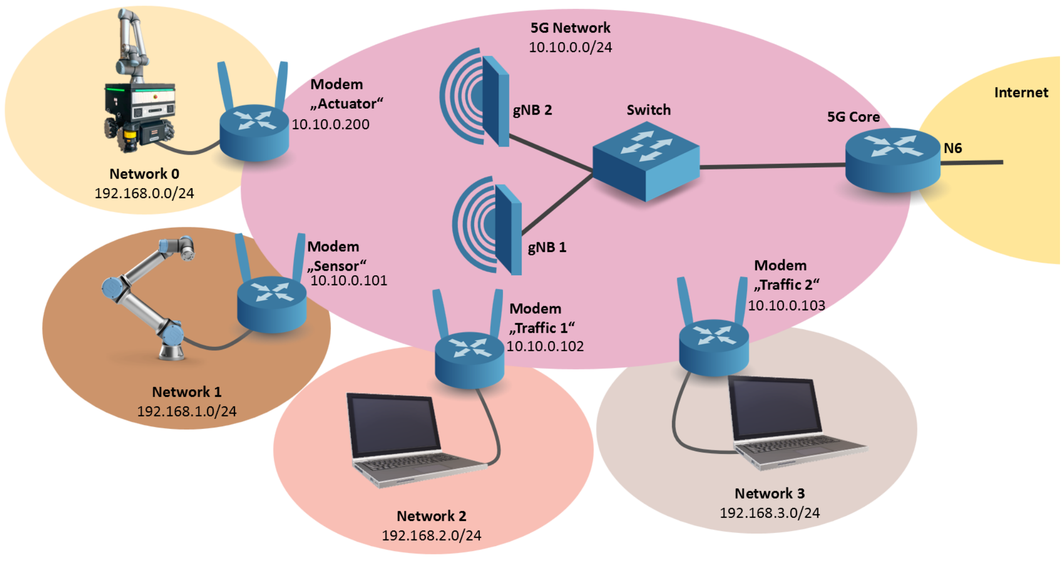


Figure 3.5: Network overview containing the 5G network, the single LANs created by the modems, and the used hardware.

4 Use-Case

This Chapter explains the use case, its implementation, and which network metrics were identified as most important.

In today's industry, modern communication technology has become indispensable. Regardless of whether it is human-to-human, human-to-machine, or machine-to-machine, a connection appropriate to the application must be established. To make the tests and insights of this thesis as close as possible to reality, a use case was sought which applies to a wide range of situations. For this purpose, the existing infrastructure was used as an implementation framework and the system's possibilities with its current state were considered. The focus was placed on testing non-GBR 5QIs for two reasons. The first was to prevent packets that exceeded the appropriate timeout from being directly discarded by the 5G system, and the second was that the interface between the 5G core and the RAN was found to be buggy concerning GBR 5QI settings.

The use case was implemented at the model factory of the Digital Factory Vorarlberg GmbH. In this factory, the articulated arm robots described in 3 perform a task and exchange data with each other during the execution. In addition, sensors for data acquisition send their information to a central location and the employees in the factory can communicate with each other via digital communication devices. The entire communication is carried out via a private 5G SA network.

The robots communicate with each other via a TCP connection and the other devices generate UDP traffic. The robot communication was set up in reality and will be explained further in this chapter. The generated traffic from the sensors and communication devices was simulated using generated UDP traffic to simplify the setup. For traffic generation, two additional laptops were connected to the network via 5G modems. On those laptops parallel video streams enabled a constant UDP stream with a configurable bandwidth.

The connection under test, further called test link, is the communication between the robots. Therefore, the two articulated robots, one stationary and one mounted on a mobile platform, are connected over the 5G network. This allows the control and monitoring of the robots via a remote connection. Moreover, the mobile robot can move freely in the area with network coverage as the only limitation. Two possible applications for this would be coordinating synchronized movements to achieve a joint task or the remote control of a robot in real-time. Due to the limited time frame of this study, the chosen implementation combines both applications in one use case. It allows to evaluate the influence of different communication setups on the machine-type communication between the robots.

In the implemented use case the stationary robot, further called the "sensor", moves along a predetermined path and transmits information regarding its movement to the mobile robot, further called the "actuator". The actuator robot receives this information and strives to trace the trajectory of the sensor. The two robots are connected to a 5G Modem, explained in Chapter 3. In the beginning, the modems are connected via Ethernet to acquire data that can further be used as a baseline. Afterward, the cable between the modems is removed and they are connected via 5G with different setups. On the sensor and the actuator modem, a laptop is connected to the LAN via Ethernet to control the data sent and received from the robots and implement further tests.

This use case was selected to investigate the framework conditions for automation through 5G technology and determine the extent to which these conditions can be met by the existing system. Furthermore, the gap between network measurements and robot movements will be closed to facilitate the interpretation of correlations for future implementations.

In the following sections, it is described how the use cases were implemented, the tracing of a trajectory is done, and the code structure is explained. Furthermore, performance metrics were identified to describe the quality of the connection and determine the influence of external factors.

4.1 Trajectory tracing with real time data exchange

The first step was to specify the way to record the trajectory of a UR5e. Based on the vendor information a suitable solution was to use the RTDE interface of the Universal Robots, since this interface allows to read and write specific data from and to the robots in a timed interval. The RTDE server runs on the robots by default and can be accessed with an RTDE client running on an external computer.

To implement this, a laptop running Ubuntu 22.04 is configured as a control station with which data is read from the sensor and movement information is sent to the actuator. To meet the low latency requirements the real-time kernel implemented by default in the mentioned OS was activated to be able to intervene in case of problems, but no explicit configuration was done. Furthermore, as described in [32], the programming environment for addressing the RTDE interface was set up and a program based on the examples, given by the vendor [33] as starting point, was written. Here the decision was made to write the code with Python because the implementation was easier to handle, and the C++ implementation only supports Linux systems. This would have been feasible with this setup, but it cannot be guaranteed that a Linux operating system will be used for this application in the future.

The written code contains the following:

- Initial configuration to specify which data is recorded from the sensor and the actuator,
- Communication information for sensor and actuator. Here, not only the RTDE interface but also the dashboard interface was used. With the RTDE interface data regarding the trajectory is received and transmitted whereas control signals are sent to the dashboard server,
- Two threads for handling the reception and transmission of the data,
- Data processing of both trajectories and writing of the data to a .xlsx file.

This setup is shown in Fig. 4.1

In the beginning, the IP addresses and the used ports are set. To ensure the program is not interrupted by minor processes running in the background the nice value is set to -19. A global variable containing trajectory movements is initialized and the staring time is recorded. A connection to the dashboard server of the sensor and the actuator is established and the used programs on the robots are started. The sensor thread sets up the RTDE client, connects to the sensor robot via RTDE, and then receives the information specified in the setup procedure in a loop with a frequency of 125 Hz. The recorded data and the global time at the reception are written in the global variable in each loop. The actuator thread establishes a connection with the actuator robot in the same way and receives the actuator data. In the same loop, the actuator thread sends the sensor data stored in the global variable to the actuator robot which then moves according to the received information. After the movement of the sensor and actuator, the robots are

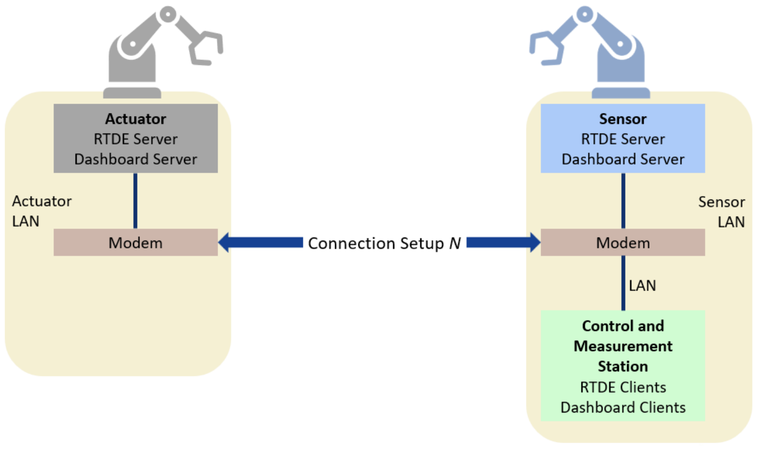


Figure 4.1: Detailed setup of the use case with the connection of the two robots and the control station. The additional connection setups between the robots are considered in more detail in Chapter 5

stopped, and the recorded data is processed and written to a file.

This implementation allows to log the data from sensor and actuator in one program. It provides a timestamp from the same timing domain to avoid the need of synchronizing the two robots over the network using SNTP or PTP. Furthermore, it allows the representation of different network behaviors by comparing the data from the sensor and actuator. The critical part of this implementation is the reading of the global variable while the sensor writes the new value to it. To prevent the actuator to read the variable in an undefined state, the variable is protected by a mutex.

In the first implementation, the trajectory tracing was done by sending the axis angles of the sensor to the actuator while the robots were connected via Ethernet with each other. These measurements showed a delay of over 100 ms and the measured delay fluctuated heavily. To improve this behavior, a change from sending axis angles to axis speed was made. On the actuator robot a change from the servoJ command, which accepts axis angles, to the speedJ command was implemented. This improved the delay time between the robots to approximately 20 ms with very little fluctuation. [29]

4.2 Code-structure

The program flow is shown in Fig. 4.2 for further understanding. On the left side you can see the main program, which establishes the connection to the robots via dashboard and then calls the programs on the robot. Then the threads for the RTDE communication with the sensor and the actuator are created and the programs on the robots are started. Meanwhile, the two threads start connecting to the RTDE servers on both robots and start exchanging data. If the program is running, data is received from the sensor and sent to the actuator. When the program is finished, the connection is terminated, and the threads are closed.

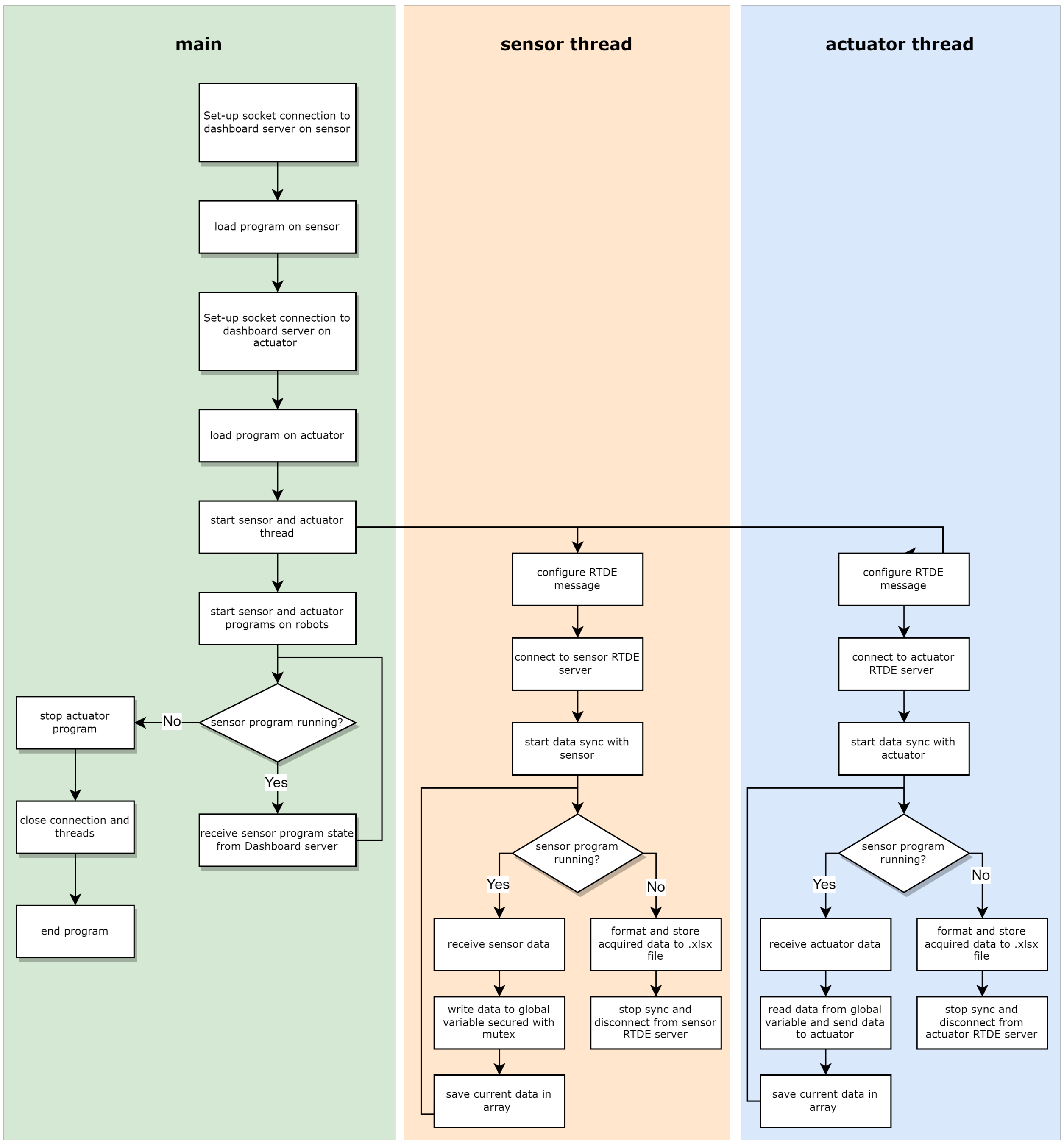

main
sensor thread
actuator thread
Set-up socket connection to dashboard server on sensor
load program on sensor
Set-up socket connection to dashboard server on actuator
load program on actuator
start sensor and actuator thread
start sensor and actuator programs on robots
sensor program running?
No
stop actuator program
close connection and threads
end program
Yes
receive sensor program state from Dashboard server
configure RTDE message
connect to sensor RTDE server
start data sync with sensor
sensor program running?
Yes
No
receive sensor data
write data to global variable secured with mutex
save current data in array
format and store acquired data to .xlsx file
stop sync and disconnect from sensor RTDE server
configure RTDE message
connect to actuator RTDE server
start data sync with actuator
sensor program running?
Yes
No
receive actuator data
read data from global variable and send data to actuator
save current data in array
format and store acquired data to .xlsx file
stop sync and disconnect from actuator RTDE server

Figure 4.2: Program flow of the trajectory tracing including the threads receiving and sending data as well as the data acquisition for further analysis.

4.3 Performance metrics

The goal is, to provide an as exact as possible real-time copy of the sensor trajectory on the actuator. The mechanical delay can only be influenced to a certain degree and therefore, the performance of the network is essential to achieve the best possible result. Since the performance of a Network can be described with different measurands the testing specified on three of them:

- Round-trip time (RTT) described the time a packet needs from sender to receiver and back,
- Jitter describes the change in RTT during measurements and can be expressed as packet delay variation (PDV) or inter-packet delay variation (IPDV), [34]
- Packet error rate (PER) describes the number of packets lost between the sender and receiver.

An additional measurement directly related to the functionality on the implemented use case is done. Here the trajectories of the sensor and the actuator robot are compared to each other and the offset in the time domain is measured. The correlation of the two trajectories provides additional insight into the jitter of the network and the necessary re-transmissions needed. The code of the programs used in this thesis is in the appendix. [35]

4.3.1 Round-trip time

The RTT gives insight into how long Data packets travel from one point of the Network to another and back. When the packets travel the same way in the network, and the conditions for upload and download are the same, then the RTT is two times the end-to-end latency or one-way packet delay. In networks where the traveled path can not be determined, one way can take longer than the other and therefore, both ways should be measured separately. Since the network architecture used in this thesis is known and the path of the packets is the same, it could be assumed that the RTT can be referred to as twice the packet delay. In [3], it is shown that this is not necessarily the case for 5G SA networks. For the presented use case, however, a split of the RTT in upload and download is not needed since the packets must always be uploaded and downloaded. Furthermore, an implementation to test up and download separately would have exceeded the time frame of this thesis and will be done in future implementations.

There are two common ways to measure RTT. First is the ICMP Echo used via, i.e., a console application with the command "ping". Therefore, an ICMP request is sent to a receiver which then sends back a reply. The time difference between sending and receiving the packets relates to the RTT. The advantages are that only one device is necessary since nearly every network device can reply to an ICMP request. A negative aspect would be that ICMP works on the network layer of the OSI model, and therefore the time for computing the upper layers is not considered. The second way would be sending TCP packets to a server which routes them directly back to the transmitter. Here it is necessary to implement a client and a server, and access to both sides of the connection under test must be given. The principle of testing is the same for both ways, but the handling of the different packet types can differ in certain ways according to [36]. To avoid this, both types of measurements have been implemented and can be compared with each other. This also increases the number of measurements and thus the relevance of the samples. [35]

TCP is connection-oriented and contains the re-transmission of packets that are not sent correctly. In [37], it is described how this re-transmission of TCP packets influences the measurement of RTT and how Karn's Algorithm can be used to improve measurement methods to exclude the increasing time measured when a packet must be resent. However, since the described use case involves the re-transmission of packets, an implementation that excludes the re-transmission time was not used, which should result in a higher RTT time for TCP packets compared to ICMP packets, but in the end, no packets are lost during the TCP communication.

It is important to notice that the RTT depends strongly on the free network capacity. The more a network is loaded with traffic, the higher the RTT times can become. Of course, this also depends on the priority of the various connections and can therefore be influenced from the outside. In this case, especially the influence of different 5QI, according to 2.1, is investigated.

4.3.2 Jitter

When the time a packet travels from one point in the network to another is always the same there is no network jitter. Since a lot of variables influence the delay of packets, jitter is prominent in every network. When the jitter exceeds a certain amount of time, packets may arrive in the wrong order, which consumes additional time to resort them, and data that arrived in time must wait for further delayed packets. This can cause problems in deterministic systems when deadlines are not met.

Especially in wireless communication, jitter is inevitable. Since data travels through the air, small changes in distance, reflections, movement, or obstacles all cause a variance in transmission time, and therefore it is an important measurement.

To further specify what is described by network jitter, the formulations according to [34] are used. There the delay variation of one-way delays is specified, and this specification is used in this thesis to differ the delay variation of the measured RTT in:

- Inter-packet delay variation, which describes the change in packet delay from one packet to the next. This metric shows how deterministic a system is and how stable the packet delay between packets is. The IPDV can be calculated as

  $$IPDV = RTT_i - RTT_{i-1} \tag{4.1}$$

  where $RTT_i$and RTT_{i-1} are the measured round-trip times of consecutive packets. In this equation, the IPDV can also have negative values. Since only the variation of the packet delay is under observation, the absolute value of 4.1 is used.

- Packet delay variation, which describes the change of packet delay during a series of measurements. If the RTT is constantly increasing over a series of measurements, this trend cannot be detected by the IPDV. The PDV, however, always relates the difference in RTT to the lowest RTT measured in a series of measurements, and thus it can be determined whether this drifts away over the time of the measurement. The PDV can be calculated as

  $$PDV = RTT_i - RTT_{min} \tag{4.2}$$

The measurement of IPDV is performed in the same process as the RTT measurement, and the PDV is calculated afterward on the basis of the gathered data. [34] [35]

4.3.3 Packet error rate

The PER relates to the number of packets lost during transmission or received after the configured timeout and is normally described in percentage. It is calculated by summing up the number of not or late received packets and dividing this number by the total number of packages transmitted. One way to measure PER is to send packets with a known value from a host to a network device, which routes the packets back to the sender. The sender checks if the packets arrive correctly and on time. This can be done via ICMP echos since there is no re-transmission that prevents packet loss. [1]

4.3.4 Trajectory comparison

The trajectory comparison is not a typical network measurement since it relates to the two measured trajectories of the robots. This is the measurement metric that allows to link the network performance to the use case implementation feasibility or requirements. Therefore, the trajectories, represented by the axis angles of the sensor and actuator robot, are compared to each other. This comparison is made based on the correlation between the two trajectories to observe the delay and jitter that is caused by the network. When comparing two data sets, the correlation coefficient R is a 2-by-2 matrix and describes the linear association of these datasets around a line. The diagonal entries describe the correlation of the data sets to themselves and are always one. The other entries describe the correlation to each other and range from minus one to one, where minus one means the data correlates around a straight line with a negative gradient, and one means the data correlates around a straight line with a positive gradient. Zero correlation means the data does not correlate at all.[38]

When using two data sets, X and Y, the calculation of R can be performed as follows

$$R = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i-\mu_X}{\sigma_X}\right)\left(\frac{Y_i-\mu_Y}{\sigma_Y}\right) \tag{4.3}$$

The data is first normalized, and the sum of the product is calculated and divided by the number of data points per data set. This calculation can now be used to find the difference between two functions on the x-axis. In Fig. 4.3, the first picture shows two sinusoidal functions that are phase-shifted by half a period to each other. This leads to an R-value of minus one. The red signal is now circularly shifted to the right in $\frac{\pi}{3}$ steps, and the R-value increases from left to right until it reaches one where the two signals overlap.

To find the offset on the x-axis, the two signals are shifted over one period, and the R-value is calculated for multiple steps during this shift. The maximum R-value then relates to the best-fitting offset in the x-axis which is shown in Fig. 4.4, where the R-value is plotted while circular shifting one signal. It is visible that the maximum R-value is at half the period of the two signals from Fig. 4.3.

This method is used to find the delay between the sensor and the actuator trajec-

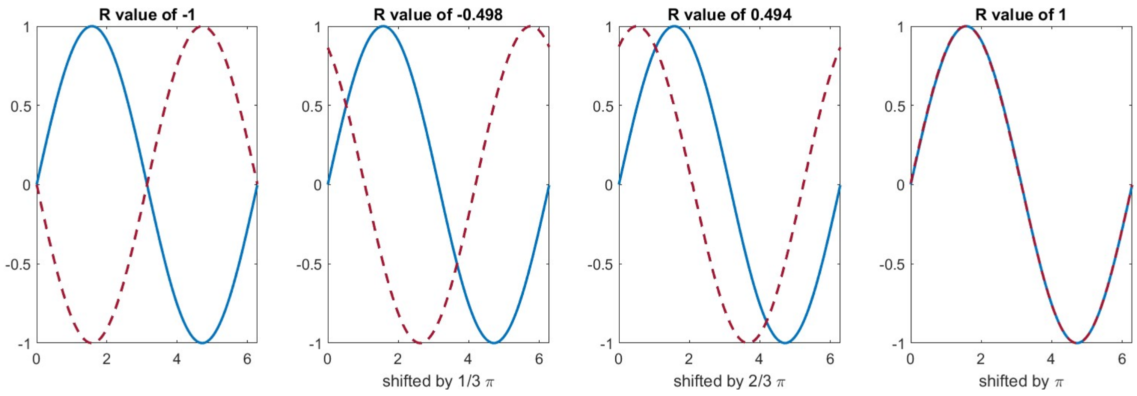


Figure 4.3: Change of correlation between two signals during circular shifting from left to right

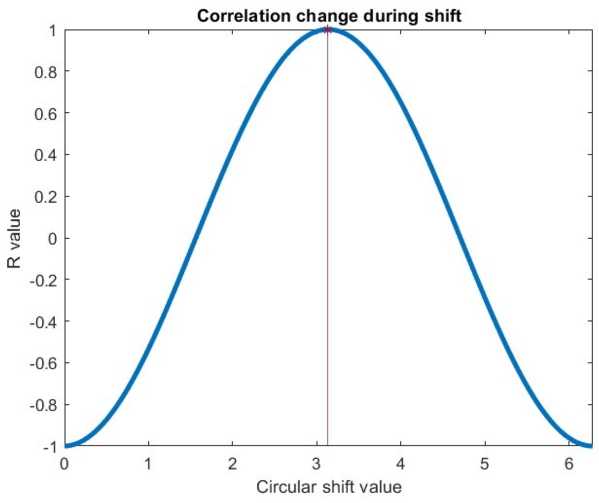


Figure 4.4: Change of the R-value while shifting over a period with marked maximum

tory. To determine the influence of jitter on the signal, the R-value provides information about the equality of the trajectories. Fig.4.5 shows how the distortion of an original signal influences the R-value. Starting on the left with two identical signals, one gets distorted more and more, and the maximum R-value gets smaller. In addition, this jitter leads to a change in the x-axis due to the heavy distortion of the signal.

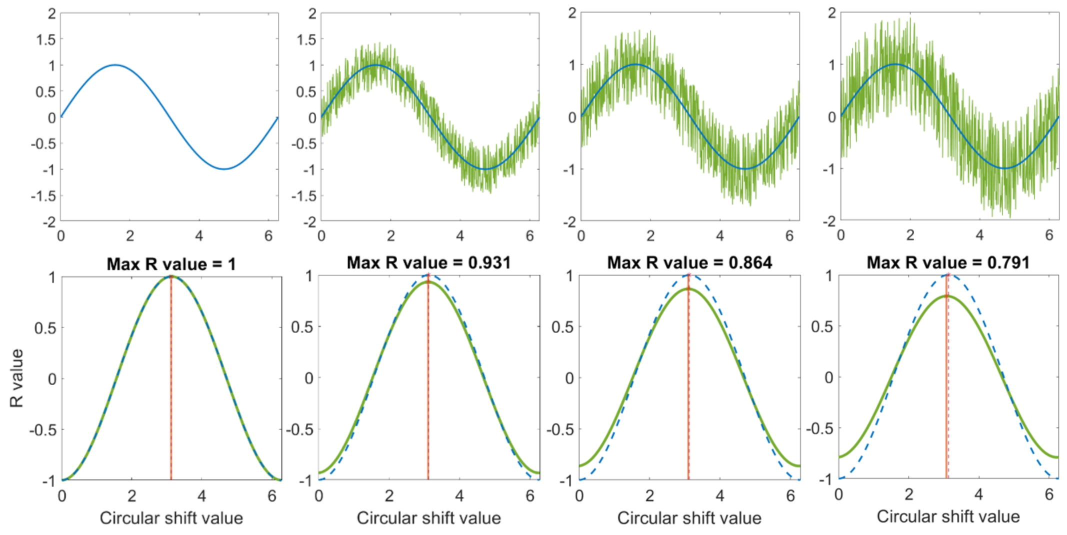


Figure 4.5: Change of the maximum R-value due to distortion of the signal with increasing intensity from left to right

5 Measurement setups and configuration

This chapter describes the implementation and visualization of the individual measurements, the focus of the different setups, and explains why they were chosen. The configuration for the measurements, starting with a wired setup and continuing with different 5G settings, is presented and first results are used as a reference. The results of the measurements regarding these setups are then presented in chapter 6.

5.1 Implementation and visualization of the measurements

First, the measured link for RTT had to be defined. Therefore the Fig. 3.5 was adapted and the communication path is shown in Fig. 5.1. A Laptop connected to the sensor modem sends the measurement packets to the 5GC where it gets routed to the actuator. The actuator then sends these packets back to their destination via the 5GC. This means that the measured RTT corresponds to two times upload and two times download. Assuming that the send time is the same for upload and download, the RTT corresponds to a fourfold one-way delay.

To measure this, a program was written that opens a console via Python code on the measurement PC connected to the sensor modem and sends ICMP requests to a Network device via its IP address. The number of packets, the packet size, and the sending interval can be configured, and the textual reply which contains information about the RTT, and the packet loss is then saved and the valuable information is extracted. One disadvantage of this implementation is that the next ICMP packet is only sent when a reply has been received from the previous ICMP request or the timeout is exceeded. This means that only one packet at a time is sent from the sender to the receiver, which can greatly increase the measurement time. For the measurements of the ICMP packets a timeout of 10 seconds was used. All packets that take longer than 10 seconds to get back to the sender are considered lost. To reduce the produced data and decrease extreme values, ten

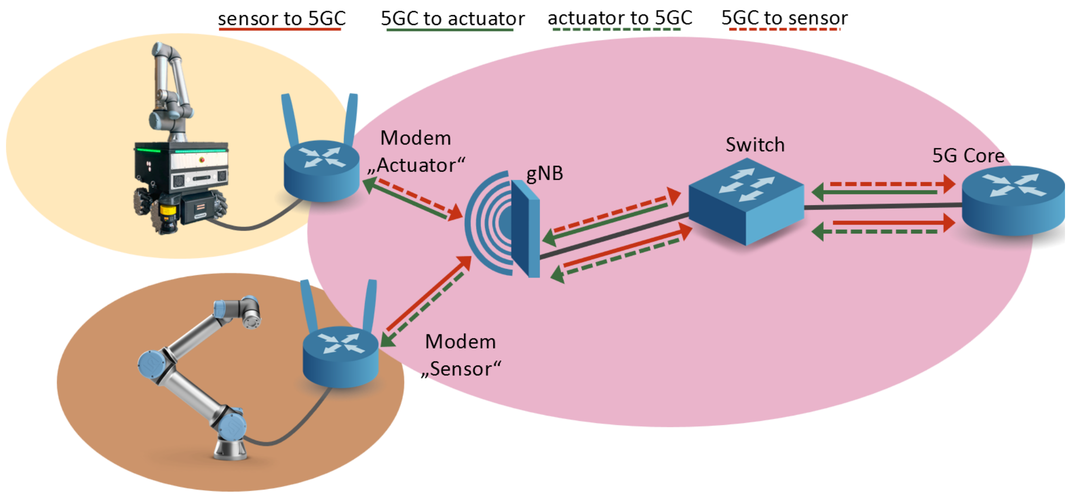


Figure 5.1: Measured link for RTT with the individual sub routes

measurement values are combined into one burst. The mean values of RTT, IPDV, and PDV as well as the summed-up packet errors represent this burst as one measurement for further analysis.

In the later stages of the thesis, it was noticed, that for further examination of the RTT, tests with a fixed transmission interval were needed. This was implemented with the hping3 tool. hping3 is a packet generator and analyzer which works similarly to the known ping but with advanced configuration possibilities, i.e., the sending of UDP and TCP packets or the changing of the interval between every sent packet down to the microseconds range. This tool was chosen mainly because it is possible to configure short and fixed interval times.

For the TCP measurement, a TCP server and client were implemented in Python. The server is connected to one side of the connection under test and opens a TCP server on a specific port. The sole purpose of the server is to receive packets and send them back to the sender. The TCP client is configurable in terms of packet size, number of packets, and waiting time between the sent packets.

It is important to visualize the measurement results as simply as possible and still get all the meaningful information from them. To achieve this for the RTT, box plots and plots of the empirical cumulative distribution function (ECDF) of the data were used.

Box plots use the Median and the quartiles of a data set to give information on the location, distribution, and dispersion. In the middle of the plot is the median and around the median is a box that includes 50% of the data and ranges from the median of the upper data half to the median of the lower data half. This range is also called the interquartile range (IQR). The lowest and highest quartiles are displayed with so-called whiskers. These whiskers usually have a maximum length of 1.5 times the IQR which leads to an upper boundary of the third quartile + 1.5 times the IQR and a lower boundary of the first quartile - 1.5 times the IQR. Everything outside these boundaries is called an outlier. [39]

For the usage of an ECDF, the cumulative distribution function (CDF) should be explained first. A distribution function shows the probability of a value related to a data set where the y-axis represents the probability density and the area under the function represents the probability percentage. When a CDF is used the probability density values are added up with the value before and the graph shows a constantly rising function. Here the values on the y-axis represent the percentage that a value is smaller than this variable. In Fig. 5.2 is a normal distribution function and a CDF shown, where the mean is zero and the standard deviation is one. The ECDF is used instead of the CDF when the data contains empirical samples.

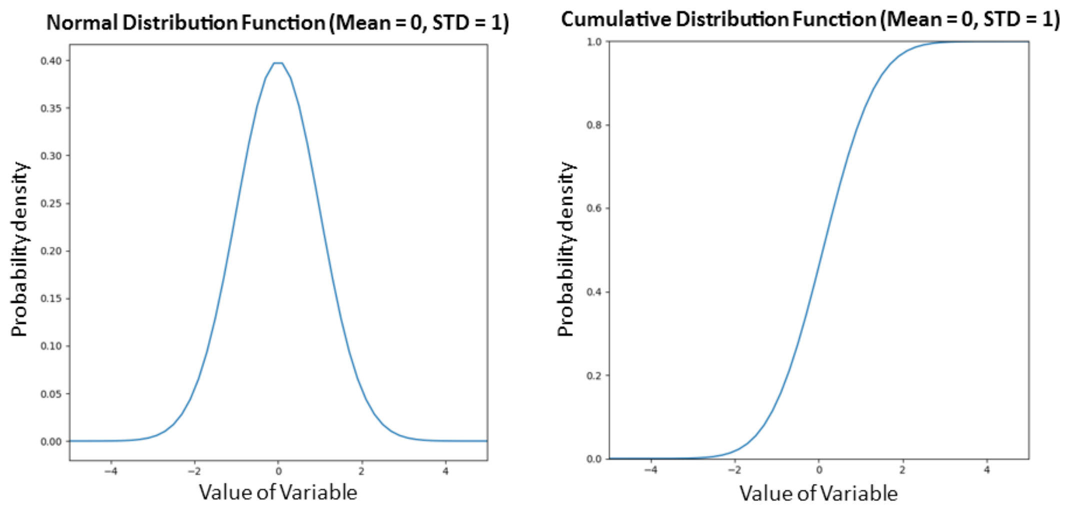


Figure 5.2: Normal distribution function compared to the CDF

The ECDF is an estimation of the CDF, where $\hat{x}$ is the data point under observation, p is the number of data points less than or equal to $\hat{x}$ and n is the number of data points in the sample. The ECDF can then be calculated as

$$\text{ECDF}(\hat{x}) = \frac{p(\hat{x})}{n} \tag{5.1}$$

This is shown in Fig. 5.3 and the function is similar to that in Fig. 5.2, but each measurement is represented as a step rather than a continuous function.

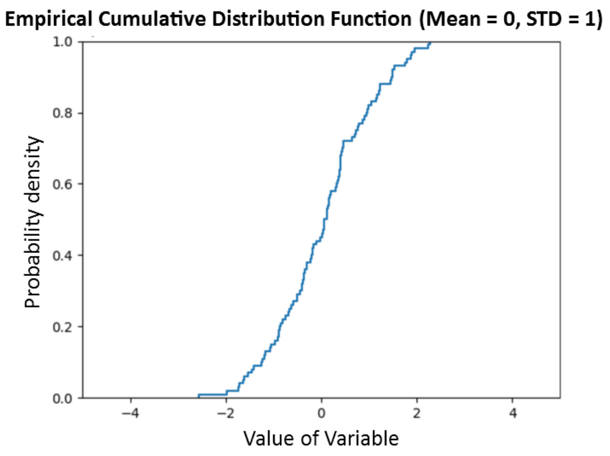


Figure 5.3: Empirical cumulative distribution function of Fig. 5.2

The measurements of the IPDV can be derived from the RTT measurements already acquired. With equations 4.1 and 4.2 the data for the packet delay variation can be calculated and prepared for further investigations.

The IPDV and PDV are plotted separately with boxplots to show the median, range, and outliers. In addition to the Boxplots, the IPDV and PDV are sorted and then plotted over the measurement index to visualize the difference and behavior in certain setups. In Fig. 5.4 is shown how different distributions have different behavior when they are sorted and plotted over the index. These plots allow to assess the dispersion and the mean as well as the distribution of the data. Fig. 5.4 shows that i.e., the bimodal distribution behaves like two normally distributed signals with a slightly shifted mean. This can provide more insights into the processing of the data over the network.

The measurement of the PER is implemented in the RTT measurement. Every packet that arrives late or not at all counts as lost and is summed up over a measurement. This sum is divided by the amount of sent packets and represents the PER in percentage. Since the PER is only a percentage value over one measurement, bar charts represent and compare the single measurements.

For the trajectory comparison, it was important to understand how delay, jitter, and packet loss can interfere with the trajectory of the actuator robot to be able to estimate for which setups a measurement of the trajectory makes sense. A script for Matlab version 2023a was written, to simulate these influences before implementing the measurements for the trajectories. This simulation is only a rough estimation of the errors that occur during the transmission and is used mainly

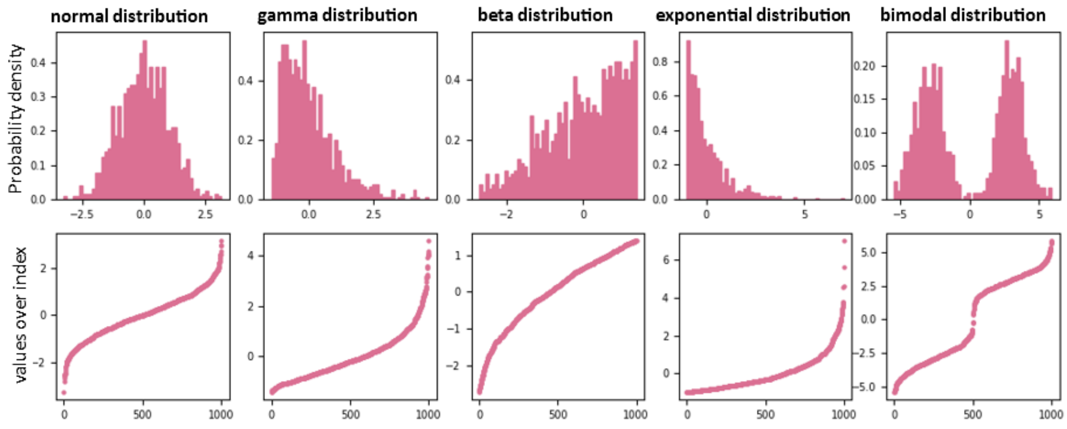


Figure 5.4: Behavior of different distributions when sorted and plotted over the index to visualize network behavior

for visualization, better understanding and to bridge the gap between the implemented measurements and the selected use case. For simplification, a sinusoidal signal with a frequency of 25mHz was used as the base function that simulates the transmitted angular velocity ω_s received from the sensor robot. This signal is sampled with 125kHz to simulate the RTDE interface of the used robots. Tests on the RTT were carried out in advance to estimate the interval with which the trajectory information arrives at the actuator. This was necessary since not every packet received will be transmitted to the actuator due to processing and transmission delays. When receiving data from the sensor, the cycle time of 8 ms can be maintained. However, when sending data over a wireless communication that takes longer than 8 ms to transmit, this data and therefore the receiving interval increases. First measures showed that factor four could be used as an approximation which led to a sample time on the actuator side of 32 ms. This base signal ω_s is sampled with 125 kHz over 40 seconds and the undisturbed signal at the actuator ω_a which contains only every fourth measurement overlaps the sensor signal completely.

Adding a delay to the signal simulates a RTT higher than zero and can be done by increasing the time vector of the actuator signal by a certain value. In addition to the added delay, a delay variation should be added. This variation can occur for two reasons:

- When a packet must be resent because of a packet error or exceeding the maximum transmission timeout. This leads to a change of ω_s at the actuator side and is visible as a steady value over at least one sample time at the actuator. This makes it possible to show a deterioration of the actuator ω_a

compared to the sensor ω_s according to a specific error percentage.

- IPDV caused by the wireless transmission. This leads to a change in RTT according to the intensity of the IPDV. In addition, a high IPDV can also cause an actuator cycle to not receive a packet which results in the first error explained. Therefore, it was necessary to implement the packet error first and then add a random value that is not steady but increases with higher packet error rates. Here a linear function according to measured data was used to simulate the relation between increasing packet loss and IPDV.

These errors are shown in Fig. 5.5, where a delay of two seconds and an error rate of 10% was added. Since the TCP position is the variable of interest, it is necessary

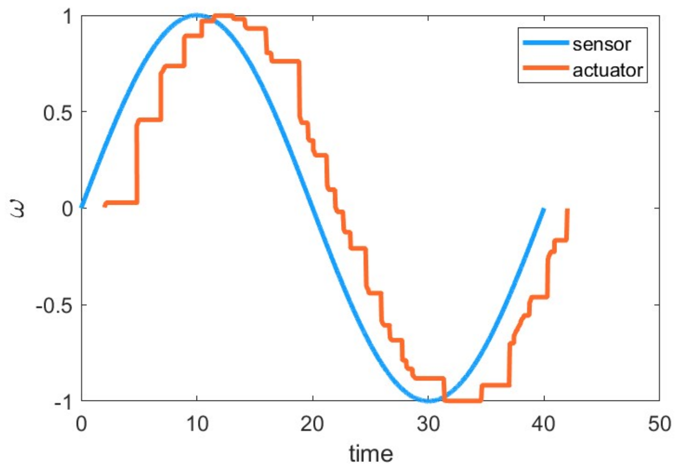


Figure 5.5: Sinusoidal angular velocity for the sensor and a simulated distorted transmission to the actuator

to integrate the angular velocity. This was done with the trapezoid rule

$$\int_i^j f(x)\,dx \approx \frac{1}{2}\sum_{n=1}^{m}(x_{n+1}-x_n)(f(x_n)+f(x_{n+1})) \tag{5.2}$$

where the function to be integrated is the sinus and the variable x is the time vector both after the changes according to the errors applied. This integration is already predefined by Matlab as the cumtrapz() function. [40].

After the integration, the axis angle of the sensor and actuator can be compared which is shown in Fig. 5.6. It is visible that the trajectory is deformed because of the sudden changes of ω_a which lead to an unsteady change of the axis angle. Another insight gained from this simulation is, that a distortion of the angular

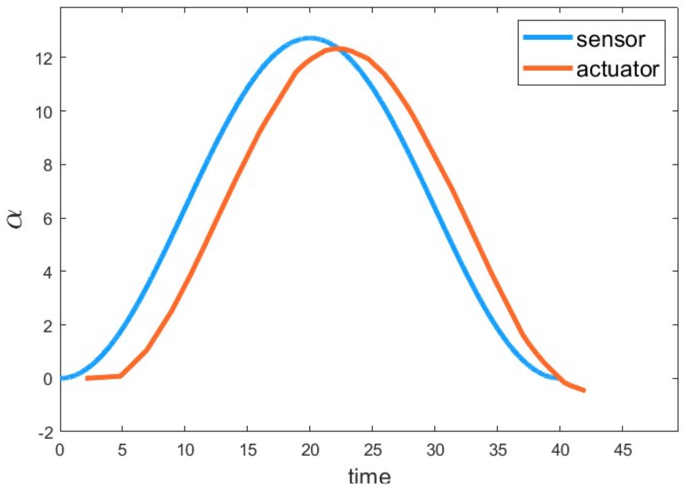


Figure 5.6: Axis angle for the sensor and a simulated distorted transmission to the actuator

velocity leads to a delayed axis angle at the actuator. This is shown in Fig. 5.7 where the transmission was simulated without any delay, but with a high error percentage of 15%. It is visible that in Fig. 5.7 the angular velocity on the left is not delayed but has a high jitter. This results in a delayed axis angle at the actuator, which comes from the partly slower change of the axis speed due to the missing packages resulting in constant speed over time. To find out at what percentage a

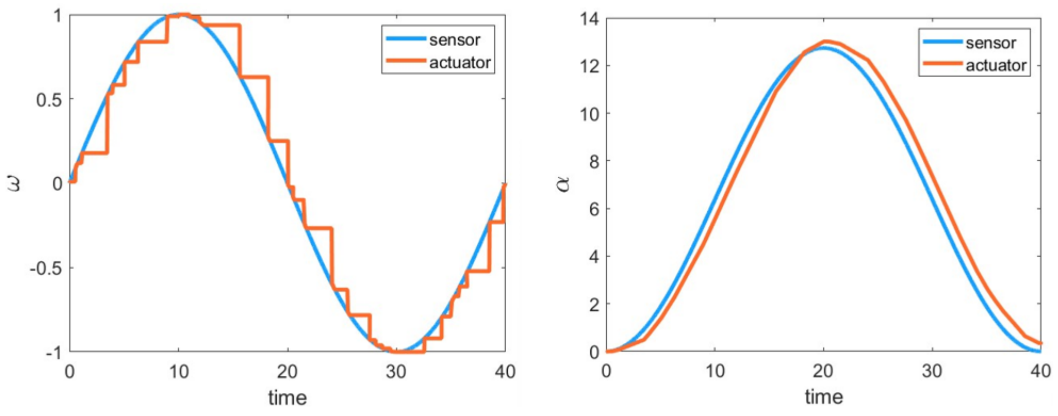


Figure 5.7: Delayed axis angle resulting from the jittered transmission of angular velocity

trajectory measurement stops making sense, the mean squared error (MSE) between the sensor and actuator trajectory for different PERs was computed. At a PER between 10% to 25% a steep ascent of the MSE is recorded which leads to the assumption, that at a PER of 10% the accuracy of the actuator trajectory reaches a critical range where the uncertainty is not sustainable. The normalized MSE is shown in Fig. 5.8 where 20 simulations for each packet loss between 0 to 25 were computed. As a first assessment, it can be said that a transmission resulting in an error of more than 10% at the actuator, which corresponds to a PER of about 22%, is unusable.

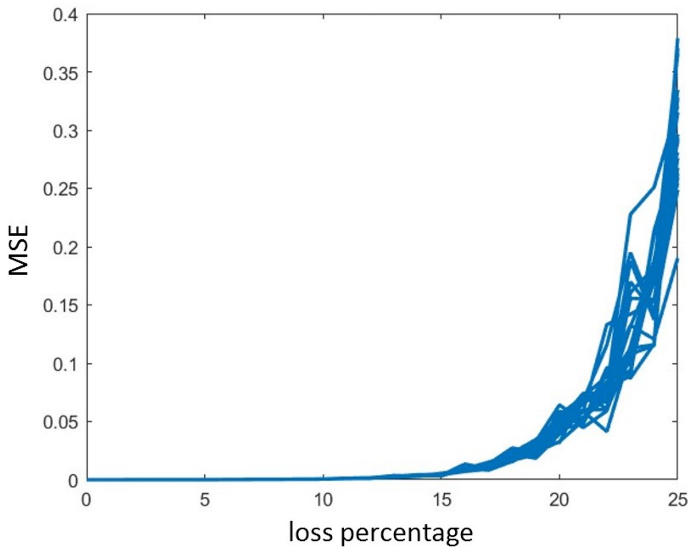


Figure 5.8: Normalized MSE between the sensor and actuator axis angle with increasing packet loss percentage

Based on the simulation, the measurements of the real trajectories, including the axis angles of all six axes and the measured time at the host computer, from the sensor and actuator were implemented in Matlab. Since the data of the sensor and actuator is not recorded synchronously in the time axis, the first step was to synchronize the data. For this purpose, the two data sets were linear interpolated and then sampled with a sampling rate of 1 kHz. This sampling rate assures that there is no large error due to the re-sampling since the data was acquired with a sampling rate of 125 Hz. To compute the delay and R-value of the two trajectories, the methods explained in this chapter were used. The already implemented Matlab function corrcoef() compares two sets of data and gives back the R matrix. This was combined with a circular shift of one data set and computed over the number of samples per data set. This was done for all six axis angles and the average value of the delay, and the R-value represent the corresponding network errors.

5.2 Basic setup as Benchmark

The first measurements were done with fixed wiring. The two robots and the modems were connected over Ethernet cables, as shown in Fig. 5.9 and the network measurements were performed. The RTT measurements were done with

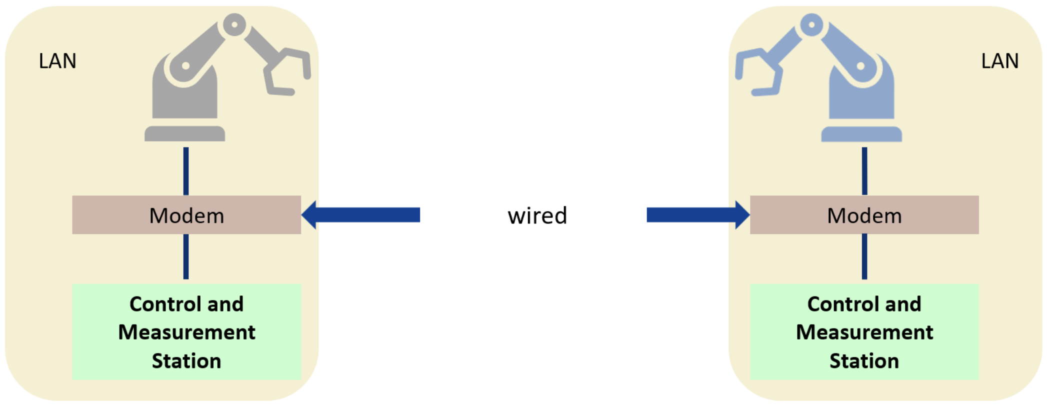

LAN
Modem
Control and Measurement Station
wired
LAN
Modem
Control and Measurement Station

Figure 5.9: Setup for the measurements in the hardwired state

ICMP and TCP packets and the results are shown in Fig. 5.10. It is important to notice, that the delay is not only caused by the transmission itself but also by the internal packet processing time and I/O buffer delay on the devices. These influences have a stronger effect, especially at very low RTT.

Considering this, the RTT of the measured ICMP packets are all under two milliseconds, whereas the TCP packets have a higher RTT with a median of approx. two and a half milliseconds. There are two main reasons for this, firstly ICMP function on the Network layer (L3) of the ISO/OSI model and TCP is categorized as a transport layer (L4) protocol and secondly, the 3-way handshake of the TCP protocol delays the packets further. When examining the ECDF plot, the data is not completely normally distributed but tends to a bimodal distribution, which is however negligible, since this takes place in a range smaller than <1ms and is also heavily influenced by device internal delays.

The Boxplot shows that the spread of the RTT is also increased for TCP packets which mainly leads to the time taken during the handover and the processing time of the TCP server which routes the packets back to the client.

When examining the delay variation shown in Fig. 5.11 the Boxplots for the IPDV show a suitable result to the RTT. The plot of the PDV indicates, that the delay variation is not trending in a direction and the data fits nearly a normal distribution.

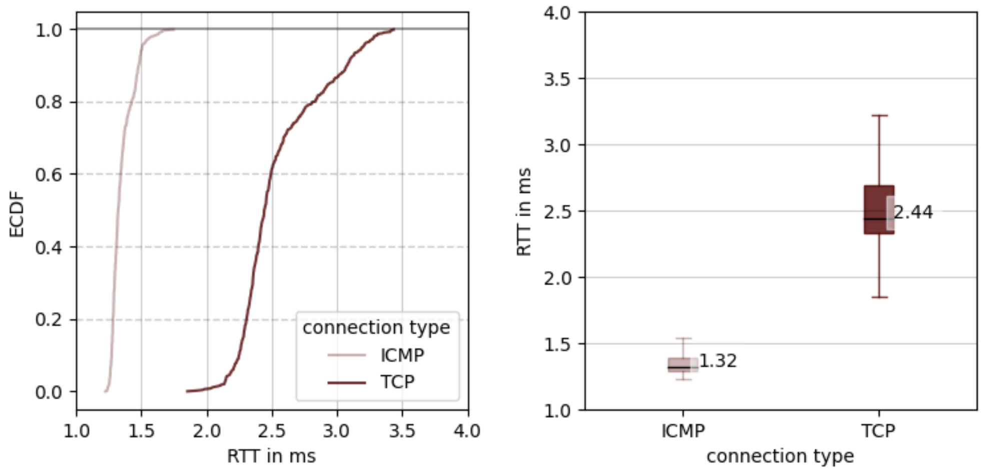


Figure 5.10: RTT for a wired setup with ICMP and TCP measurements plotted as ECDF and as boxplots

It is worth mentioning, that the spread of the data is higher on the upper side of the median, which indicates a gamma distribution. This stems from the above-mentioned device delays and can not be influenced at the physical layer.

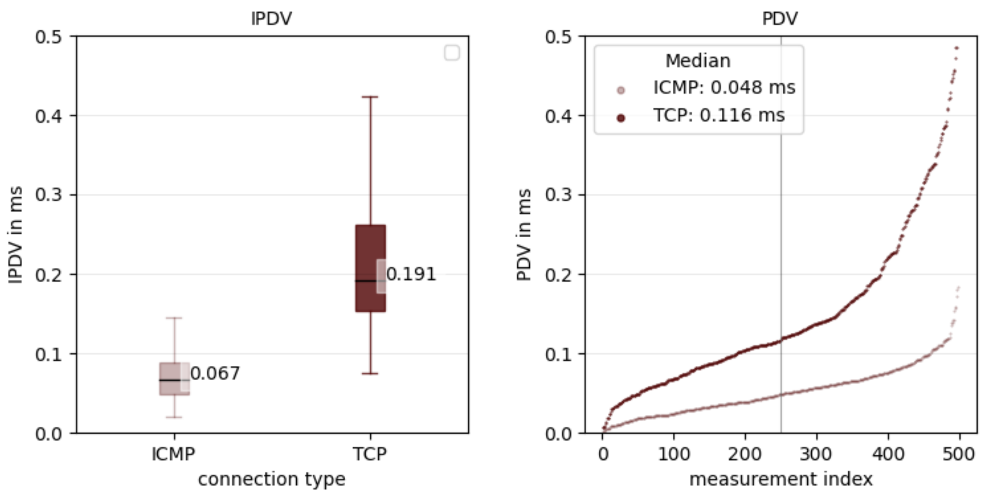


Figure 5.11: IPDV and PDV for a wired setup with ICMP and TCP measurements plotted as Boxplots and as a scatterplot to visualize the data distribution

In the next step, the trajectories of the two robots were recorded and compared with each other. Therefore, the sensor executes a given movement and the actuator follows. The movement lasts 25.7 seconds and uses every axis with a different speed. The recorded axis angles for the sensor and actuator are shown in Fig. 5.12 with the corresponding R values and delay. The R values are all >0.9998, which indicates that, besides a time delay, there is not much of a difference between the sensor and actuator trajectories. The delay is not the same for all six actuator axis and ranges from 0 to 32 milliseconds. This is due to the tracing of speed values instead of axis angles which causes faster and slower tracing according to the axis movement. The zero value on axis three needs additional explanation. This value means, that on average, the actuator reaches the given value simultaneously with the sensor. This is also due to the tracing of speed values and can happen when the axis speed is delayed right when the movement slows down. The actuator overshoots the sensor angle and therefore reaches the upcoming angles quicker or at the same time as the sensor.

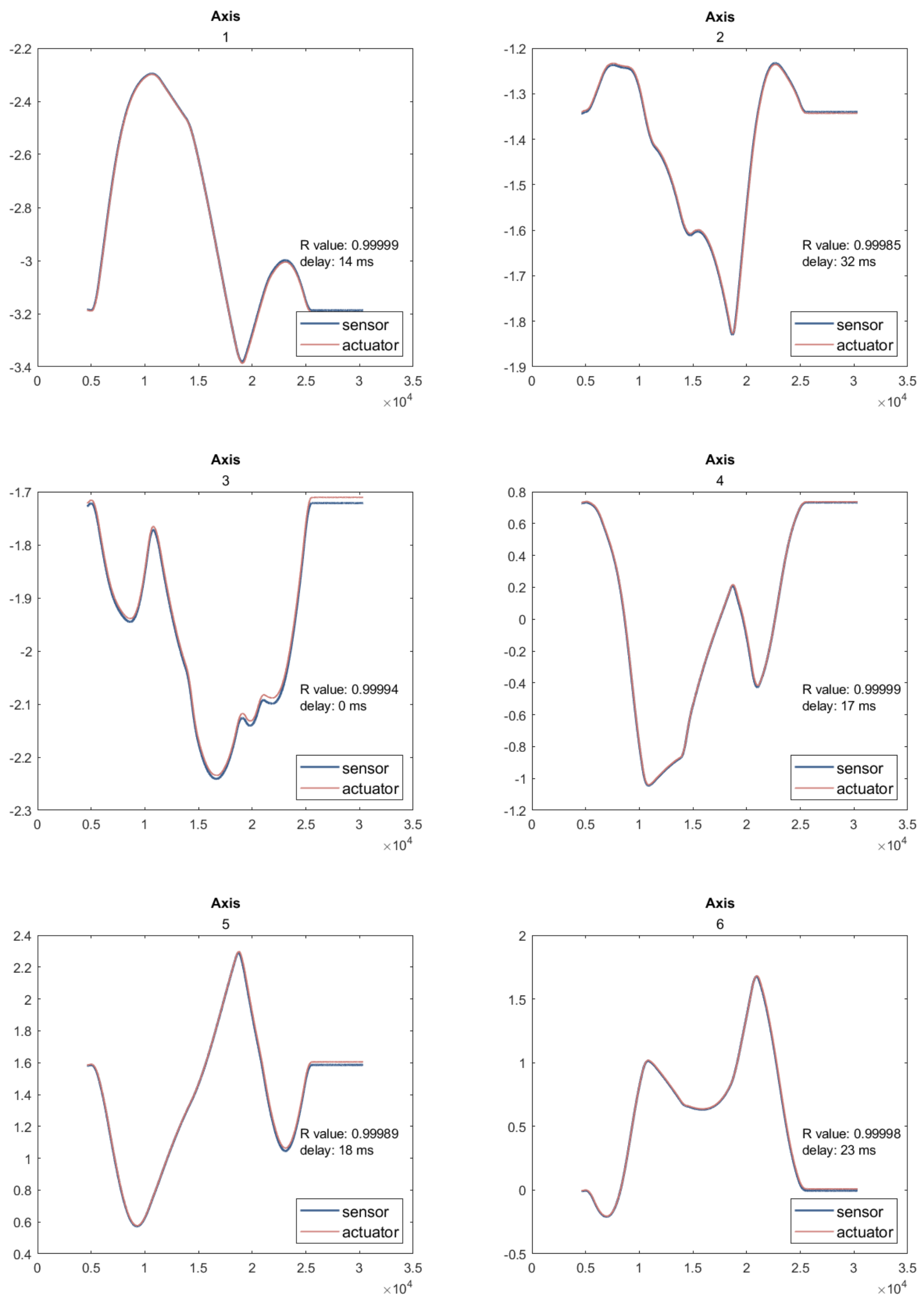


Figure 5.12: Overview of the compared sensor and actuator axis angles in a wired setup with the corresponding R values and delay

A real difference in the axis angles is only visible when looking closer at one angle. Therefore, axis two was used, and in Fig. 5.13 the visualization of the angle from 15 to 20 seconds is shown in detail. Between 18.5 and 19 seconds the overshoot, and the resulting change of the actuator from lagging to leading, as mentioned above is visible.

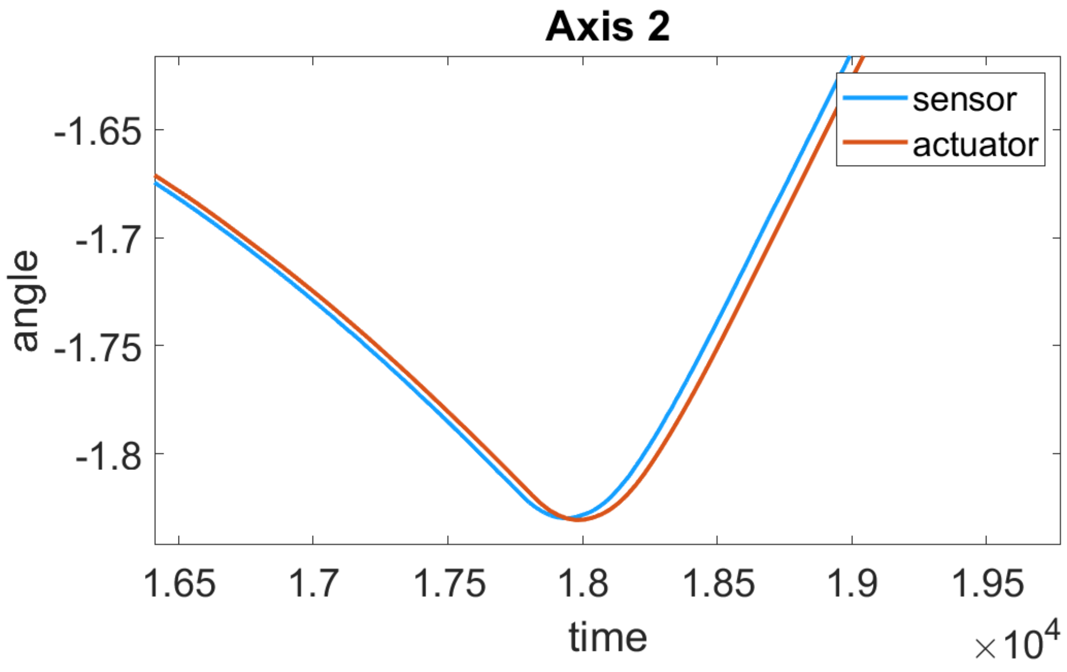


Figure 5.13: Detailed view on one axis to show the delay

The measurement for the wired setup was repeated five times. The mean delay showed 19.33 ms and the R value is 0.999925. For the upcoming measurements, this value can be used as a benchmark to differentiate between the delay and distortion caused by the hardware components and the implementation of the trajectory tracing, and the added interference by the communication setup.

One goal of this work was to find a usable setup for the use-case used. For this purpose, the non-GBR quality of service combinations were considered in advance. It was decided that the 5QI values 5, 8, and 80 would be used for the tests as they use very different priority levels and packet delay budgets as shown in tab. 5.1.

To find out how these settings relate to each other, several tests were performed. The tests were carried out with a packet size of 256 bytes for the test link and an MTU size of 1500 bytes for the network load if not mentioned otherwise and a

5QI Value	Resource Type	Default Priority Level	Packet Delay Budget	Packet Error Rate	Example Service
5	Non-GBR	10	100 ms	10^(-6)	IMS Signalling Video
8	Non-GBR	80	300 ms	10^(-6)	Video Buffered-streaming TCP-based communication
80	Non-GBR	68	10 ms	10^(-6)	Low Latency eMBB applications and Augmented Reality

Table 5.1: The 5QI values 5, 8, and 80 which were used for further testing [7, S. 109-110]

256-QAM was used on the RAN side. The different setups will now be explained in detail.

5.3 Setup 1: Behavior under load

The first setup was used to determine the relationship of communication with each 5QIs to itself and to other 5QIs under stress. For this, a test link was established with two modems that are configured with the same 5QI. To load the network, two additional modems were also connected to the 5G network, and UDP streams were performed on these modems at a specified bandwidth to simulate network traffic. Every UE in the system was configured with the same priority level, to check the behavior of the 5QIs compared to traffic with the same priority level. To provide traffic with a certain bandwidth, the maximum bandwidth was restricted on the 5GC for the two UEs which produced the traffic. This restriction in combination with video streaming produced a steady traffic that simulates real UDP download accurately. The real-time traffic analysis of the modems in Fig. 5.14 shows that the download traffic is steady, and the restrictions are accurate.

ICMP and TCP tests were then performed on the test link to measure the quality of the connection. At the beginning every 5QI was tested without traffic to get a baseline for RTT, packet delay variation, and error rate. After that, the test

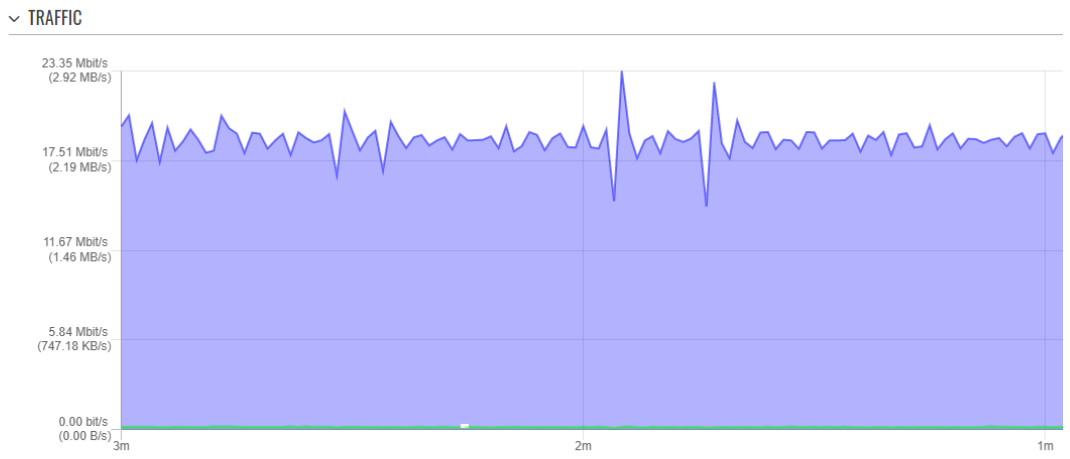


Figure 5.14: UDP real-time traffic log over 3 minutes from one of the modems while performing a video stream with 20 Mbps

link was stressed with traffic of the same 5QI with increasing bandwidth. The last measurements tested a specific 5QI for the test link and applied traffic of one of the other two 5QIs. This was done to check the influence between the 5QIs when they have the same priority level.

One of the first insights was, that the system can handle a bandwidth of approximately 50 Mbps. When increasing the traffic higher than that, the modems get disconnected after a short time and will not reconnect for a certain time. Often was a restart of either the modems or the UPF of the 5GC necessary to establish a connection again. As a result, the load intensity was increased up to a maximum of 50 Mbps.

Tab. 5.2 lists the measurements carried out to provide an overview of everything tested in this setup.

Setup 1.	test link 5QI	network traffic 5QI	load intensity in Mbps
1	5	-	-
2	8	-	-
3	80	-	-
4	5	5	10
5	5	5	20
6	5	5	30
7	5	5	40
8	5	5	50
9	8	8	10
10	8	8	20
11	8	8	30
12	8	8	40
13	8	8	50
14	80	80	10
15	80	80	20
16	80	80	30
17	80	80	40
18	80	80	50
19	5	8	10
20	5	8	20
21	5	8	30
22	5	8	40
23	5	8	50
24	5	80	10
25	5	80	20
26	5	80	30
27	5	80	40
28	5	80	50
29	8	5	10
30	8	5	20
31	8	5	30
32	8	5	40
33	8	5	50
34	8	80	10
35	8	80	20
36	8	80	30
37	8	80	40
38	8	80	50
39	80	5	10
40	80	5	20
41	80	5	30
42	80	5	40
43	80	5	50
44	80	8	10
45	80	8	20
46	80	8	30
47	80	8	40
48	80	8	50

Table 5.2: Measurements carried out for setup 1

5.4 Setup 2: Influence of the packet size

Since the use case can require a diverse size of data packets to be sent, the influence of the packet size was of interest to this thesis. This setup was about determining the packet size for the test link that is least impacted and how the packet size of the traffic is related to that. Therefore, the packet size of ICMP and TCP packets for the test link was changed from 128 to 256, 512, and 1024 bytes and tested, first without traffic and then with a steady packet size of 256 bytes for the traffic. Afterward, the measurements on the test link were repeated, but with 512, 1024, and 1500 bytes packet size for the traffic. To exclude any influence on the measurements due to different 5QI settings, both the test link and the traffic were carried out with a setting of 5QI 80 and the same priority level. The traffic was applied with a steady bandwidth of 20 Mbps and tab. 5.3 gives an overview of the measurements carried out in the setup.

Setup 2.	test link packet size in bytes	traffic packet size in bytes
1	128	no traffic
2	256	no traffic
3	512	no traffic
4	1024	no traffic
5	128	256
6	256	256
7	512	256
8	1024	256
9	128	512
10	256	512
11	512	512
12	1024	512
13	128	1024
14	256	1024
15	512	1024
16	1024	1024
17	128	1500
18	256	1500
19	512	1500
20	1024	1500

Table 5.3: Measurements carried out for setup 2 with a fixed 5QI of 80 for the test link and the traffic

5.5 Setup 3: Influence of the transmission interval

The RTDE interface of the UR5 robots allows the configuration of a sampling rate of up to 200Hz. This led to the need to perform a test setup that tests different transmission intervals. Since the measurements in setups one and two used the approach of only one data packet sent at a time, the measurements needed to be changed. This setup uses the hping3 implementation mentioned in 5 to test the different interval times. Since the behavior of ICMP and TCP showed little difference, apart from the increased RTT, in the setups before and the packet loss is of interest in this setup, only ICMP tests were performed. To perform tests similar to a real implementation, the same 5QIs as well as different 5QIs for test link and traffic were examined with a sending interval from 1 ms, 10 ms, and 100 ms. Since setup one showed the influence between the used 5QI, only 5QI 80 and 5QI 5 were used to test this setup. In addition, the packet size of the test link was altered between 128 bytes and 1024 bytes to examine how this influences the measurements. Tab. 5.4 shows the executed measurements where the traffic was applied with a steady 20 Mbps and a packet size of 1500 bytes.

Setup 3.	test link 5QI	sending interval in ms	test link packet size in bytes	network traffic 5QI
1	5	1	128	80
2	5	2	128	80
3	5	10	128	80
4	5	100	128	80
5	5	1	1024	80
6	5	10	1024	80
7	5	100	1024	80
8	80	1	128	5
9	80	2	128	5
10	80	10	128	5
11	80	100	128	5
12	80	1	1024	5
13	80	10	1024	5
14	80	100	1024	5
15	5	1	128	5
16	5	2	128	5
17	5	10	128	5
18	5	100	128	5
19	5	1	1024	5
20	5	10	1024	5
21	5	100	1024	5
22	5	5	128	80
23	5	10	128	80
24	5	20	128	80
25	5	40	128	80
26	5	5	128	5
27	5	10	128	5
28	5	20	128	5
29	5	40	128	5

Table 5.4: Measurements carried out for setup 3 with a fixed packet size for the traffic of 1500 bytes and a bandwidth of 20 Mbps

6 Results

This chapter presents the measurements and results of the setups explained in 5. Every subsection starts with the presentation of the RTT followed by the IPDV, PDV, and PER. After explaining the results, a distinction is made from case to case whether a trajectory measurement is valuable. If so, this is presented, and the results are explained. The end of this chapter summarizes the most important insights gained and puts the results in relation to each other.

6.1 Results of Setup 1: Behavior under load

Basic performance without traffic

A first measurement was done without traffic for all three 5GI values. Fig. 6.1 represents the RTT as an ECDF and boxplots and shows, that the median is around 30 ms for all three measurements.

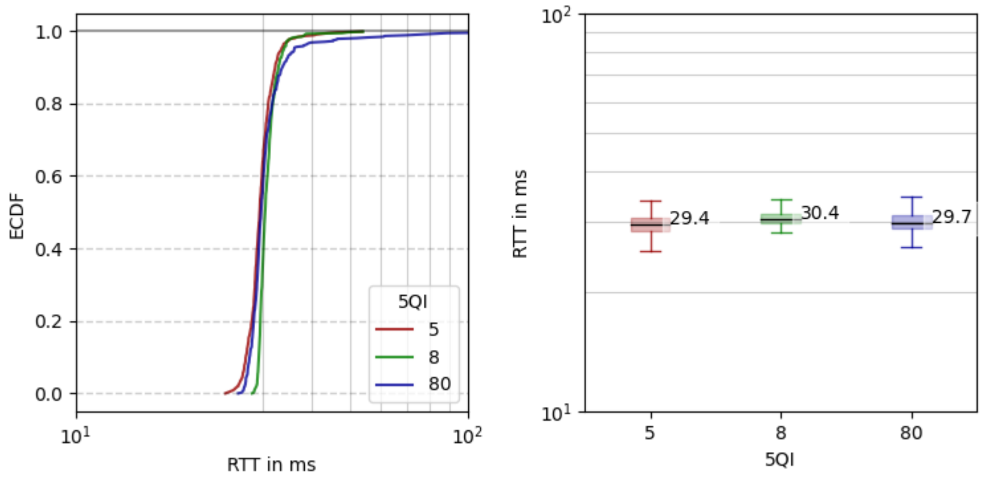


Figure 6.1: ECDF and boxplots of the RTT of the basic 5G setup for the three 5QIs surveyed without traffic

As expected, there is little difference in the measurements since traffic is that of the test link only, and can provide the maximum service for every measurement. At first glance, RTT values seem high compared to the expected latencies of a 5G system. However, as explained in Chapter 5, the RTT metric measured comprises four communication latencies. These are from the UE to the gNB and to the 5G core, plus the inverse path from the 5G core to gNB and the UE. If symmetric communication delays could be assumed, our RTT values amount to around 7.5 ms, in the expected range for a 5G SA system, and in agreement with the values reported in [3]. The distribution is approximately normal with a skew to the right resulting from multiple outliers that are above 40 ms, especially for the measurements of the 5QI 80.

When looking at the IPDV and PDV in Fig. 6.2 we find that the IPDV is lower than 6 ms in 99.65% of the measurements when using the 1.5 IQR rule explained in Ch. 5 and under the assumption that the measurements follow a normal distribution. That assumption holds in the PDV plot, except for the measurements with 5QI of 8 with a much higher variance that prevents a clear fit.

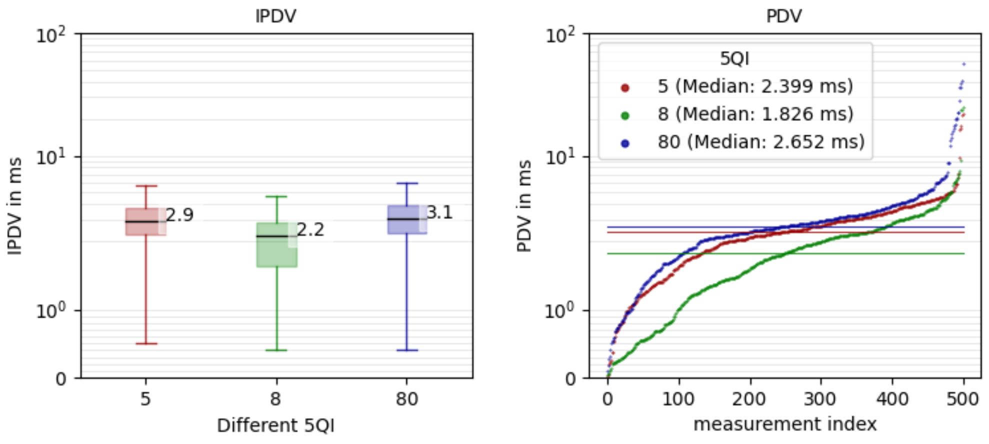


Figure 6.2: IPDV and PDV of the basic 5G setup for the three 5QIs surveyed without traffic

An unexpected result is the high PER measured (Fig. 6.3). When compared to [7], the PER should be around 10^{-6}. The measured values are between 0.14% and 0.02% which is a factor of 10000 higher than what the standard describes. A deeper investigation into the reasons behind this gap led to the research in [3], where it was discovered that many losses occur in the core. To narrow down this behavior further tests are necessary, capable of discriminating losses occurring at the radio interface from those introduced by processing errors inside the core. These tests go beyond the scope of this thesis and will be tackled in follow-up work.

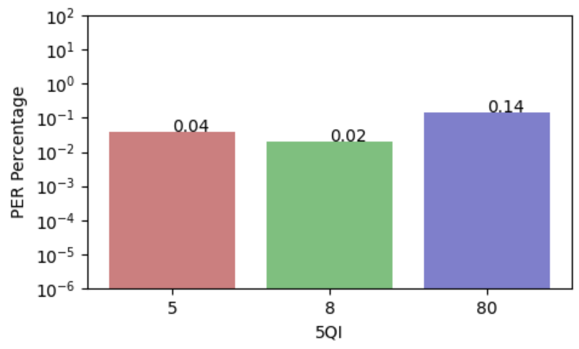


Figure 6.3: PER of the basic 5G setup for the three 5QIs surveyed without traffic

Regarding the real-time trajectory copying through the 5G network, the trajectories for the basic setup measured an average delay of 64.22 ms and a R-value of 0.9999. This is reasonable compared to the wired measurements since the delay is 45 ms higher than the wired setup. These 45 ms results from the 30 ms delay caused by the 5G test link and the sampling rate of the two robots, which is 125 Hz each and thus reads or writes a new value every 8 ms. The R-value is slightly worse than the wired setup since the IPDV is higher. The trajectories for axis one of the three 5QIs without traffic are shown in Fig. 6.4 where an almost perfect match is observed.

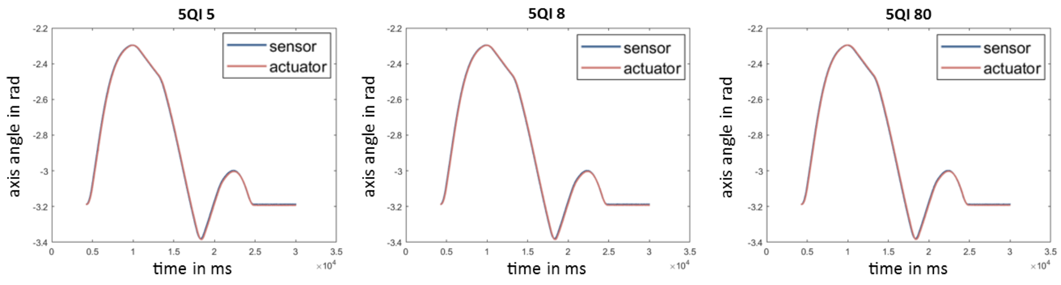


Figure 6.4: Trajectories for axis one of the no-traffic setup for the different 5QIs

6.1.1 Performance with additional network traffic

The next measurements consider adding load to the network through traffic from additional UEs connected to the network, using the same 5QI as the test link.

Since every 5QI showed the same behavior for this setup, only the results for the 5QI 5 are shown in Fig. 6.5 with increasing intensity of the added network traffic from 10 to 50 Mbps.

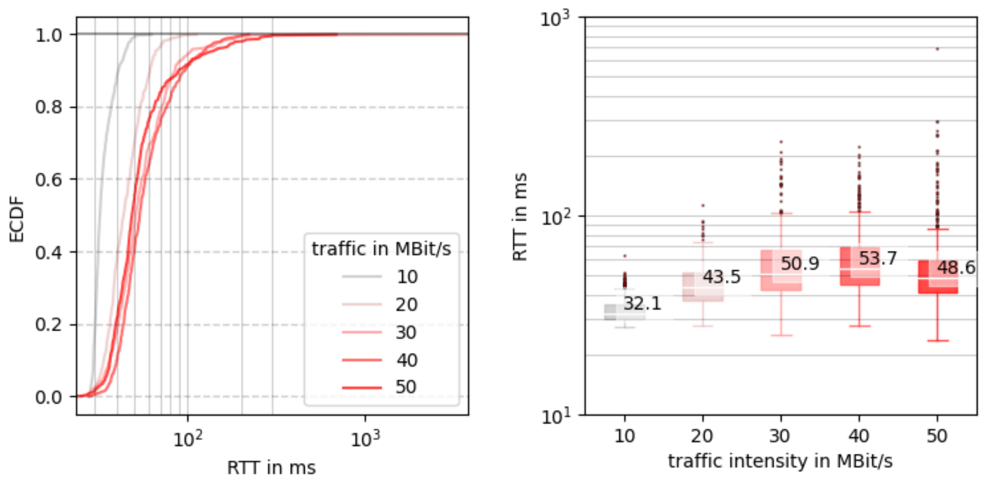


Figure 6.5: RTT for a 5QI of 5 for the test link and the traffic

When increasing traffic from 10 to 30 Mbps the RTT increases from 32 ms to 50 ms, which indicates that individual performance depends strongly on the network load, even under mild traffic loads. This is an observation missing in experimental research available in the literature, that disregards the network load. For loads over 40 Mbps, the median of the RTT stays steady at around 50 ms but the outliers increase. This means that the distribution of the RTT skews more to the right and therefore the tendency to higher RTTs increases.

The small drop in RTT from 40 to 50 Mbps traffic can be attributed to two effects:

- When loading the network over its capacity, packets get dropped. When test link packets with a high RTT drop, the average delay decreases, with a corresponding increase in the PER.
- When the network gets overloaded, traffic behavior changes from a steady flow to bursts. Between these bursts, a relatively low number of packets gets consumed by the traffic hosts. This can lead to an increasing capacity for the test link.

The IPDV and PDV show similar behavior to the RTT. In Fig. 6.6, the IPDV shown on the left side increases from 4.6 ms up to 15.7 ms at 40 Mbps traffic. At 50 Mbps traffic, the IPDV drops to 12.7 ms, which may also stem from the above explanations. The plot of the PDV gives additional insight into the distribution of the data and shows that with increasing traffic, the skewness to the right increases to a certain degree.

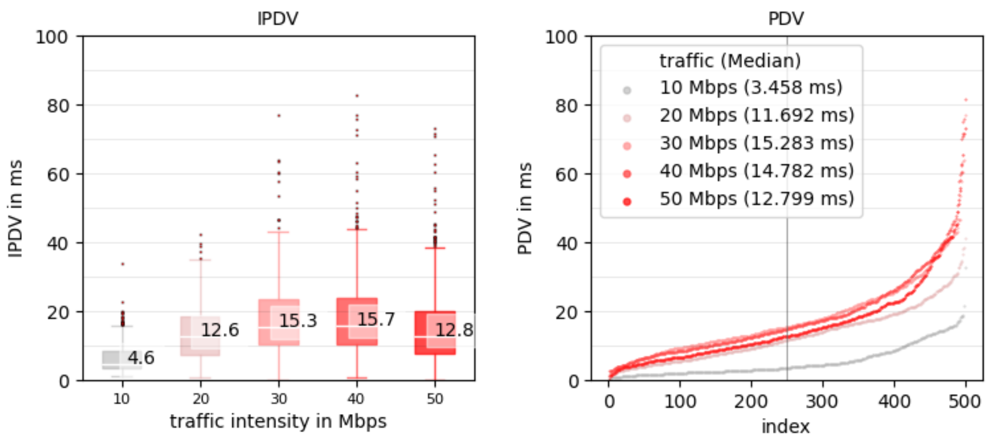


Figure 6.6: IPDV and PDV for a 5QI of 5 for the test link and the traffic

When looking at the PER in Fig. 6.7, the initial 10 Mbps of traffic do not show any noticeable change compared to the unloaded tests. From then on, however, the PER increases drastically to a maximum value of 3.56 percent. This means that 178 of the 5000 packets sent were lost. Since the network should not be busy with traffic of less than 50 Mbps, further investigations are necessary to explain the high PERs for these measurements.

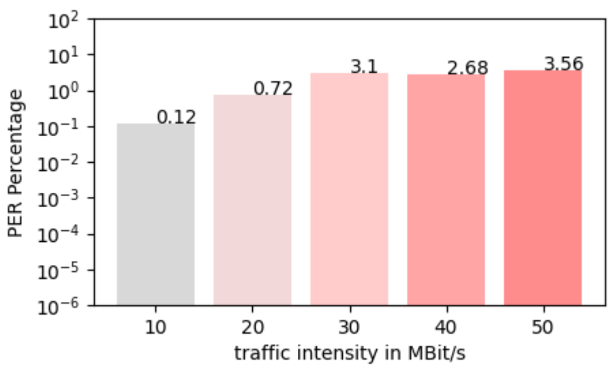


Figure 6.7: PER for a 5QI of 5 for the test link and the traffic

These results show that a network load with the same 5QI as the test link increases the RTT and IPDV to a certain intensity of traffic. This behavior can change when the network is at its maximum load because the traffic is no longer stable. However, the PER increases steadily and is much higher than assumed in the theory.

To find out how traffic load affects the behavior of the actuator, the trajectories were recorded during increasing traffic. In Fig. 6.8, the trajectories of axis one for the sensor and actuator with increasing traffic are shown. The trajectories show a correlation between the measurements of the RTT, IPDV, PER, and the R-values and delay of the trajectories. With 10 Mbps traffic, the trajectory is almost the same as with no traffic. However, if the traffic is increased, a sharp increase in the delay is noticeable. It is also visually recognizable how the trajectory is distorted at the actuator and the R-value decreases as a result. These results confirm the insights from [5], which show a deviation of the axis speed with high network traffic applied.
To investigate the interaction of different traffic types, tests between the different 5QIs were done with applied traffic of 40 and 50 Mbps as these correspond to high network load and are thus of most practical interest.

Starting with a test link with a 5QI of 5, traffic with 5QI 8 and 80 were applied. Fig.6.9 shows that neither traffic with 5QI 8 nor 80 has a major influence on the RTT, IPDV, or PER of the test link. The results are even better than the measurements where the test link and traffic have the same 5QI. This is because the 5QI 5 has the highest priority of the three and therefore there is little to no influence from the others. Nevertheless, this is interesting because when configuring the 5QI profiles, a manual priority of one was assigned to all UEs used. Thus, it can be determined that, when two UEs with the same configured priority are using the same medium with limited resources, the UE with the higher default priority wins.

The second test considered a 5QI of 80 for the test link. 5QI 80 has a higher default priority than 8 and a lower default priority than 5, which means, that different behavior should be determined. Fig. 6.10, shows the results from these measurements. It is clear to see that these assumptions were correct, as the median of the RTT, shown in part b of the Fig. 6.10b, increases up to 117.8 ms against traffic with a 5QI of 5 and the ECDF plot shows a heavily one-sided distribution where a traffic of 5QI 8 has only a minimal effect on the test link. The IPDV and PDV plots show a similar result, with an average IPDV of 33.8 against 50 Mbps of 5QI 5 traffic. In addition, the PDV plot shows that the data tends towards a bimodal distribution with an accumulation of values under 10 ms and around 45 ms. This can also be attributed to the behavior of the traffic, which changes from a steady stream to

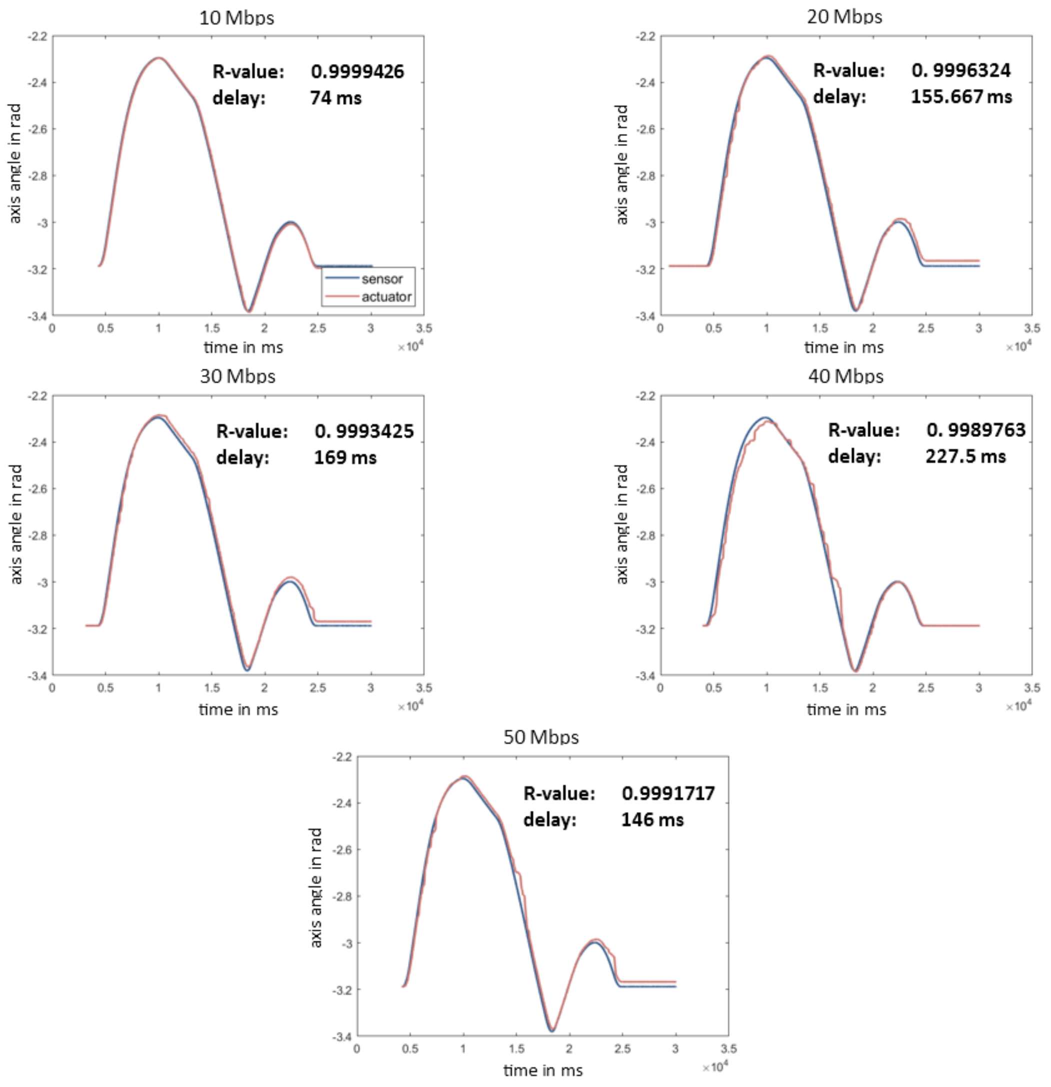
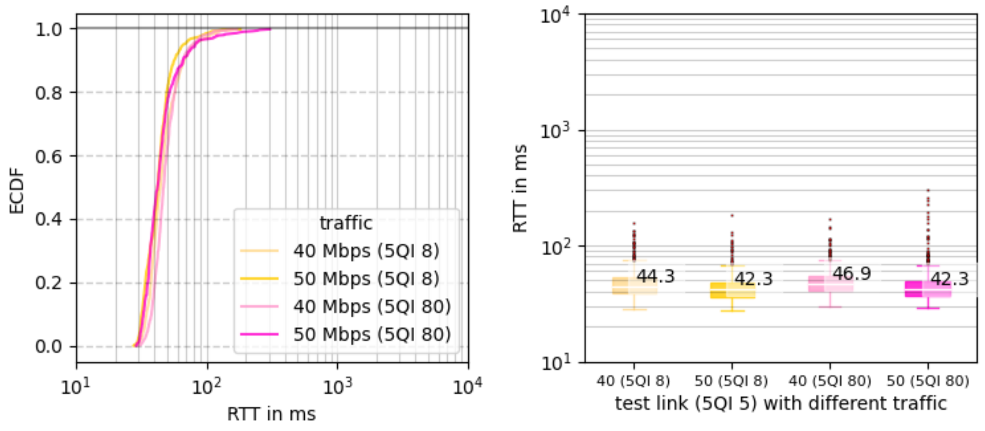
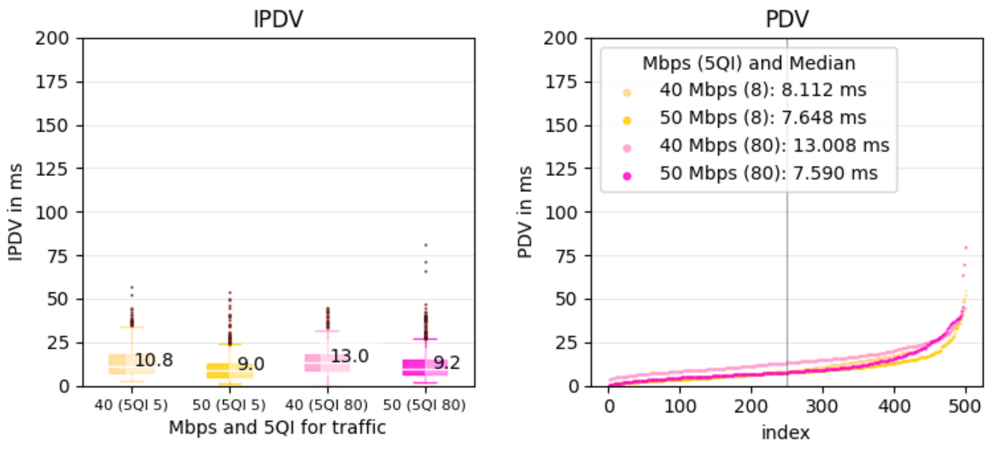


Figure 6.8: Trajectories for axis one with increasing traffic from 10 to 50 Mbps where the delay and the R-values are noted

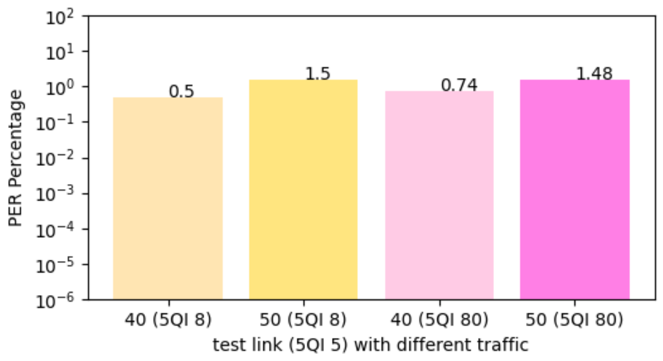
bursts at rates close to the maximum network capacity. The PER is shown in part c of the Fig. 6.10c confirms the previous findings and shows that a communication with a test link of 5QI 80 has an unusable quality against traffic with a 5QI of 5, but against traffic with 5QI 8 a relatively stable test link could be measured.



(a) RTT for test link with 5QI 5 and different traffic setup

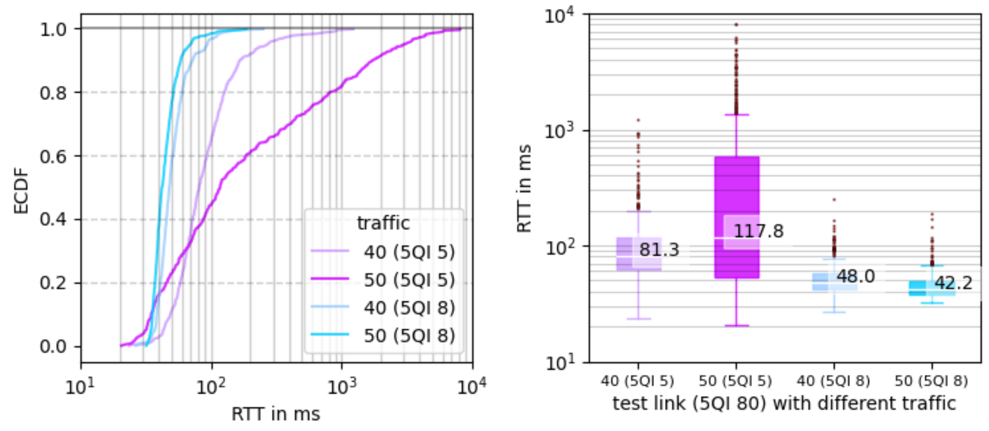


(b) IPDV and PDV for test link with 5QI 5 and different traffic setup

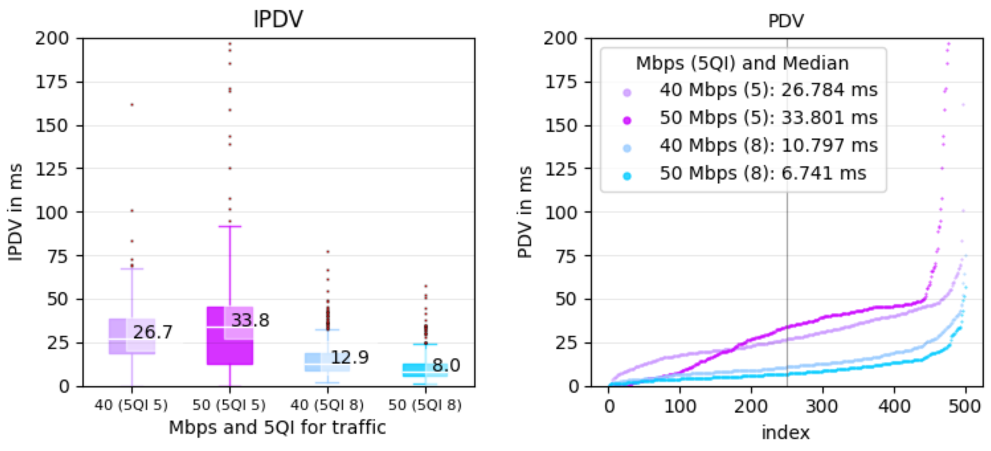


(c) PER for test link with 5QI 5 and different traffic setup

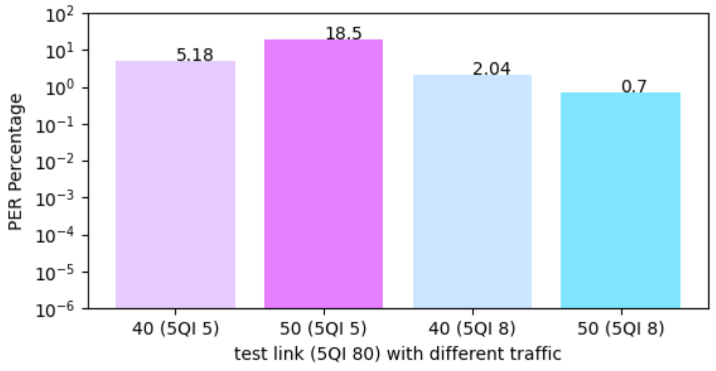
Figure 6.9: Results for the tests with 5QI 5 for the test link and 5QI 8 and 80 for the traffic



(a) RTT for test link with 5QI 80 and different traffic setup



(b) IPDV and PDV for test link with 5QI 80 and different traffic setup

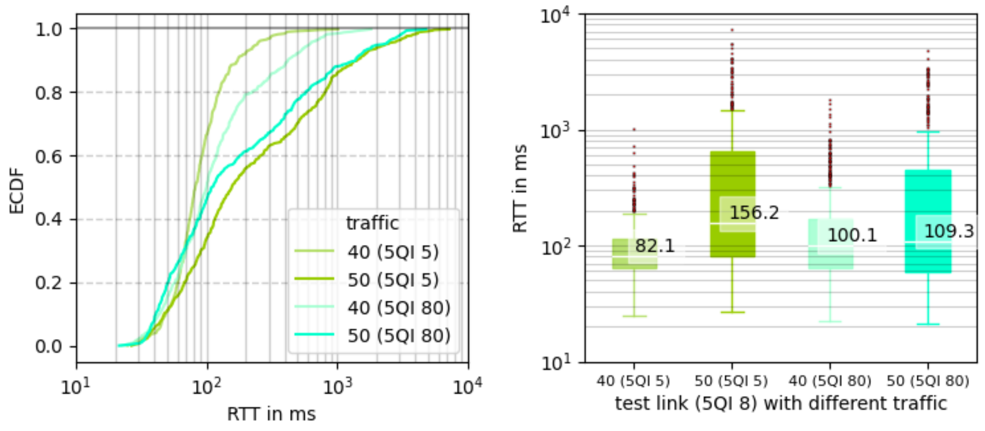


(c) PER for test link with 5QI 80 and different traffic setup

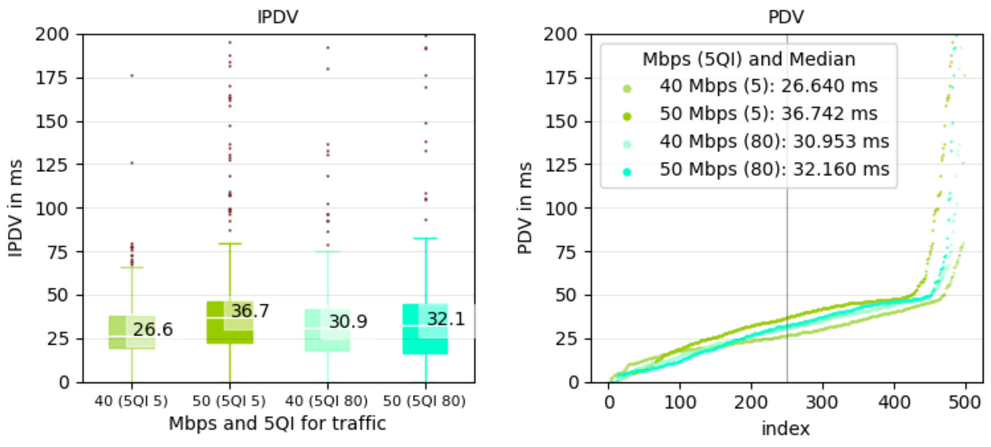
Figure 6.10: Results for the tests with 5QI 80 for the test link and 5QI 5 and 8 for the traffic

To have a comprehensive evaluation of the interaction between 5QIs, Fig. 6.11 shows the results of a final test performed with a 5QI of 8 for the test link and 5QI of 5 and 5QI of 80 for the traffic. The results for these tests are consistent with those of the previous measurements. Since the test link has the lowest priority, it is strongly influenced by both traffic configurations and shows a very high RTT as well as IPDV and PDV. The distribution of the RTT becomes more and more right skewed with increasing traffic and the delay variation changes again to a bimodal distribution at 50 Mbps traffic. The PER also strongly increased, which makes the usability of 5QI 8 for automation purposes highly questionable.

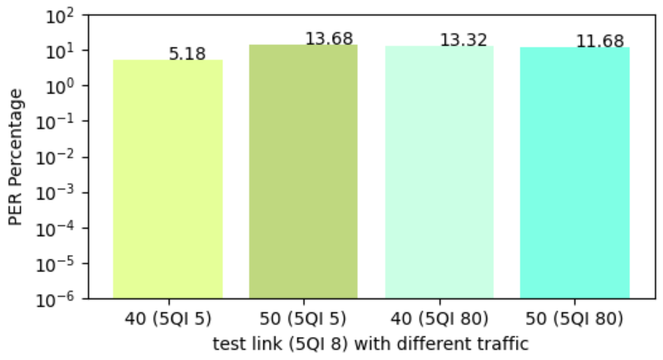
In addition to the ICMP tests, TCP tests were performed to analyze the behavior of connection-oriented communication. For this, all 5QI pairings were tested with the same setups as the ICMP tests. Based on the results from the ICMP tests, results only for a traffic of 50 Mbps are presented in Fig. 6.12. The RTT, IPDV, and PDV show the same behavior for the different pairings as in the ICMP tests, but with significantly higher delay and delay variation. It should be noted that due to the large change between the setups, the y-axis for the PDV plot was chosen to be logarithmic. However, the distribution of the PDV measurements of these setups is similar to that of the ICMP tests. If the IPDV and PDV values in Fig. 6.12b are considered, it is recognized that these are significantly higher than the values of the ICMP tests. This is due to the fact that when retransmitting packets, the delay variation can change significantly. Since the ICMP tests showed error rates of more than 10%, every tenth packet must be re-transmitted and thus the delay fluctuates strongly. These results underline the importance of the right choice of 5QI pairing to ensure a stable connection.



(a) RTT for test link with 5QI 8 and different traffic setup

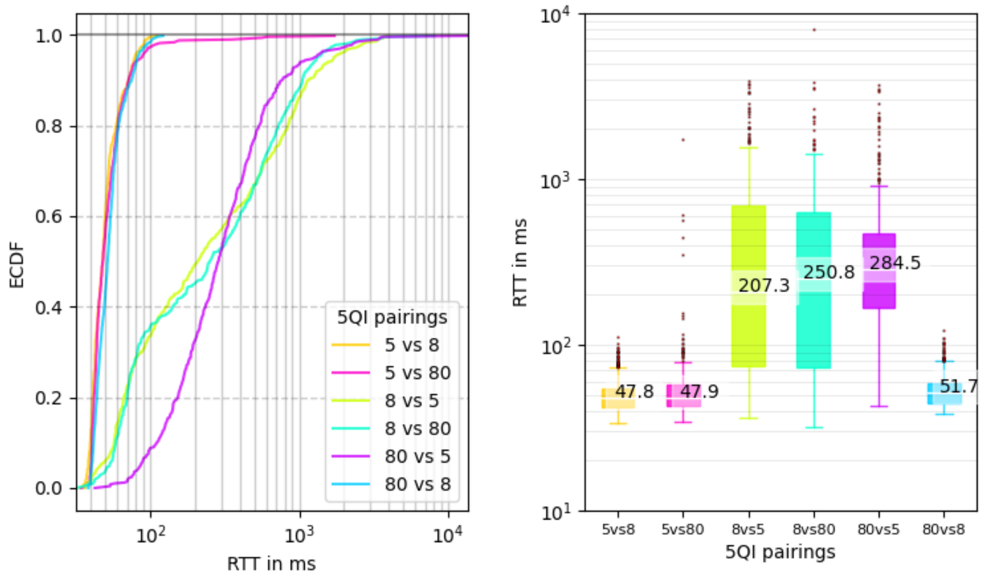


(b) IPDV and PDV for test link with 5QI 8 and different traffic setup

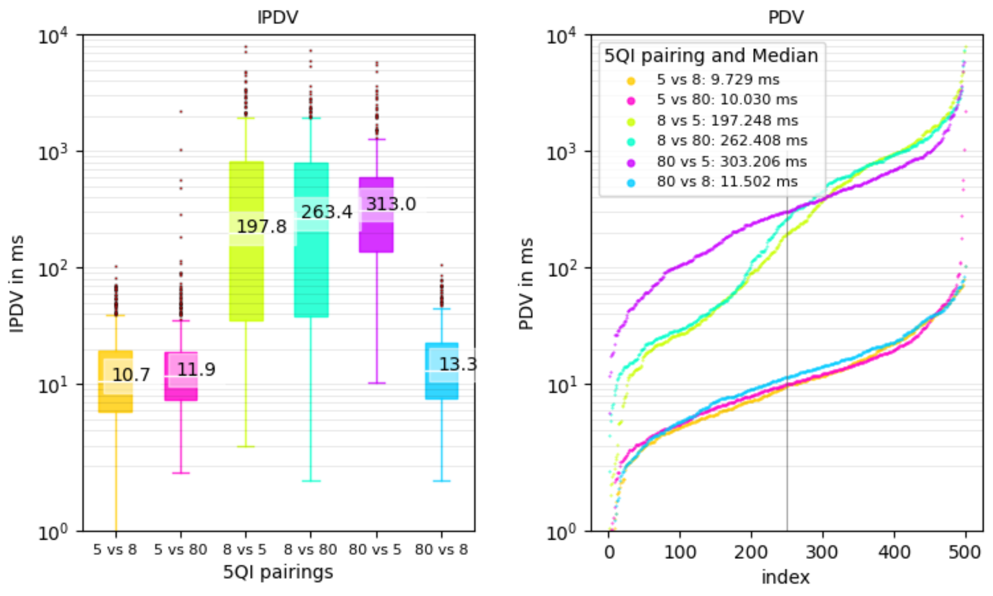


(c) PER for test link with 5QI 8 and different traffic setup

Figure 6.11: Results for the tests with 5QI 8 for the test link and 5QI 5 and 80 for the traffic



(a) RTT for TCP packets with different 5QI test link and traffic pairings



(b) IPDV and PDV for TCP packets with different 5QI test link and traffic pairings

Figure 6.12: Results for the TCP tests with different 5QI test link and traffic pairings with a traffic intensity of 50 Mbps

The last tests for TCP communication were done to connect the use case with the measured values and find out, what this means for the movements of the two robots. Towards that end, tests were carried out with two different setups. One with a test link with 5QI 5 against traffic of 5QI 8 with increasing intensity. Based on the measurements already performed in Fig. 6.9, it can be expected that this setup is very well suited for recording the trajectory, and a high R-value and low delay can be expected. The second measurement was performed with a test link with 5QI 8 and a traffic of 5QI 5 with increasing intensity. The measurements already performed can be seen in Fig. 6.11 and suggest that the conditions here are the other way around. For this setup, a very bad R-value and a high delay are to be expected.

In Fig. 6.13 the test link with 5QI 5 is shown on the left and the test link with 5QI 8 is shown on the right both with increasing traffic from 20 to 50 Mbps. The R-value and delay for the trajectory on the left setup is relatively stable up to 50 Mbps traffic whereas the quality of the right setup decreases rapidly. This goes so far that recording an entire trajectory for the tests at 50 Mbps was not possible because the connection was constantly interrupted as seen in the picture in the right lower corner, where the trajectory could only be recorded for eight seconds.

These tests lead to the conclusion that the 5QI settings and the default priority of a UE profile, makes a crucial difference with respect to RTT, IPDV, PDV, and PER. These should be assigned to the individual UEs in accordance with the use case and the expected network load, to give priority to critical applications during high network utilization.

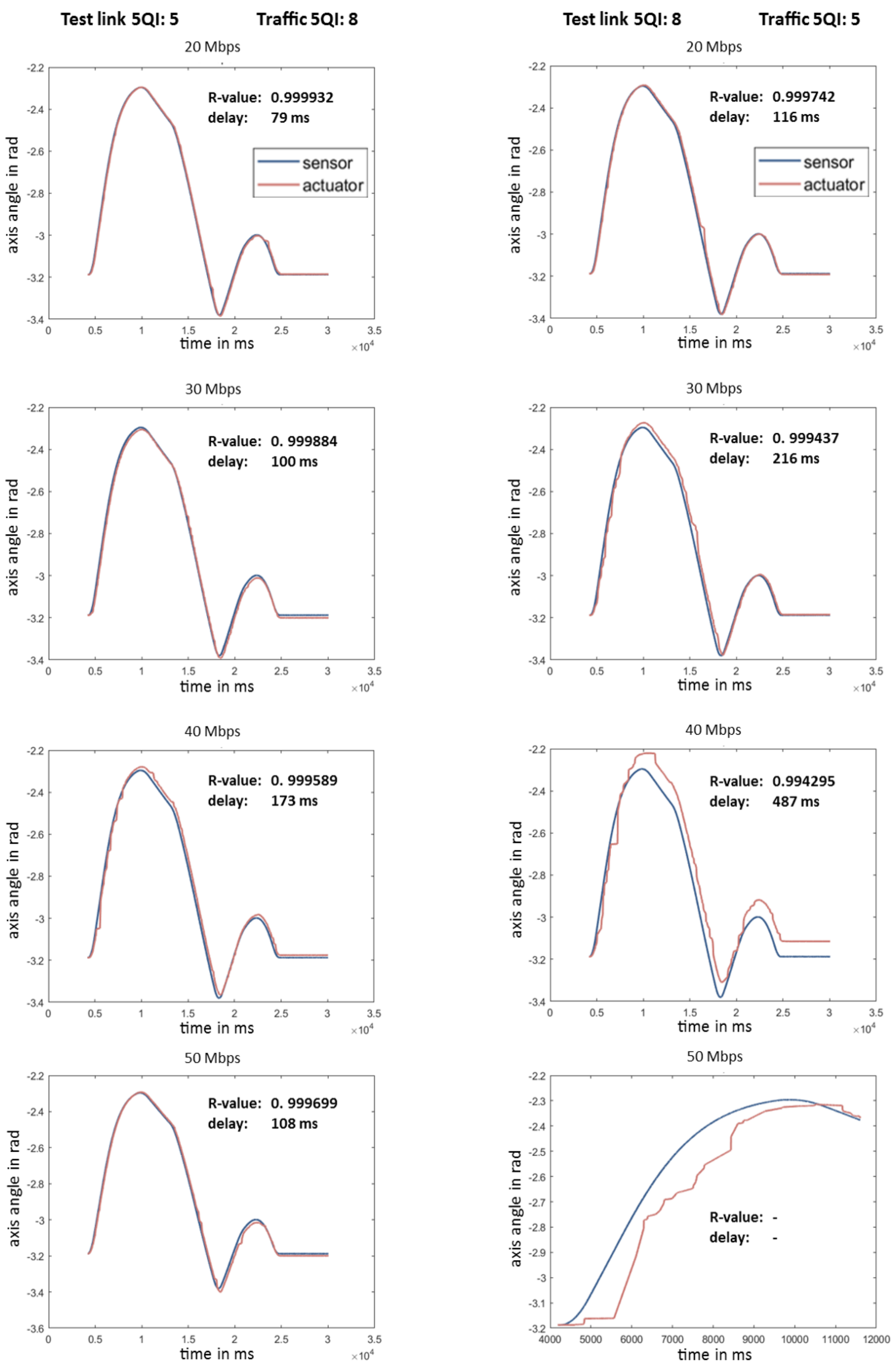


Figure 6.13: Trajectory comparison with increasing traffic where on the left setup the traffic has a small impact and, on the right, the traffic has a big impact on the test link.

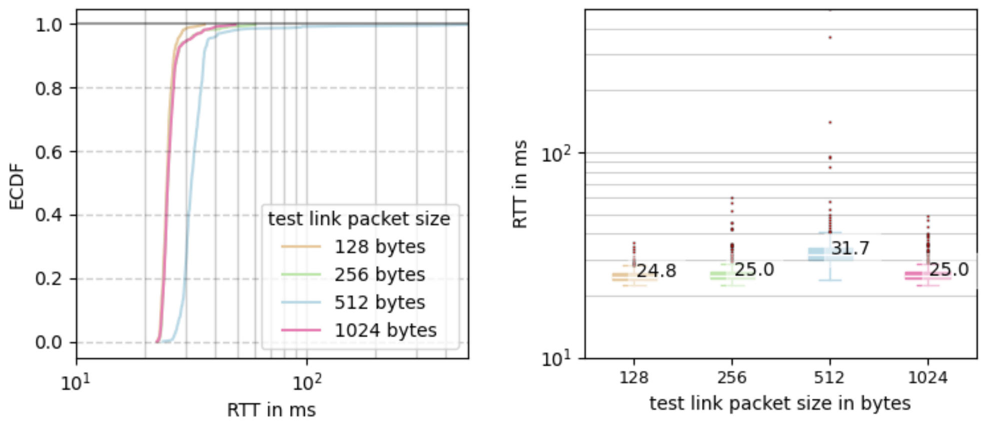
6.2 Results of Setup 2: Influence of the packet size

The results in this section show how the packet size influences the 5G test link. It is emphasized that the measurements are as close as possible to reality and therefore only certain packet sizes were examined. The first setups correspond to ICMP tests and were done with a traffic packet size of 256, 512, 1024, and 1500 bytes. First, without applied traffic for baseline performance in Fig. 6.14, that show a slightly higher RTT and delay variation for packets of the size of 512 bytes which can be the result of measurement errors.

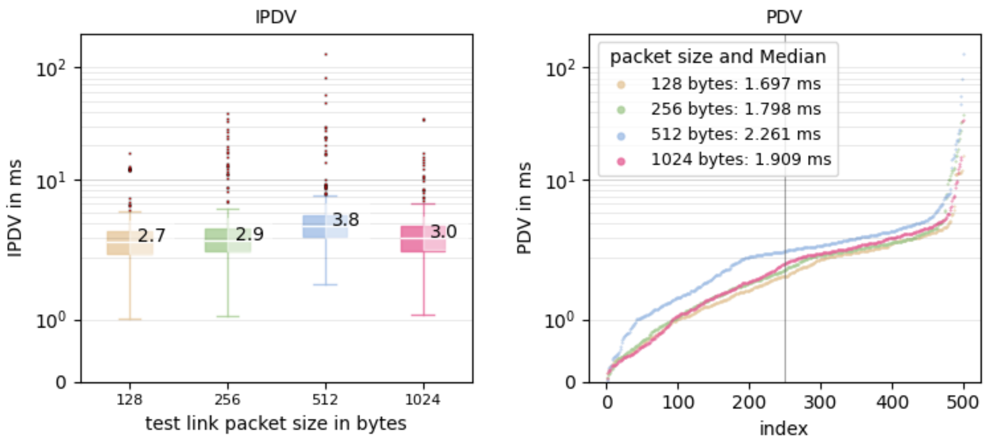
The next step was adding traffic and examining the behavior of the test link under stress. The results in Fig. 6.15 show the RTT for these measurements with ascending packet size for traffic starting with the smallest packet size in Fig. 6.15a.

These results show, that the RTT for 1024-byte long packets is significantly higher than for the other packet sizes tested. This is somehow intuitive because longer packets are more difficult to schedule. It can be said that packets with 1024 bytes take longer to be transmitted in every setup but especially against traffic with smaller packets the difference is significant. Furthermore, it can be said that traffic with a packet size between 512 and 1024 bytes cause the greatest delay at the test link. This leads to the assumption that the system can handle either small or large packets very well but struggles with processing traffic of medium sized packets. This behavior needs further investigation. If the ECDF plots in Fig. 6.15b and Fig. 6.15c are examined more closely, the behavior of packages with a size of 128 bytes shows a tendency towards a bimodal distribution. This behavior is comparable to the results in [4] and can be attributed to the RAN. Since the used n41 frequency band uses TDD and therefore a time multiplexing method, it can happen that the asymmetrical up- and download time slots cause this distribution type.

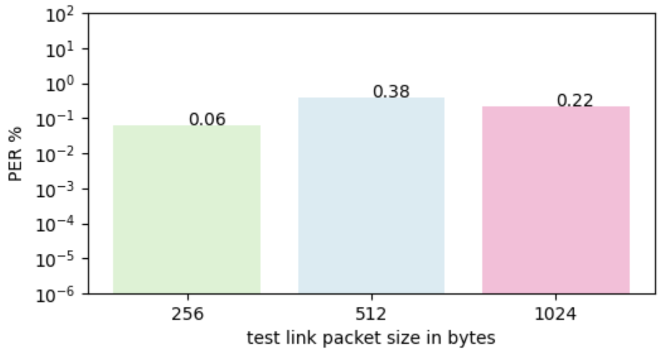
The delay variation in Fig. 6.16 shows a similar behavior and thus confirms the findings from the RTT results that the 1024-byte packets are poorly forwarded by the system. It should also be noted that a packet size of 512 or 1024 bytes has the greatest influence on the test link whereas traffic with 128 and 1500 byte show nearly no additional distortion of the test link.



(a) RTT for ICMP packets with different packet sizes

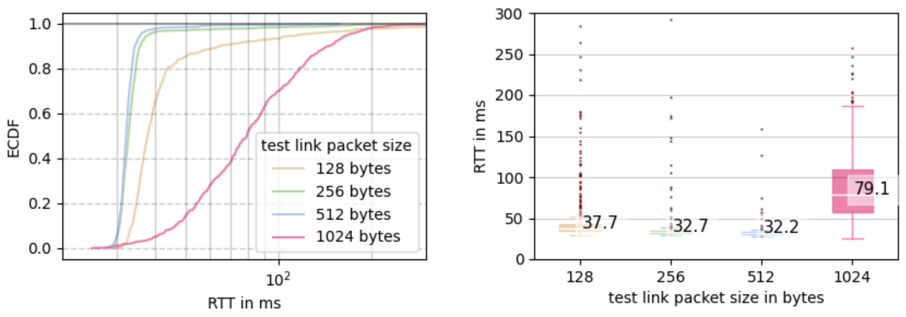


(b) IPDV and PDV for ICMP packets with different packet sizes

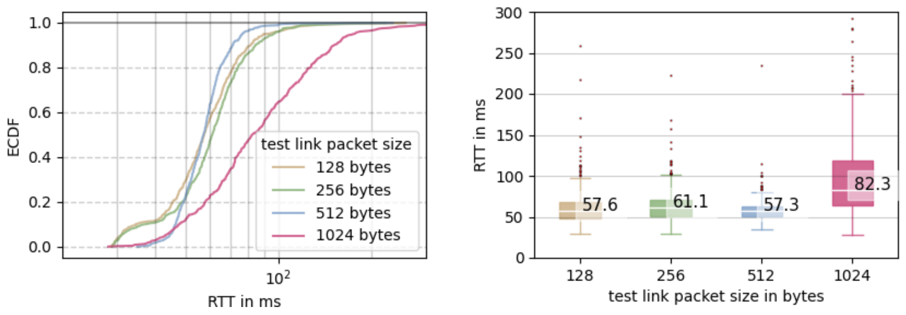


(c) PER for ICMP packets with different packet sizes

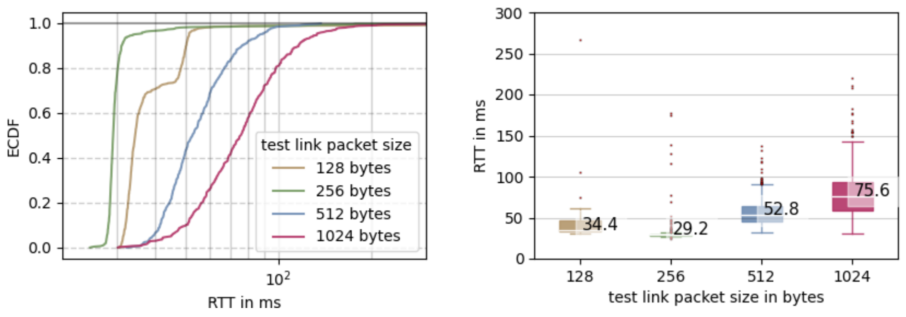
Figure 6.14: Results for the ICMP tests against with no traffic



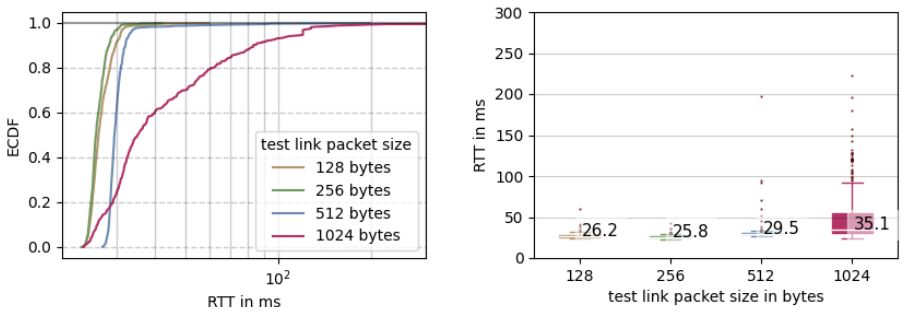
(a) RTT for ICMP packets against traffic with a packet size of 256 bytes



(b) RTT for ICMP packets against traffic with a packet size of 512 bytes

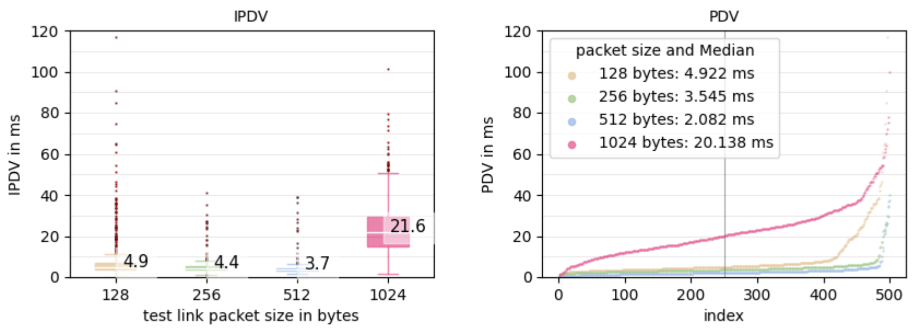


(c) RTT for ICMP packets against traffic with a packet size of 1024 bytes

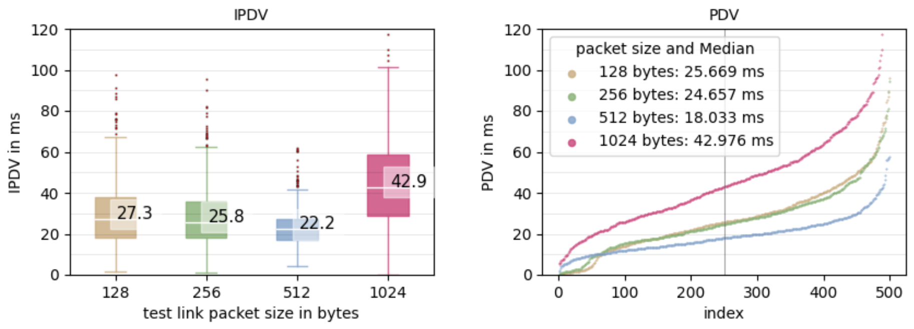


(d) RTT for ICMP packets against traffic with a packet size of 1500 bytes

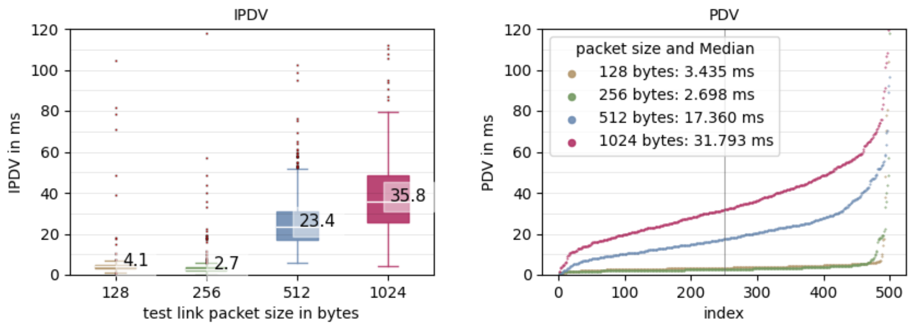
Figure 6.15: RTT for the ICMP tests against different traffic packet sizes with a traffic intensity of 20 Mbps



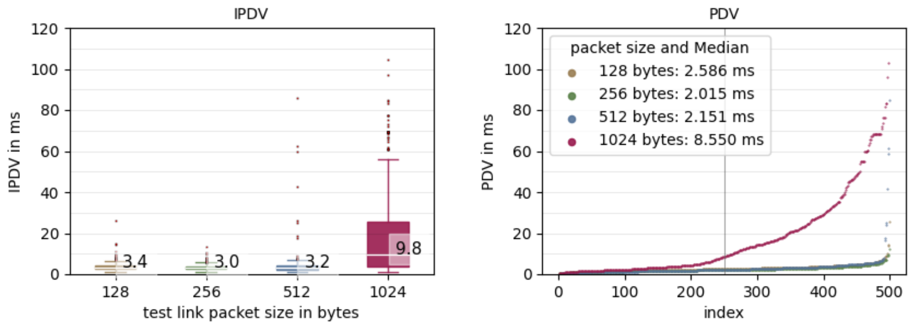
(a) Traffic packet size of 256 bytes



(b) Traffic packet size of 512 bytes



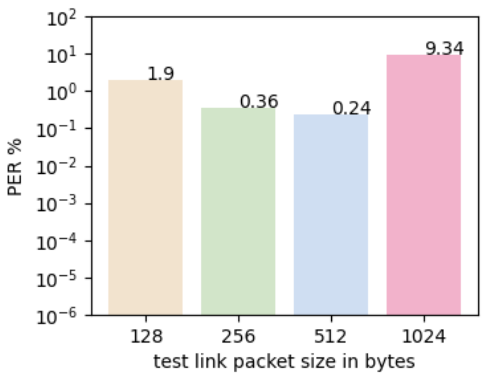
(c) Traffic packet size of 1024 bytes



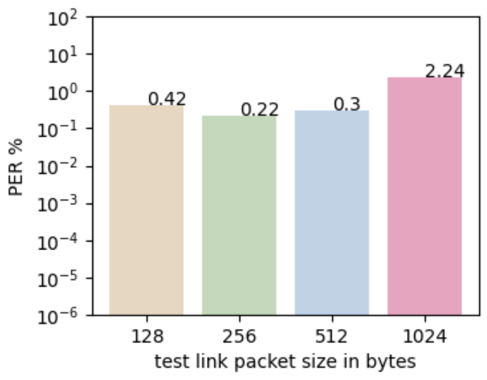
(d) Traffic packet size of 1500 bytes

Figure 6.16: IPDV and PDV for the ICMP tests against different traffic packet sizes and with a traffic intensity of 20 Mbps

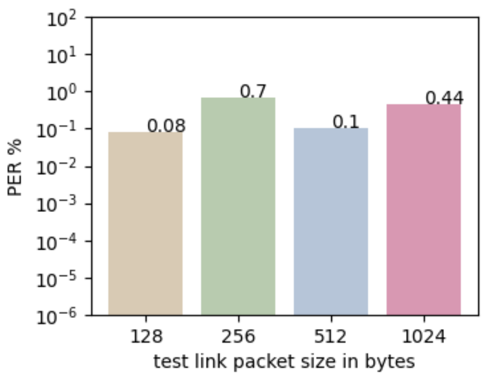
The ICMP test results are completed with the PER measurements in Fig. 6.17 and shows that a big packet size for the test link is not only slow and highly varying but also has a high error rate. The remaining results of the PER measurement are consistent with the insights already obtained and show that the packet size of the test link is steady above a certain size. It has to be noted that in Fig. 6.17d only three bars are plotted since the measurements for a test link packet size of 256 bytes did not show any packet errors.



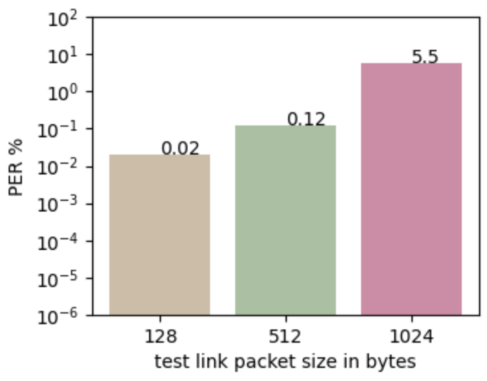
(a) Traffic packet size of 256 bytes



(b) Traffic packet size of 512 bytes



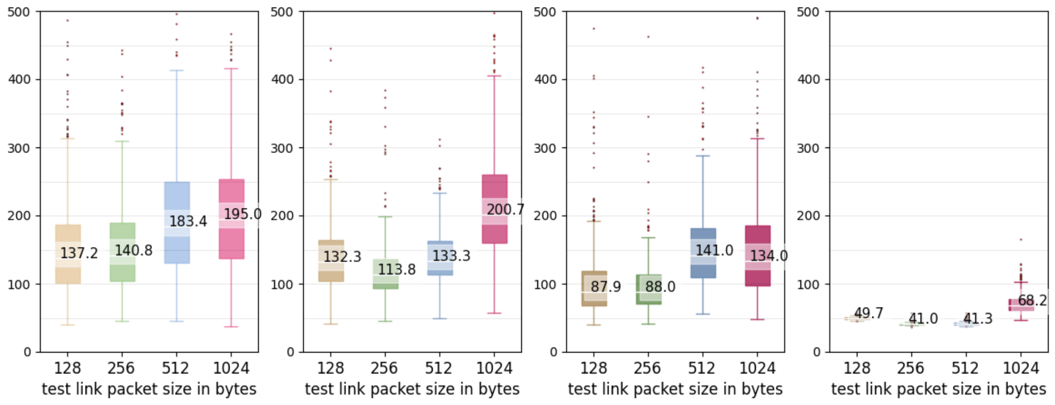
(c) Traffic packet size of 1024 bytes



(d) Traffic packet size of 1500 bytes (PER is zero for a test link packet size of 256 bytes)

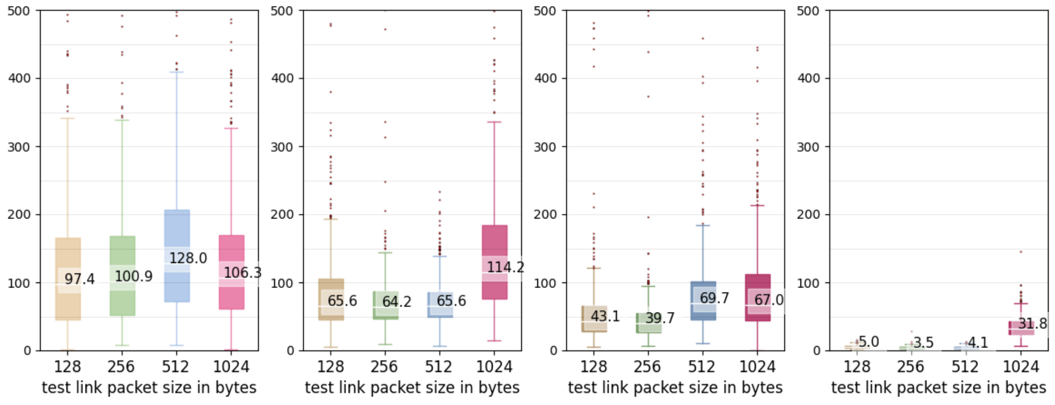
Figure 6.17: PER for the ICMP tests against different traffic packet sizes and with a traffic intensity of 20 Mbps

In addition to the ICMP tests, TCP tests were performed with the same setups. The results are summed up for the RTT in Fig. 6.18 and for the IPDV in Fig. 6.19 and show one significant difference.



(a) Traffic packet size of 256 bytes (b) Traffic packet size of 512 bytes (c) Traffic packet size of 1024 bytes (d) Traffic packet size of 1500 bytes

Figure 6.18: RTT for the TCP tests with a different packet size for the test link and the traffic with a traffic intensity of 20 Mbps



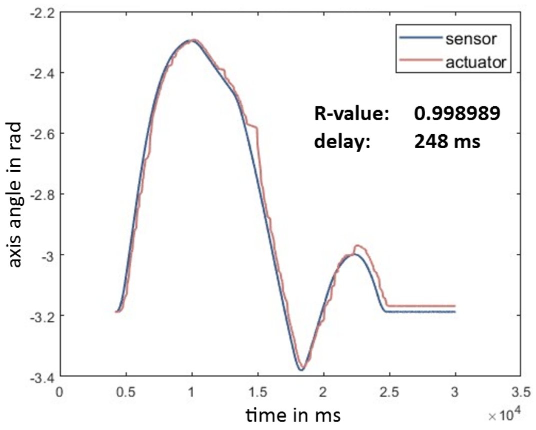
(a) Traffic packet size of 256 bytes (b) Traffic packet size of 512 bytes (c) Traffic packet size of 1024 bytes (d) Traffic packet size of 1500 bytes

Figure 6.19: IPDV and PDV for the TCP tests with a different packet size for the test link and the traffic with a traffic intensity of 20 Mbps

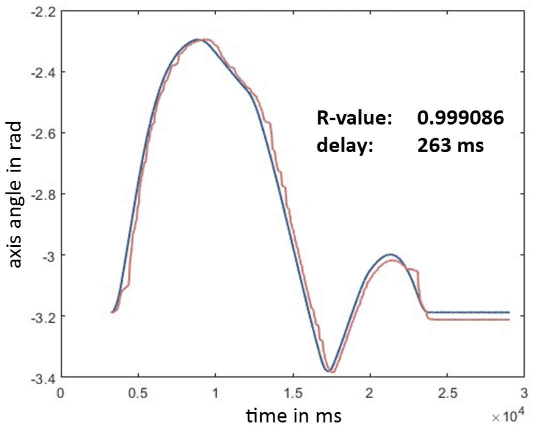
Compared to the ICMP tests, the results for a traffic packet size of 256 bytes show a higher RTT and IPDV. The RTT is visible in Fig. 6.18a and can be compared with fig 6.15a and the IPDV in Fig. 6.19a can be compared with fig 6.16a where the measurements for the ICMP tests for traffic of a packet size of 256 bytes are shown.

This behavior may be related to the higher PER values but needs further investigation. The rest of the measurements have the same information content as the ICMP tests, only with higher values, which is in agreement with the results from 6.1.

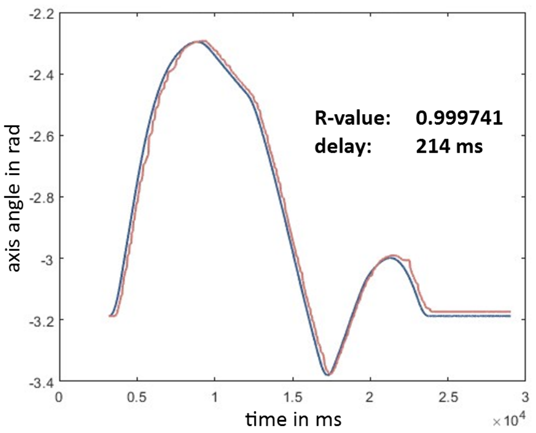
The TCP tests have shown that the RTT and IPDV are relatively high compared to the tests for the behavior under load when the packet size for the traffic is chosen smaller than 1500 bytes. To test the impact on the use case, the control program for the robots was modified to send data packets of different sizes from the sensor to the actuator while recording the trajectory. Starting with a test link packet size of 128 bytes, the results for different test link packet sizes are shown in Fig. 6.20. As expected, the delay is the lowest and R-value the highest when the network is loaded with traffic that has a packet size of 1500 bytes. With increasing packet size the R-value rises and the delay drops except for a 512 bytes traffic packet size. The change from 1024 bytes to 1500 bytes packet size for the traffic results in a significant delay reduction by a third of the measured delay.



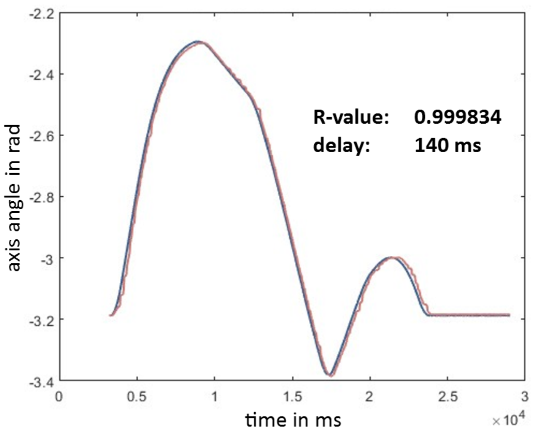
(a) Traffic packet size of 256 bytes



(b) Traffic packet size of 512 bytes



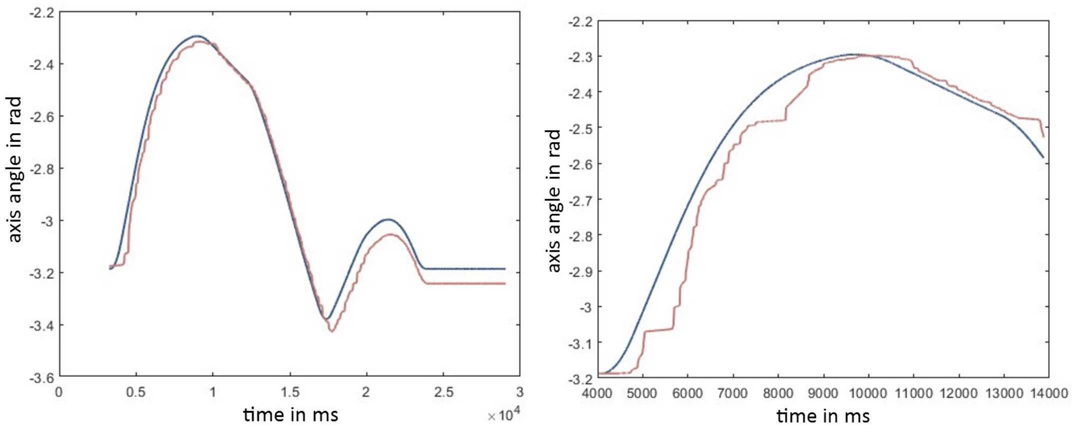
(c) Traffic packet size of 1024 bytes



(d) Traffic packet size of 1500 bytes

Figure 6.20: Trajectory comparison with a packet size of 128 bytes for the test link and varying packet sizes for the traffic

The measurements with 512 bytes traffic packet size had to be redone several times since the connection was very unstable in this setup. The robots either lost consecutive packets and therefore distorted the tracing flow or the modems lost connection and the test link got interrupted as a result. The first measurement trajectories are visible in Fig. 6.21 and show the interrupted connection on the left and a distorted result on the right. The next step was to increase the packet size for the test link to 256 bytes and repeat the measurements with the results shown in Fig. 6.22. The trend is consistent with the previous measurements, where a higher packet size for the traffic increases the R-value and decreases the delay. For the measurement with a traffic packet size of 512 bytes in Fig. 6.22b, no whole trajectory could be recorded, even after repeated testing, which also supports the previous conclusion.

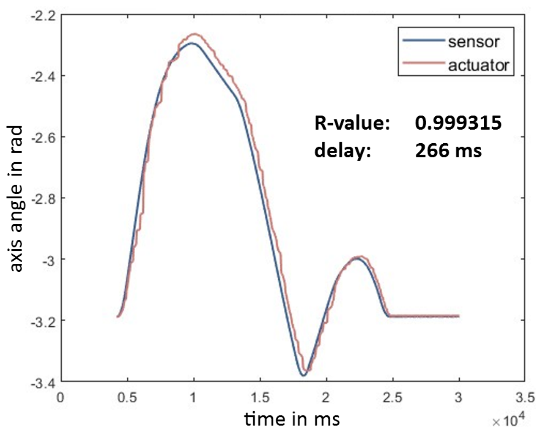


(a) Distorted measurement because of high packet loss (b) Faulty measurement because of complete connection loss

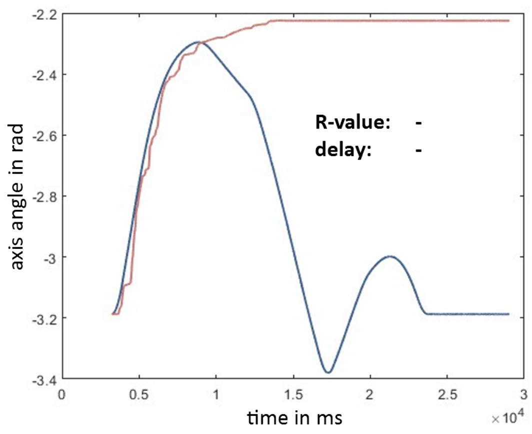
Figure 6.21: Measured errors for the trajectory with a packet size of 128 bytes and 512 bytes packet size for the traffic

The tests with a packet size of 512 and 1024 bytes for the test link were so unstable that even after multiple tests no trajectories could be calculated and therefore cannot be presented. This is probably due to the fact that the sending interval of 125 Hz for the sensor and the actuator in combination with the increasing packet size leads to errors. Thus the connection is only partially usable.

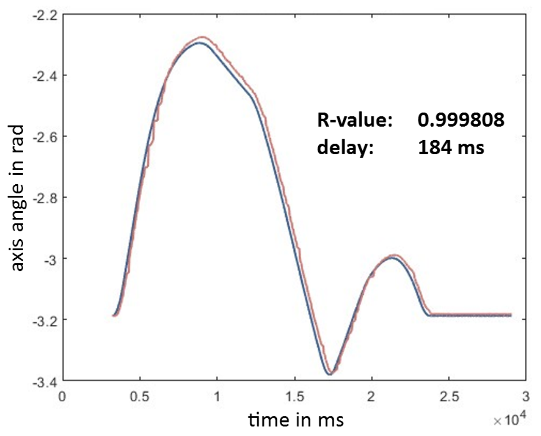
The results of this setup lead to the conclusion, that the system in principle can handle a few bigger packets for the traffic better than many smaller packets but certain setups need to be tested before being used for automation purposes. The connection problems with a packet size of 512 bytes for the traffic are particularly noticeable, which requires further tests and investigations to understand this behavior.



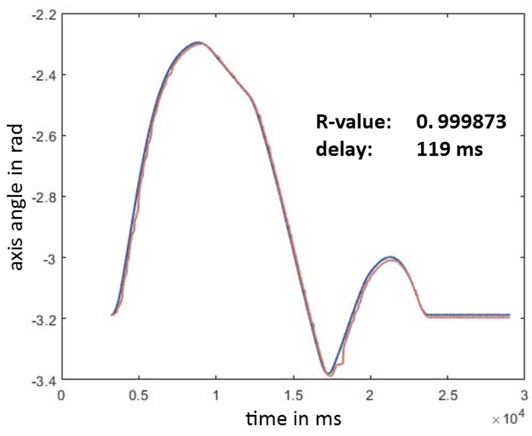
(a) Traffic packet size of 256 bytes



(b) Traffic packet size of 512 bytes



(c) Traffic packet size of 1024 bytes



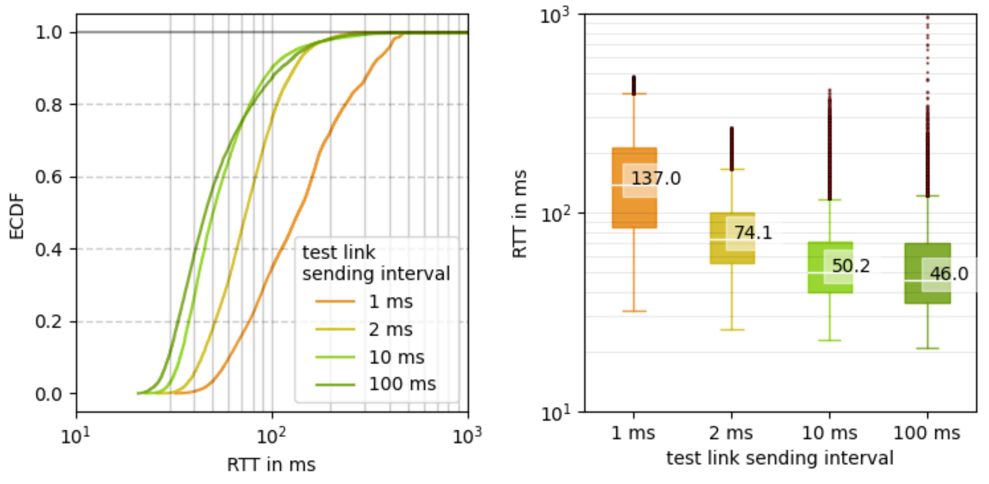
(d) Traffic packet size of 1500 bytes

Figure 6.22: Trajectory comparison with a packet size of 256 bytes for the test link and varying packet size for the traffic

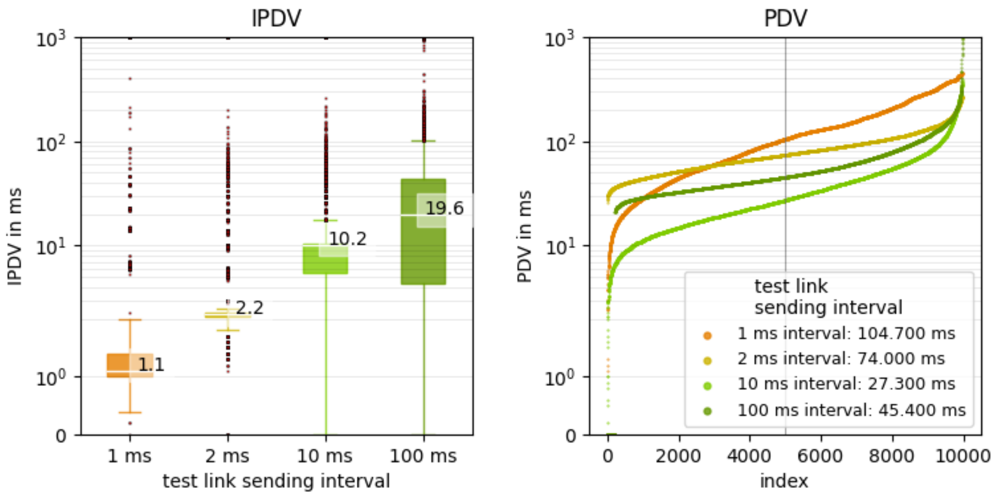
6.3 Results of Setup 3: Influence of the transmission interval

After exploring the quality of service configurations and the influence of different packet sizes, the transmission interval was also of interest. This metric is especially important in machine-to-machine communication since the sending intervals can be programmed and therefore it can be directly linked to the use case. Initially, the test link was configured with a 5QI of 5 and the traffic with a 5QI of 80. This configuration was chosen because the previous measurements showed that the influence on the test link is very small. In Fig. 6.23 are the results shown for a packet size of 128 bytes for the test link, where a sending interval of one, two, ten, and one hundred milliseconds was chosen. The RTT shows that a fast interval leads to a higher delay between the sensor and actuator. Thus, it can be concluded that the data packets are delayed and that becomes a bottleneck if they are sent too fast. If the IPDV in Fig. 6.23b is considered, it can be concluded that the faster intervals have much better behavior than the slower ones. However, looking at the PDV, results show significantly worse behavior at send intervals of one and two milliseconds. The PER shows zero percent for one and ten milliseconds and Fig. 6.23c shows the remaining values.

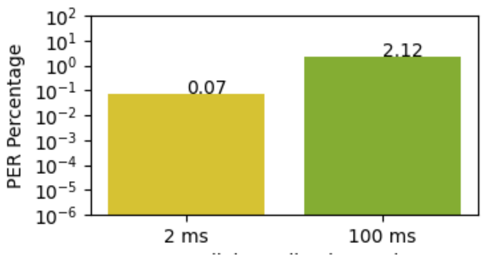
The overall picture shows that with very fast transmission intervals a problem arises which cannot be explained more precisely based on these plots. For this purpose, further analyses of the data were carried out and presented in a histogram, in Fig. 6.24 on the left, and as a time series in Fig. 6.24 on the right. These plots can be used to explain the results better. The histograms show that the spread of the data is very wide for fast intervals and very narrow for slow intervals. The scatter plots on the right explain why that happens. Data does not have steady RTT values but is transported intermittently, resulting in a sawtooth-like signal. Like the bimodal distribution during the packet size tests, this relates to the used multiplexing method. In [3], similar behavior was observed with a frequency band using TDD at high transmission intervals, which is attributed to the reservation and expiry of a communication channel. This intermittent data transmission can lead to problems, especially with machine-to-machine communication, which runs through fixed periodic cycles. A possible solution to this problem could be decreasing the sending interval or changing from TDD to FDD.



(a) RTT for a test link different sending intervals

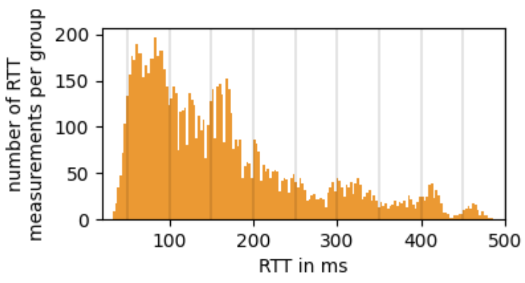


(b) IPDV and PDV for a test link with different sending intervals

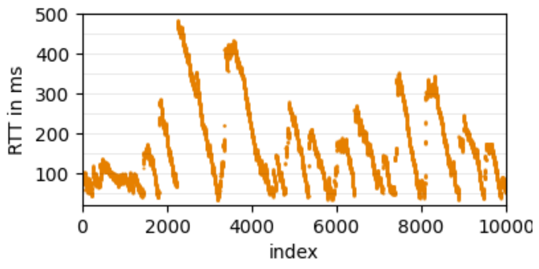


(c) PER for test link with different sending intervals

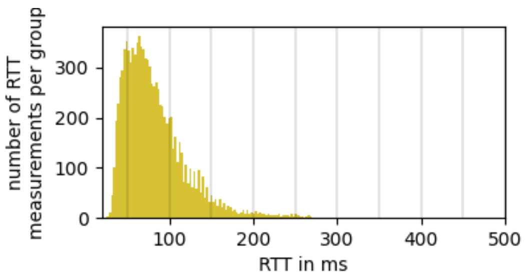
Figure 6.23: Results for testing different sending intervals with 5QI 5 and a packet size of 128 bytes for the test link and a network load of 20 Mbps with 5QI 80



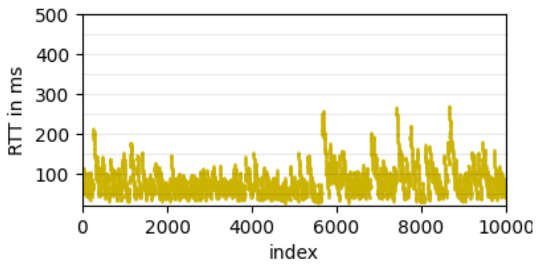
(a) Histogram of the RTT for a sending interval of 1 ms



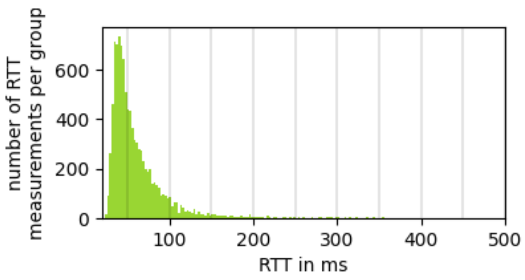
(b) Scatter plot of the RTT for a sending interval of 1 ms



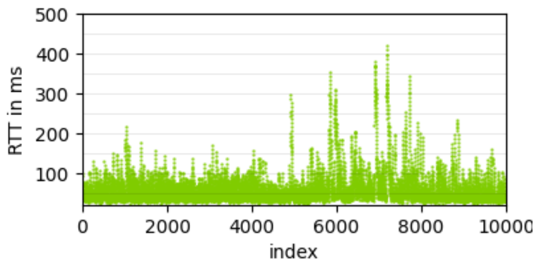
(c) Histogram of the RTT for a sending interval of 2 ms



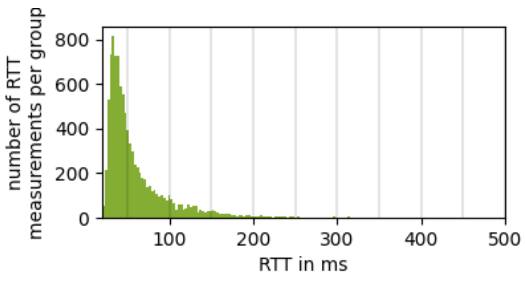
(d) Scatter plot of the RTT for a sending interval of 2 ms



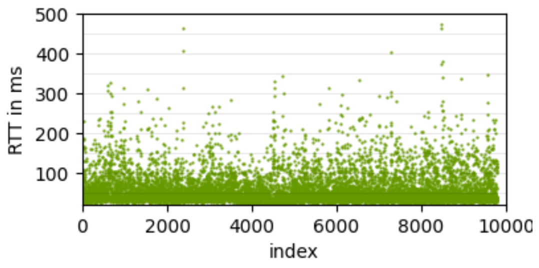
(e) Histogram of the RTT for a sending interval of 10 ms



(f) Scatter plot of the RTT for a sending interval of 10 ms



(g) Histogram of the RTT for a sending interval of 100 ms



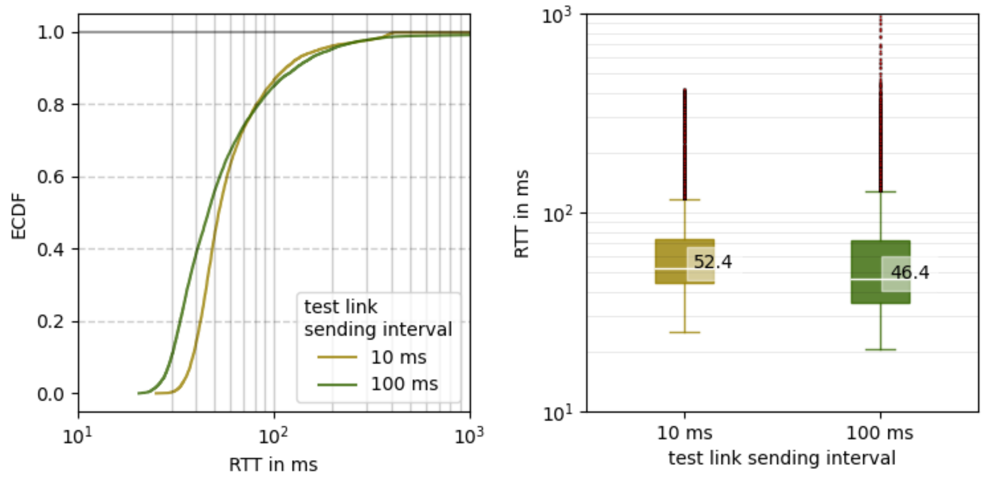
(h) Scatter plot of the RTT for a sending interval of 100 ms

Figure 6.24: Histogram and scatter plot of the RTT with a 5QI of 5 and a packet size of 128 bytes with changing sending interval and traffic of 20 Mbps with a 5QI of 80

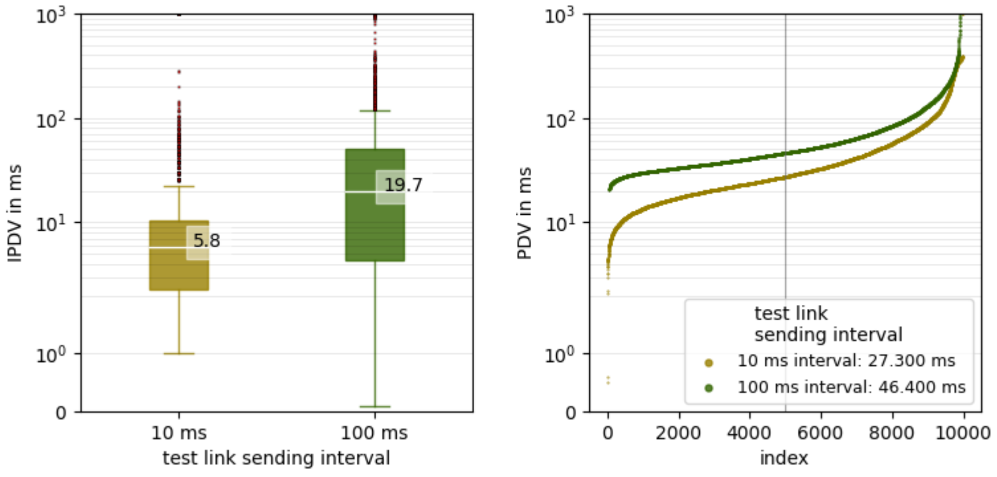
The following tests focus on the same setup but with a larger packet size of 1024 bytes and only one, 10, and 100 milliseconds sending intervals. The larger packets resulted in a connection loss right after the start of the measurements for the sending interval of one millisecond. The reason can be either a processing delay at the modems, which leads to the packets piling up and thus overfilling the modem buffer, or it can also be traced back to the core, which cannot process the packets properly and thus discards them. A test to determine which of these is the cause falls beyond the scope of this thesis and is planned within the following research steps. Fig. 6.25 shows the results for the two remaining measurements and the results of the RTT are comparable to the measurements with the smaller packets. The delay variation improved, which can be linked to the packages being all sent individually and not in bursts as with the smaller packages. In Fig. 6.25c is the very high PER for the fast interval visible and the PER for a 10 ms interval is not shown since it is zero.

In order to determine more precisely which settings are suitable for the use case, tests with a packet size of 178 bytes, which is the size of the packages sent by the sensor and the actuator, and on the basis of the available interval times of the robots were carried out. Here, the aim was to investigate the influence of the sawtooth profile from Fig. 6.24b on the trajectory and how this changes when the interval time is increased. Therefore, the fastest sending interval possible on the RTDE interface of the UR5 robots, which is five milliseconds, was the starting point, and additional tests with 10, 20, and 40 ms were done. The RTT, IPDV, PDV, and PER were measured for these intervals to determine whether these are viable options for testing on the robots. Fig. 6.26 shows the results and the RTT looks stable for all interval times. When looking at the delay variation in Fig. 6.26b, the fastest interval has a similar behavior as the delay variation in Fig. 6.24b. This leads to the observation, that a five millisecond interval causes the packets to be sent in batches.

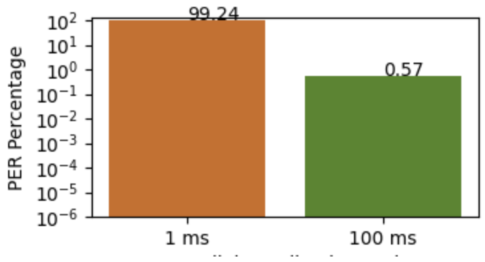
To investigate the data further, histograms and scatter plots of the RTT are used again. Fig. 6.27 shows the results and at first sight, there is no tendency to a sawtooth signal as in Fig. 6.24b visible, but the spread of the data for the fastest interval is noticeably higher.



(a) RTT for test link different sending intervals

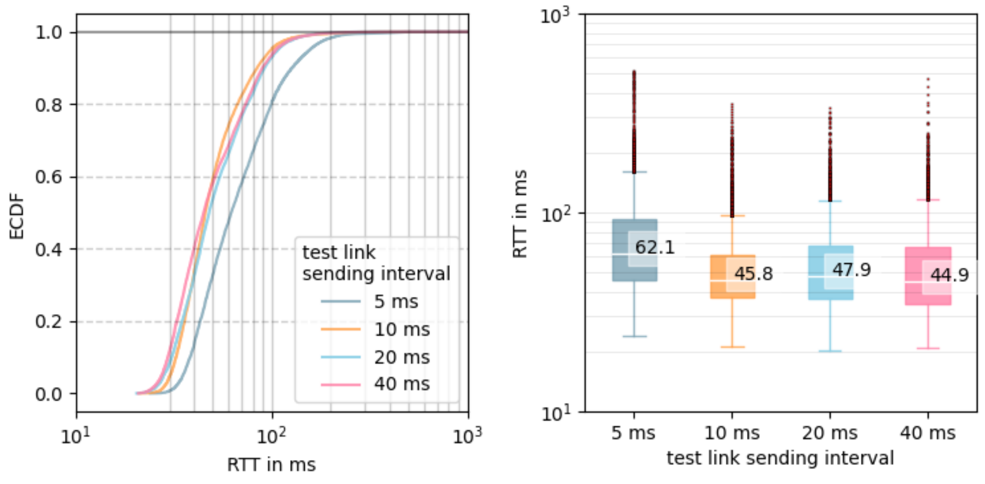


(b) IPDV and PDV for test link with different sending intervals

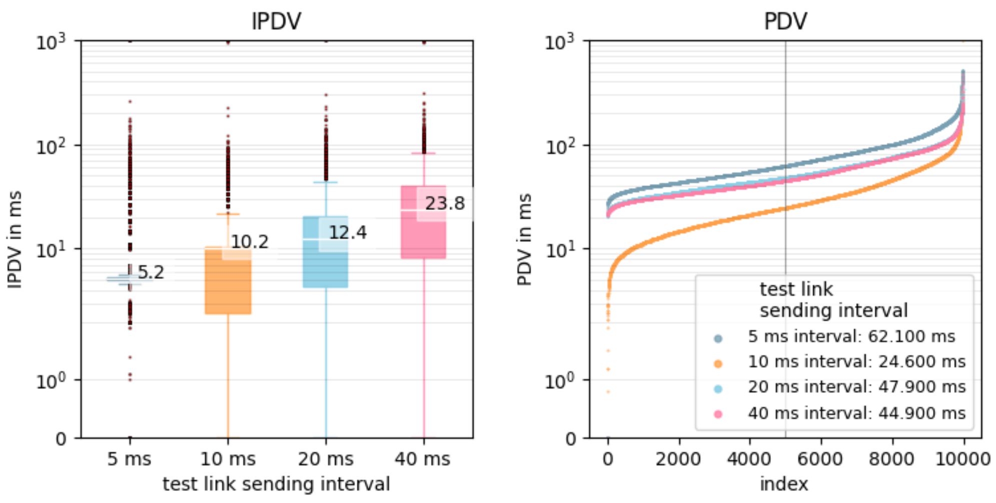


(c) PER for test link with different sending intervals

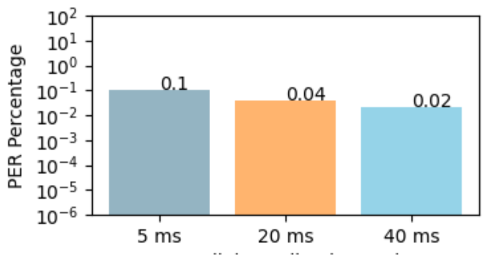
Figure 6.25: Results for testing different sending intervals with 5QI 5 and a packet size of 1024 bytes for the test link and a network load of 20 Mbps with 5QI 80



(a) RTT for test link different sending intervals

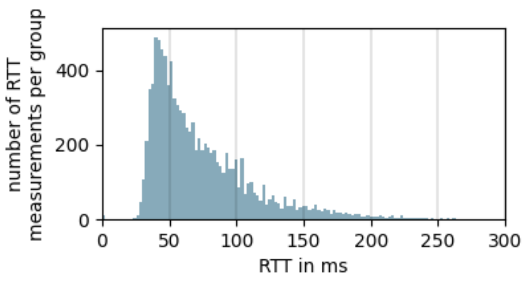


(b) IPDV and PDV for test link with different sending intervals

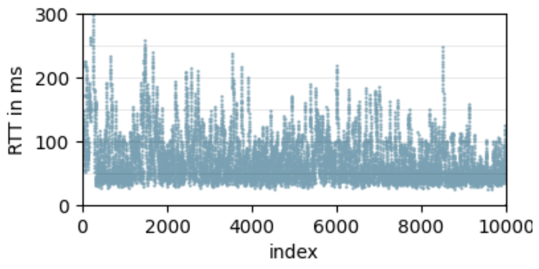


(c) PER for test link with different sending intervals

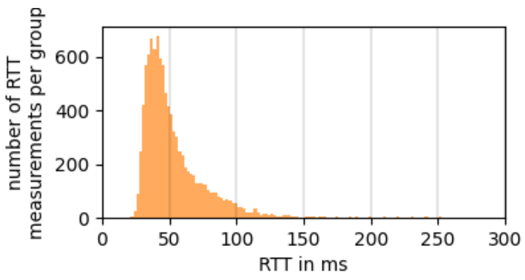
Figure 6.26: Results for testing different sending intervals with 5QI 5 and a packet size of 178 bytes for the test link and a network load of 20 Mbps with 5QI 80



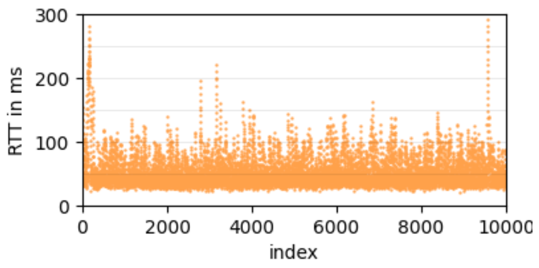
(a) Histogram of the RTT for a sending interval of 5 ms



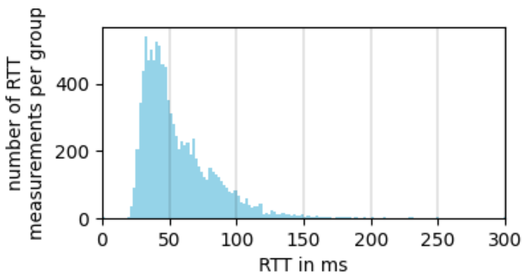
(b) Scatter plot of the RTT for a sending interval of 5 ms



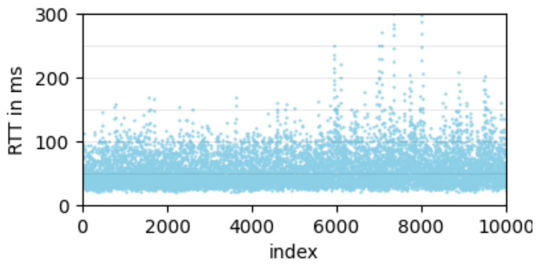
(c) Histogram of the RTT for a sending interval of 10 ms



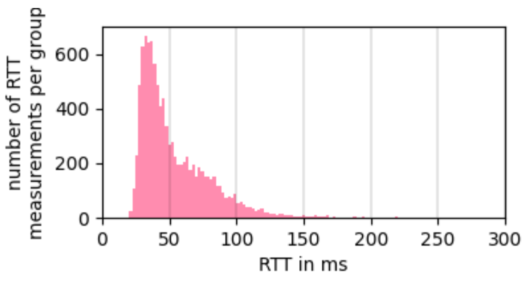
(d) Scatter plot of the RTT for a sending interval of 10 ms



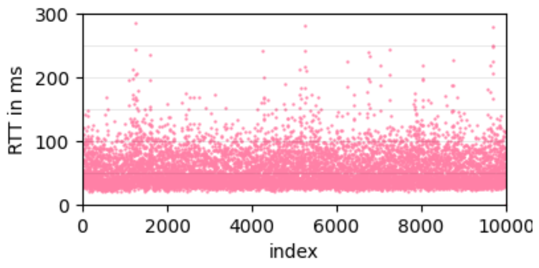
(e) Histogram of the RTT for a sending interval of 20 ms



(f) Scatter plot of the RTT for a sending interval of 20 ms



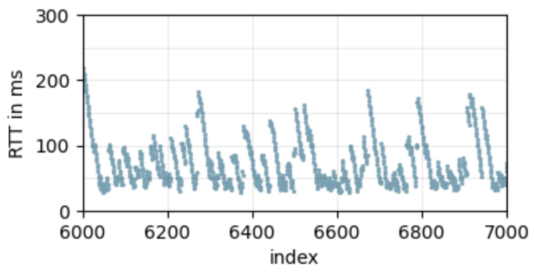
(g) Histogram of the RTT for a sending interval of 40 ms



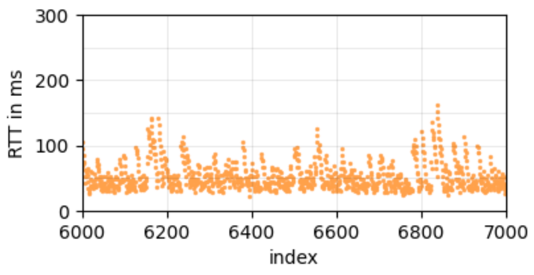
(h) Scatter plot of the RTT for a sending interval of 40 ms

Figure 6.27: Histogram and scatter plot of the RTT with a 5QI of 5 and a packet size of 128 bytes with changing sending interval and traffic of 20 Mbps with a 5QI of 80

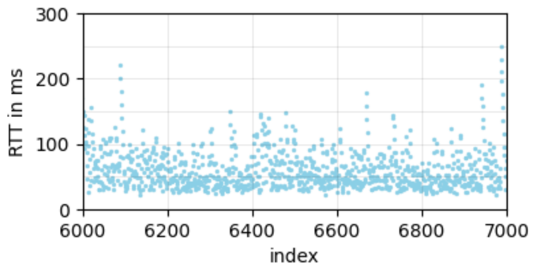
When zooming into the scatter plots on the values between measurement 6000



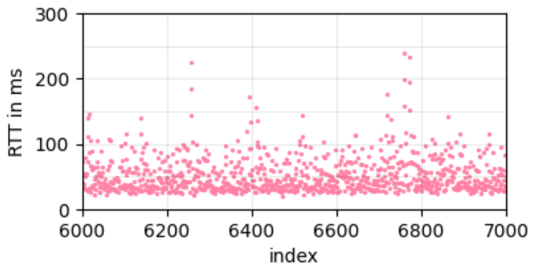
(a) Scatter plot of the RTT for a sending interval of 5 ms



(b) Scatter plot of the RTT for a sending interval of 10 ms



(c) Scatter plot of the RTT for a sending interval of 20 ms

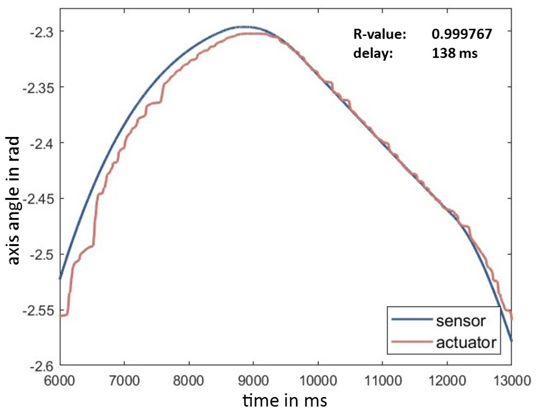


(d) Scatter plot of the RTT for a sending interval of 40 ms

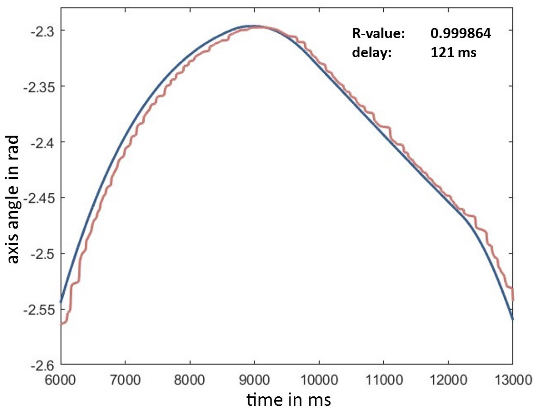
Figure 6.28: Zoom in on the scatter plot of the RTT with a 5QI of 5 and a packet size of 128 bytes with changing sending interval and traffic of 20 Mbps with a 5QI of 80

and 7000, shown in Fig. 6.28, it becomes visible that the same problems as before occur at a 5 ms transmission interval, only with lower intensity. These four intervals were additionally tested on the robots to see the direct influence on the trajectory of the actuator.

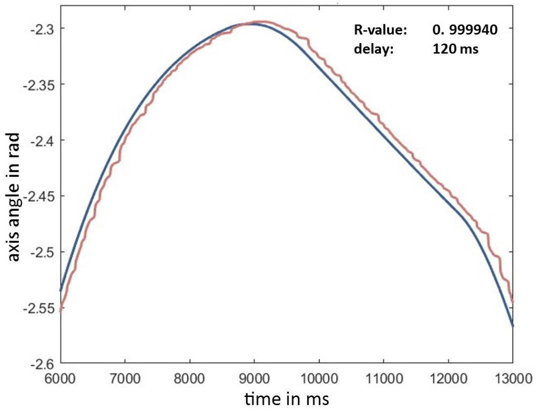
Fig. 6.29 shows a zoomed-in plot of the trajectories of the two robots for the different transmission intervals and confirms the negative influence of the five ms interval. This leads to the conclusion that the network should first be tested for the lowest viable transmission interval before machine communication is used in the process of automation.



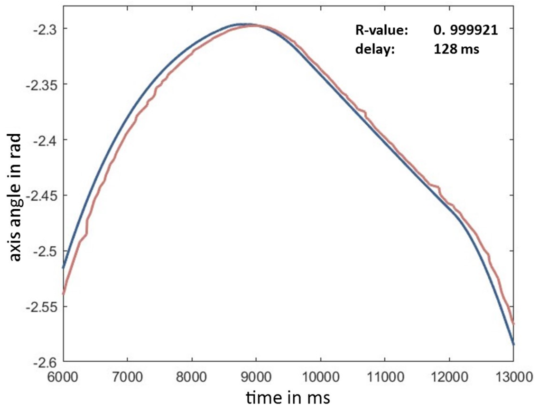
(a) Test link with a sending interval of 5 ms



(b) Test link with a sending interval of 10 ms



(c) Test link with a sending interval of 20 ms



(d) Test link with a sending interval of 40 ms

Figure 6.29: Trajectory comparison with a packet size of 178 bytes and changing transmission interval for the test link and 20 Mbps traffic

7 Conclusion and Outlook

5G stand-alone networks are one possible way to establish a wireless connection adapted to different use cases. This is particularly interesting for industrial purposes when it is a private, closed network in which the mobile subscriber has complete freedom of configuration. To what extent this is already applicable and where the current limits of such a system lie were one driver of this thesis. Therefore, a private 5G SA network in an industrial environment was investigated.

A brief introduction to 5G and the associated technologies was presented to give the reader a basic understanding of the material. The 5G core, the radio access network, and the related interfaces were described, and simple procedures explained.

To know the findings of this work as close as possible to industrial applications, a test environment that meets these requirements for testing, but is still applicable in practice, is essential. For this purpose, a use case was implemented that enables the control of two articulated arm robots. The measurements focused on the round-trip time, the inter-packet and packet delay variation, as well as the packet error rate as decisive variables. In addition, the trajectories of the two robots were recorded and compared with each other to visualize the effects of the network disturbances directly at the axis angles of the robots. The connection between these robots, called test link, was first tested with a fixed cabling to determine the baseline metrics of the implemented use case. Subsequently, the cabling was replaced by a wireless connection via 5G with different configuration setups.

For the measurements, four variables were examined in more detail as decisive factors for machine-to-machine communication:

- The 5G quality of service identifier, in short 5QI,
- The packet size of transmitted data,
- The transmission interval with which the data is sent,
- The influence of network traffic on a test link during the tests performed.

The 5QIs allow UEs to be assigned to a priority to allocate network resources efficiently. Here, three different 5QIs were examined for their behavior with limited network capacity. For this purpose, the 5G network was loaded with UDP traffic, and different test link to traffic combinations were tested. The test link behaves the same for every 5QI when tested against traffic of the same 5QI. The RTT and delay variation increase with a higher traffic bandwidth until the network reaches the maximum bandwidth. When the network reaches its capacity, the same priority for the test link and traffic causes unstable behavior, leading to a higher packet drop on the test link and the traffic. The advantages of prioritizing are visible when testing different 5QI for the test link and traffic with ICMP packets. When the test link has a higher priority, stable behavior is feasible even with a high network load. However, if the test link has a lower priority than the traffic, the measurements will show unusable behavior for automation purposes with PERs of 10% and higher. The TCP tests show the same results for the higher prioritized test link, but a much increased RTT and delay variation for the lower prioritized test link. The reason for this behavior is the high PER and the resulting resending of packets which delays the communication. An important insight was that manually overwriting the priority of a 5QI either had no effect in this system or the default priorities of the 5QIs were used when the test link and traffic had the same manually configured priority. The trajectories of the sensor and actuator robot show similar behavior to the measurements performed. The delay of the trajectories and the R-value are stable for higher-priority connections and show a sharp drop for lower-priority connections. These results are comparable to the results of [5], where the tests with network traffic showed a strong deviation from sensor to actuator trajectory. Therefore, it is crucial to identify essential connections already during network planning and prioritize them using the assigned 5QI. Nevertheless, it must be noted that the measured values are far from the target values specified in [7] for an SA 5G network, and especially the high PERs make the use of this system for automation purposes questionable in its current state and further improvement is necessary.

Since the size of the packets to be transmitted often depends on the application, tests are carried out for different packet sizes to examine the influence more closely. Especially the combination of test link and traffic packet size was the focus of these tests to determine which relation is best suited for the implemented use case. The results show that communication without traffic behaves similarly for all packet sizes. The measured RTT can be compared to the measurements in [4], where upload and download time was measured separately, and the RTT data distribution caused by the TDD during these tests was also explained. If added traffic to the network, it is visible that the 5G system can handle small packets in principle better than medium-sized and large packets. Considering the test link and the traffic separately and in more detail, small packets with a size of 256 bytes or smaller show the best results, whereas the traffic packet size shows the least influence on the test link with the largest tested packet size of 1500 bytes. The robot trajectory shows a similar behavior compared to the results from the measurements. Especially traffic with a packet size of 1500 bytes shows viable results, whereas traffic with a packet size of 512 bytes caused massive problems. Further was, a measurement of the trajectory not possible with a test link packet size bigger than 256 bytes, highlighting the importance of small packet sizes for automation purposes. The conclusion drawn from these measurements is that communication that is bandwidth oriented, like video streaming, should be executed with large packet sizes, whereas applications more oriented towards reliability and lower latency should use smaller packets. This improves the individual connections' performance and reduces interference between them.

The transmission interval of the packets is often configurable for industrial IoT devices. For example, the sensor and actuator robots use a configurable update interval for their RTDE interface. This led to the measurements with changing transmission intervals to examine the influence on the test link. The results show that if data is sent in an interval that the 5G system can not handle, the transmission occurs in batches, leading to unusable behavior for i.e. robot control. The results from [3] show similar behavior and associate this with the use of TDD, where a channel gets reserved for a specific time, and if the reservation expires, the packet queue stacks up, which leads to s saw-tooth-like signal for the RTT. The transmission interval was observed more precise with the packet size used for the trajectory tracing and in the practicable sending interval from 5 to 40 milliseconds. This showed that a transmission interval of five milliseconds is already borderline, and thus increased interference occurs. The actuator robot's trajectory shows no significant delay increase when changing the transmission interval from 5 to 40 milliseconds since the RTT is higher than 40 milliseconds. This leads to the conclusion that when implementing a 5G network for industrial purposes, the interval requirements of the sensors must first be considered, and based on this it can be decided whether an implementation is even feasible.

Measuring the influence of network disturbances directly on the robot trajectory could be implemented. First, simulations were performed to compare the expected influence of the simulations with the measurements. Subsequently, the direct influence of RTT, delay variation, and error rate on the delay and the correlation coefficient between the sensor and actuator robot trajectory could be observed.

In summary, machine-to-machine communication can already be implemented sensibly using SA 5G. Compared to what has been specified by the 3GPP standard, however, some compromises must be made, especially with regard to the packet error rate. This has significantly higher values than described and can therefore lead to problems, especially in critical implementations. The measured end-to-end delays are in the range that a one-way delay in the sub-10 millisecond range can be achieved, but to reach the one millisecond limit described for URLLC applications, the system must be further optimized. The problems caused by using TDD lead to the conclusion that further tests with a different frequency band that supports FDD are necessary. Furthermore, the testing of one-way delay is important to detect the asymmetric behavior described in [3] and subsequently to adjust the system to a symmetric behavior. It was found that the hardware in UEs is often not yet certified for the frequency band used, making connection impossible. The 5G modems used are also very error-prone, which often leads to connection losses and makes it necessary to restart the modem. This is critical in an industrial environment and makes it difficult to implement quality robot control.

7.1 Outlook

Since these were the first tests with this system, the possibilities for future research directions are vast. Whether it is the improvement of the system at hand or exploring different technologies, some possible directions for future research are:

- The implementation of network slicing enables a comprehensive decoupling of different applications from the Radio Access Network and 5G Core Network. This reduces interference from User Equipment that require different use cases. Research in this direction is particularly valuable for the automation of large-scale systems and should therefore be a priority.

- Secure and real-time remote control of robots via wireless systems is particularly interesting in the field of medicine and in hard-to-reach locations. The implemented use case can serve as a basis for further research in the area of trajectory recording, processing, and implementation of movements. A first step would be to replace the sensor robot with an input unit that allows for arbitrary control of the robot. Here, the implementation of a toolchain from an input trajectory to the transmitted axis angles should be prioritized.

- The rollout of new 3GPP releases includes the utilization of frequency range two in the higher range above 24 GHz. This enables the transmission of very large data volumes with the trade-off of short ranges. The presented use case could greatly benefit from this, and it should be explored as soon as possible to investigate frequency range two as well.

- Further improvement of the system at hand will be key. For machine communication, symmetric transmission is especially interesting. To observe this, the first step would be the implementation of one-way delay measurements to consider upload and download separately. The implementation can be done using a similar model as [3] and allows, in particular, to look at the behavior of the RAN in detail. The adjustment from a TDD to an FDD channel is dependent on the frequency band and therefore needs a research license. Obtaining one for future tests is essential since it would also solve the bandwidth restriction. Until this happens, the RAN should be configured in such a way that upload and download receive the same resources, and thus the influence of asymmetric communication is reduced. Furthermore, it will be important to optimize and automate the testing methods for this setup. The implementation of a traffic generator is essential for further testing. It should be configurable to generate both TCP and UDP packets of different sizes and with variable sending intervals. Additionally, it should be capable of monitoring the generated traffic.

However, further research possibilities strongly depend on the available devices and up-to-date software. It has already been observed during this work that the theoretical standards and marketing of 5G network providers are ahead of the actual implementation capabilities, indicating a need for catching up in this regard.

Bibliography

[1] D. Chandramouli, R. Liebhart, and J. Pirskanen, Eds., *5G for the connected world*, First edition. Hoboken, NJ: John Wiley & Sons, Inc, 2019, ISBN: 978-1-119-24713-5 978-1-119-24707-4.

[2] M. Sauter, *Grundkurs Mobile Kommunikationssysteme: 5G New Radio und Kernnetz, LTE-Advanced Pro, GSM, Wireless LAN und Bluetooth*, en. Wiesbaden: Springer Fachmedien Wiesbaden, 2022, ISBN: 978-3-658-36962-0 978-3-658-36963-7. DOI: `10.1007/978-3-658-36963-7`. [Online]. Available: `https://link.springer.com/10.1007/978-3-658-36963-7` (visited on 03/14/2023).

[3] J. Rischke, P. Sossalla, S. Itting, F. H. P. Fitzek, and M. Reisslein, “5G Campus Networks: A First Measurement Study,” en, *IEEE Access*, vol. 9, pp. 121 786–121 803, 2021, ISSN: 2169-3536. DOI: `10.1109/ACCESS.2021.3108423`. [Online]. Available: `https://ieeexplore.ieee.org/document/9524600/` (visited on 03/06/2023).

[4] G. Cainelli, M. Donga, L. Rauchhaupt, and L. Underberg, “Performance evaluation of a 5G device in a non public network,” en, in *2021 IEEE 4th 5G World Forum (5GWF)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 334–339, ISBN: 978-1-66544-308-1. DOI: `10.1109/5GWF52925.2021.00065`. [Online]. Available: `https://ieeexplore.ieee.org/document/9605002/` (visited on 03/06/2023).

[5] Sven Wittig, Mathis Schmieder, Henrik Klessig, *et al.*, “Private 5G Networks for Connected Industries, Final Project Report, 5G CONNI,” Tech. Rep. 861459. [Online]. Available: `https://5g-conni.eu/` (visited on 03/12/2023).

[6] 3GPP, *TR 21.915: "Release description"; V 15.00.0*, 2019. [Online]. Available: `https://www.etsi.org/deliver/etsi_tr/121900_121999/121915/15.00.00_60/tr_121915v150000p.pdf` (visited on 03/27/2023).

[7] 3GPP, *TS 23.501: "System architecture for the 5G System"; V 15.13.0*, 2022. [Online]. Available: `https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.13.00_60/ts_123501v151300p.pdf` (visited on 03/23/2023).

[8] 3GPP, *TS 29.500: "Technical Realization of Service Based Architecture"; V 15.07.0*, 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129500/15.07.00_60/ts_129500v150700p.pdf (visited on 02/23/2023).

[9] 3GPP, *TS 38.401: "NG-RAN: Architecture description"; V 15.09.0*, 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138400_138499/138401/15.09.00_60/ts_138401v150900p.pdf (visited on 04/09/2023).

[10] 3GPP, *TS 38.201: "Physical layer; General description" V 15.00.0*, 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138200_138299/138201/15.00.00_60/ts_138201v150000p.pdf (visited on 04/09/2023).

[11] 3GPP, *TS 38.300: "NR and NG-RAN Overall description" V 15.13.0*, 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138300_138399/138300/15.13.00_60/ts_138300v151300p.pdf (visited on 04/09/2023).

[12] P. A. Höher, *Grundlagen der digitalen Informationsübertragung: von der Theorie zu Mobilfunkanwendungen ; mit 31 Tabellen und 234 Beispielen* (Lehrbuch), ger, 2., überarb. u. erw. Aufl. Wiesbaden: Springer Vieweg, 2013, ISBN: 978-3-8348-1784-6.

[13] R. v. Nee and R. Prasad, *OFDM for wireless multimedia communications* (Artech House universal personal communications series). Boston: Artech House, 2000, ISBN: 978-0-89006-530-3.

[14] R. E. Ziemer and W. H. Tranter, *Principles of communication: systems, modulation, and noise*, eng, 7. ed. Hoboken, NJ: Wiley, 2015, ISBN: 978-1-118-80457-5 978-1-118-07891-4.

[15] L. Rusch, “"Equalization", Lecture Notes;” online, Université Laval. [Online]. Available: http://wcours.gel.ulaval.ca/2020/a/GEL7014/default/5notes/chp8_equalization_light_en.pdf (visited on 04/12/2023).

[16] A. Goldsmith, *Wireless communications*, eng. Cambridge: Cambridge University Press, 2005, OCLC: 320129180, ISBN: 978-1-60119-764-1.

[17] M. Werner, *Nachrichtentechnik: eine Einführung für alle Studiengänge* (Lehrbuch), ger, 8., vollständig überarbeitete und erweiterte Auflage. Wiesbaden [Heidelberg]: Springer Vieweg, 2017, ISBN: 978-3-8348-2580-3. DOI: 10.1007/978-3-8348-2581-0.

[18] T. M. Inc., *Phased array system toolbox version: 5.0 (r2023a)*, 2022.

[19] IETF, *IETF RFC 4960*, 2007. [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc4960.txt.pdf (visited on 04/10/2023).

[20] 3GPP, *TS 38.410: "NG-RAN; NG general aspects and principles"; V 15.02.0*, 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138400_138499/138410/15.02.00_60/ts_138410v150200p.pdf (visited on 04/10/2023).

[21] 3GPP, *TS 38.470: "NG-RAN; F1 general aspects and principles" V 15.08.0*, 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138400_138499/138470/15.08.00_60/ts_138470v150800p.pdf (visited on 04/10/2023).

[22] 3GPP, *TS 33.501: "Security architecture and procedures for 5G System" V 15.16.0*, 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.16.00_60/ts_133501v151600p.pdf (visited on 04/11/2023).

[23] 3GPP, *TS 23503: "Policy and charging control framework for the 5G System" V 15.10.0*, 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123503/15.10.00_60/ts_123503v151000p.pdf (visited on 04/11/2023).

[24] 3GPP, *TS 123502: "Procedures for the 5G System"; V 15.16.0*, 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/15.16.00_60/ts_123502v151600p.pdf (visited on 03/23/2023).

[25] Athonet, *5G standalone core network*, 2023. [Online]. Available: https://athonet.com/products/athonet-5g-core/ (visited on 03/23/2023).

[26] Mikrotik, *CRS305-1G-4S+IN switch*, 2023. [Online]. Available: https://mikrotik.com/product/crs305_1g_4s_in (visited on 03/23/2023).

[27] BTI Wireless, *5G NR Femtocell nCELL F2240*, 2023. [Online]. Available: https://www.btiwireless.com/5G-NR-Femtocell-nCELL-F2240-pd44258730.html (visited on 03/23/2023).

[28] Universal Robots, *Dashboard Server Remote Control Interface, e Series*, 2022. [Online]. Available: https://s3-eu-west-1.amazonaws.com/ur-support-site/42728/DashboardServer_e-Series_2022.pdf (visited on 04/23/2023).

[29] Universal Robots, *Real-Time Data Exchange (RTDE) Guide*, en, 2022. [Online]. Available: https://s3-eu-west-1.amazonaws.com/ur-support-site/22229/Real_Time_Data_Exchange_(RTDE)_Guide.pdf (visited on 03/14/2023).

[30] Robotnik, *RB-Kairos+ mobile manipulator*, 2023. [Online]. Available: https://robotnik.eu/products/mobile-manipulators/rb-kairos (visited on 03/23/2023).

[31] Teltonika, *RUTX50 industrial 5G router*, 2023. [Online]. Available: `https://teltonika-networks.com/de/products/routers/rutx50` (visited on 03/23/2023).

[32] Universal Robots, *Real-Time Data Exchange (RTDE) Guide*. [Online]. Available: `https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/` (visited on 05/01/2023).

[33] Universal Robots, *RTDE_python_client_library [Source Code]*. [Online]. Available: `https://github.com/UniversalRobots/RTDE_Python_Client_Library` (visited on 05/01/2023).

[34] A. Morton and B. Claise, *Packet Delay Variation Applicability Statement, document RFC 5481*, 2009. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc5481` (visited on 06/03/2023).

[35] D. Comer, *Computer networks and internets* (Always learning), en, Sixth edition, Global edition. Boston Columbus Indianapolis New York: Pearson, 2015, ISBN: 978-1-292-06117-7 978-0-13-358793-7.

[36] W. Li, D. Zhang, G. Xie, and J. Yang, "TCP and ICMP in Network Measurement: An Experimental Evaluation," in *Parallel and Distributed Processing and Applications*, D. Hutchison, T. Kanade, J. Kittler, *et al.*, Eds., vol. 3758, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 870–881, ISBN: 978-3-540-29769-7 978-3-540-32100-2. DOI: `10.1007/11576235_87`. [Online]. Available: `http://link.springer.com/10.1007/11576235_87` (visited on 05/07/2023).

[37] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," en, *Computer Communication Review*,

[38] D. Freedman, R. Pisani, and R. Purves, *Statistics*, 4th ed. New York: W.W. Norton & Co, 2007, OCLC: ocm76142955, ISBN: 978-0-393-92972-0 978-0-393-93043-6.

[39] U. Kuckartz, S. Rädiker, T. Ebert, and J. Schehl, *Statistik: eine verständliche Einführung* (Lehrbuch), ger, 2., überarbeitete Auflage. Wiesbaden: Springer VS, 2013, ISBN: 978-3-531-19889-7.

[40] B. E. Abali and C. Çakiroğlu, *Numerische Methoden für Ingenieure: mit Anwendungsbeispielen in Python* (Lehrbuch), ger. Berlin [Heidelberg]: Springer Vieweg, 2020, ISBN: 978-3-662-61324-5.

Appendix

Listing 7.1: Code for robot control and trajectory measurement

```
# name:      rtde_trajectory_measurement
# author:    m.sonnberger
# date:      15.04.2023
# use:       collects trajectory data from sensor and sends it to actuator

#           opens thread to measure sensor and actuator trajectory (angles, speed, time) and
#           sends the data from the sensor to the actautor to remote control the movements

#           necessary librarys: rtdelibs from universal robots and configuration files
#           (.xml) for RTDE

import sys
import time
import logging
import threading
import rtde_read_write_functions as rtde_wr
import numpy as np
import pandas as pd
import socket

import os
import psutil

import logging

import rtdelibs.rtde as rtde
import rtdelibs.rtde_config as rtde_config

os_used = sys.platform
process = psutil.Process(os.getpid())

if os_used == "linux":
    pid = os.getpid()
    os.system("sudo renice -n -19 -p " + str(pid))
```

```python
elif os_used == "win32":
    process.nice(psutil.REALTIME_PRIORITY_CLASS)
else:
    print("OS not supported)")

sys.path.append("..")

MAX_ITERATIONS = 24000

RTDE_SENSOR_IP = '192.168.1.200'
RTDE_SENSOR_PORT = 30004

RTDE_ACTUATOR_IP = '10.10.0.200'
RTDE_ACTUATOR_PORT = 30004

DASHBOARD_SENSOR_IP = "192.168.1.200"
DASHBOARD_SENSOR_PORT = 29999

DASHBOARD_ACTUATOR_IP = "10.10.0.200"
DASHBOARD_ACTUATOR_PORT = 29999

actualJointSpeed = [0.0,0.0,0.0,0.0,0.0,0.0]

running = 1

def list_to_setp(sp, list):
    for i in range(0, 6):
        sp.__dict__["input_double_register_%i" % i] = list[i]
    return sp

def setp_to_list(sp):
    sp_list = []
    for i in range(0, 6):
        sp_list.append(sp.__dict__["input_double_register_%i" % i])
    return sp_list

def test_print_thread():
    global actualJointSpeed
    while True:
        with lock_jointAngles:
            print("actualJointAngles = ", str(actualJointSpeed))
        time.sleep(0.008)

```

```python
def sendDashboardCommand(dashboard, cmd):
    cmd = cmd + '\n'
    print(cmd)
    dashboard.sendall(cmd.encode())
    time.sleep(2)
    rcvd = dashboard.recv(4096)
    print(rcvd)
    return rcvd
    # return True

# sensor thread
def rtde_sensor(sensorIP, sensorPort, configFilename):

    global actualJointSpeed
    global startTime
    global running
    logging.getLogger().setLevel(logging.INFO)

    conf = rtde_config.ConfigFile(configFilename)

    state_names, state_types = conf.get_recipe("state")
    watchdog_names, watchdog_types = conf.get_recipe("watchdog")
    speed_slider_names_s, speed_slider_types_s = conf.get_recipe("speed_slider")

    con = rtde.RTDE(sensorIP, sensorPort)
    con.connect()

    print("Connected to Sensor")

    # get controller version
    con.get_controller_version()

    # setup recipes
    states = con.send_output_setup(state_names, state_types)
    speed_slider_sensor = con.send_input_setup(speed_slider_names_s, speed_slider_types_s)

    speed_slider_sensor.speed_slider_mask = 1
    speed_slider_sensor.speed_slider_fraction = 0.3

    watchdog = con.send_input_setup(watchdog_names, watchdog_types)

    # The function "rtde_set_watchdog" in the "rtde_control_loop.urp"
    # creates a 1 Hz watchdog
    watchdog.input_int_register_0 = 0

    # Datastorage for trajectory
    trajectory = np.zeros((MAX_ITERATIONS,14), dtype=float)

    # start data synchronization
    if not con.send_start():
        sys.exit()

    con.send(speed_slider_sensor)

    sensorTime = 0
    i = 0

    while(running):

        state = con.receive()

        sensorTime = ((time.perf_counter_ns()-startTime)/1000000)

        if state is None:
            break

        with lock_jointAngles:
            actualJointSpeed = [round(state.actual_qd[0],5),
                                round(state.actual_qd[1],5),
                                round(state.actual_qd[2],5),
                                round(state.actual_qd[3],5),
                                round(state.actual_qd[4],5),
                                round(state.actual_qd[5],5)]

        trajectory[i] = [   sensorTime,
                            round(state.timestamp, 5),
                            round(state.actual_q[0],5),
                            round(state.actual_q[1],5),
                            round(state.actual_q[2],5),
                            round(state.actual_q[3],5),
                            round(state.actual_q[4],5),
                            round(state.actual_q[5],5),
                            round(state.actual_qd[0],5),
                            round(state.actual_qd[1],5),
                            round(state.actual_qd[2],5),
                            round(state.actual_qd[3],5),
                              round(state.actual_qd[4],5),
                              round(state.actual_qd[5],5)]

        # kick watchdog
        watchdog.input_int_register_0 = 1
        con.send(watchdog)
        i = i+1

    # change nice value back to normal
    os.system("sudo renice -n 0 -p " + str(pid))

    # write trajectory + time in .xlsx file
    sensorData = pd.DataFrame({     'Global Time':trajectory[:,0],
                                    'Sensor Time':trajectory[:,1],
                                    'Axis Angle 1': trajectory[:,2],
                                    'Axis Angle 2': trajectory[:,3],
                                    'Axis Angle 3': trajectory[:,4],
                                    'Axis Angle 4': trajectory[:,5],
                                    'Axis Angle 5': trajectory[:,6],
                                    'Axis Angle 6': trajectory[:,7],
                                    'Axis Speed 1': trajectory[:,8],
                                    'Axis Speed 2': trajectory[:,9],
                                    'Axis Speed 3': trajectory[:,10],
                                    'Axis Speed 4': trajectory[:,11],
                                    'Axis Speed 5': trajectory[:,12],
                                    'Axis Speed 6': trajectory[:,13],
                                })

    # delete rows with zeros
    sensorData = sensorData.loc[sensorData["Global Time"] != 0]

    sensorData.to_excel('data/sensor_data.xlsx', sheet_name='jointAngles_time')

    con.send_pause()
    con.disconnect()

# actuator thread
def rtde_actuator(actuatorIP, actuatorPort, configFilename):

    global actualJointSpeed
    global startTime

    logging.getLogger().setLevel(logging.INFO)
```

```python

    conf = rtde_config.ConfigFile(configFilename)

    state_names, state_types = conf.get_recipe("state")
    setp_names, setp_types = conf.get_recipe("setp")
    speed_slider_names_a, speed_slider_types_a = conf.get_recipe("speed_slider")
    watchdog_names, watchdog_types = conf.get_recipe("watchdog")

    con = rtde.RTDE(actuatorIP, actuatorPort)
    con.connect()

    print("Connected to Actuator")

    # get controller version
    con.get_controller_version()

    # setup recipes
    states = con.send_output_setup(state_names, state_types)
    setp = con.send_input_setup(setp_names, setp_types)
    speed_slider_actuator = con.send_input_setup(speed_slider_names_a, speed_slider_types_a)

    watchdog = con.send_input_setup(watchdog_names, watchdog_types)

    speed_slider_actuator.speed_slider_mask = 1
    speed_slider_actuator.speed_slider_fraction = 1

    setp.input_double_register_0 = 0
    setp.input_double_register_1 = 0
    setp.input_double_register_2 = 0
    setp.input_double_register_3 = 0
    setp.input_double_register_4 = 0
    setp.input_double_register_5 = 0

    # The function "rtde_set_watchdog" in the "rtde_control_loop.urp"
    # creates a 1 Hz watchdog
    watchdog.input_int_register_0 = 0

    # Datastorage for trajectory
    trajectory = np.zeros((MAX_ITERATIONS,14), dtype=float)

    # start data synchronization
    if not con.send_start():
        sys.exit()
```

```

con.send(speed_slider_actuator)

while actualJointSpeed == [0.0,0.0,0.0,0.0,0.0,0.0]:
    time.sleep(1/100000)


i = 0

while(running):

    state = con.receive()
    actuatorTime = ((time.perf_counter_ns()-startTime)/1000000)

    if state is None:
        break

    with lock_jointAngles:
        new_setp = actualJointSpeed
    list_to_setp(setp, new_setp)


    # send new setpoint
    con.send(setp)

    trajectory[i] = [   actuatorTime,
                        round(state.timestamp, 5),
                        round(state.actual_q[0],5),
                        round(state.actual_q[1],5),
                        round(state.actual_q[2],5),
                        round(state.actual_q[3],5),
                        round(state.actual_q[4],5),
                        round(state.actual_q[5],5),
                        round(state.actual_qd[0],5),
                        round(state.actual_qd[1],5),
                        round(state.actual_qd[2],5),
                        round(state.actual_qd[3],5),
                        round(state.actual_qd[4],5),
                        round(state.actual_qd[5],5)]

    # kick watchdog
    watchdog.input_int_register_0 = 1
    con.send(watchdog)
```

```python
        i = i+1

    # change nice value back to normal
    os.system("sudo renice -n 0 -p " + str(pid))

    # write trajectory + time in .xlsx file
    actuatorData = pd.DataFrame({   'Global Time':trajectory[:,0],
                                    'Actuator Time':trajectory[:,1],
                                    'Axis Angle 1': trajectory[:,2],
                                    'Axis Angle 2': trajectory[:,3],
                                    'Axis Angle 3': trajectory[:,4],
                                    'Axis Angle 4': trajectory[:,5],
                                    'Axis Angle 5': trajectory[:,6],
                                    'Axis Angle 6': trajectory[:,7],
                                    'Axis Speed 1': trajectory[:,8],
                                    'Axis Speed 2': trajectory[:,9],
                                    'Axis Speed 3': trajectory[:,10],
                                    'Axis Speed 4': trajectory[:,11],
                                    'Axis Speed 5': trajectory[:,12],
                                    'Axis Speed 6': trajectory[:,13],
                               })

    # delete rows with zeros
    actuatorData = actuatorData.loc[actuatorData["Global Time"] != 0]
    actuatorData.to_excel('data/actuator_data.xlsx', sheet_name='jointAngles_time')

    con.send_pause()
    con.disconnect()



lock_jointAngles = threading.Lock()

dashboardSensor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
dashboardSensor.settimeout(2)
dashboardSensor.connect((DASHBOARD_SENSOR_IP, DASHBOARD_SENSOR_PORT))

sendDashboardCommand(dashboardSensor,'programState')
sendDashboardCommand(dashboardSensor,'load rtde/Trajectory_measurement/rtde_sensor.urp')

dashboardActuator = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
dashboardActuator.settimeout(2)
dashboardActuator.connect((DASHBOARD_ACTUATOR_IP, DASHBOARD_ACTUATOR_PORT))
```

```python

sendDashboardCommand(dashboardActuator,'programState')
sendDashboardCommand(dashboardActuator,'load rtde/Trajectory_measurement/rtde_actuator.urp')

time.sleep(2)
# create sensor thread
thread_sensor = threading.Thread(target=rtde_sensor,
                                 args=( RTDE_SENSOR_IP,
                                        RTDE_SENSOR_PORT,
                                        "configuration_rtde_sensor.xml"))

# create actuator thread
thread_actuator = threading.Thread(target=rtde_actuator,
                                   args=(RTDE_ACTUATOR_IP,
                                         RTDE_ACTUATOR_PORT,
                                        "configuration_rtde_actuator.xml"))

startTime = time.perf_counter_ns()

# start sensor thread
print("Thread sensor start")
thread_sensor.start()

# start actuator thread
print("Thread actuator start")
thread_actuator.start()

# start Actuator
print("Start Actuator")
sendDashboardCommand(dashboardActuator,'play')
time.sleep(1)

# start Sensor
print("Start Sensor")
sendDashboardCommand(dashboardSensor,'play')

while(running):
    received = sendDashboardCommand(dashboardSensor,'programState')

    if received.find(b'STOPPED') != -1:
        print("program stopped")

        # stop Actuator
        sendDashboardCommand(dashboardActuator,'stop')
        running = 0


actualJointSpeed = [0.0,0.0,0.0,0.0,0.0,0.0]
```

Listing 7.2: Configuration file for the actuator RTDE client

```
<!--
name:       configuration_rtde_actuator
author:     m.sonnberger
date:       15.04.2023
use:        configuration file for RTDE actuator
-->

<?xml version="1.0"?>
<rtde_config>
    <recipe key="state">
        <field name="timestamp" type="DOUBLE"/>
        <field name="actual_q" type="VECTOR6D"/>
        <field name="actual_qd" type="VECTOR6D"/>
        <field name="output_int_register_1" type="INT32"/>
    </recipe>

    <recipe key="setp">
        <field name="input_double_register_0" type="DOUBLE"/>
        <field name="input_double_register_1" type="DOUBLE"/>
        <field name="input_double_register_2" type="DOUBLE"/>
        <field name="input_double_register_3" type="DOUBLE"/>
        <field name="input_double_register_4" type="DOUBLE"/>
        <field name="input_double_register_5" type="DOUBLE"/>
    </recipe>

    <recipe key="speed_slider">
        <field name="speed_slider_mask" type="UINT32"/>
        <field name="speed_slider_fraction" type="DOUBLE"/>
    </recipe>

    <recipe key="watchdog">
        <field name="input_int_register_0" type="INT32"/>
    </recipe>
</rtde_config>
```

Listing 7.3: Configuration file for the sensor RTDE client

```xml
<!--
name:       configuration_rtde_sensor
author:     m.sonnberger
date:       15.04.2023
use:        configuration file for RTDE sensor
-->

<?xml version="1.0"?>
<rtde_config>
    <recipe key="state">
        <field name="timestamp" type="DOUBLE"/>
        <field name="actual_q" type="VECTOR6D"/>
        <field name="actual_qd" type="VECTOR6D"/>
    </recipe>

    <recipe key="speed_slider">
        <field name="speed_slider_mask" type="UINT32"/>
        <field name="speed_slider_fraction" type="DOUBLE"/>
    </recipe>

    <recipe key="watchdog">
        <field name="input_int_register_0" type="INT32"/>
    </recipe>

</rtde_config>
```

Listing 7.4: Code for ICMP testing with ping

```python
# name:         ping_tester
# author:       m.sonnberger
# date:         27.03.2023
# use:          ICMP tester for measurements of RTT, jitter and PER

#              sends CYCLES * COUNT ICMP packets with a size of SIZE to a
#              network device HOSTNAME and measures RTT, jitter and PER

import os
import sys
import numpy as np
import pandas as pd
import time
import sys
import os
import psutil

# target
HOSTNAME = "10.10.0.200"

# measurement configurations
CYCLES = 500
COUNT = 10
INTERVAL = 0.01
SIZE = 256

# set scheduler priority of code to REALTIME (highest)
os_used = sys.platform
# Set highest priority for the python script on the CPU
process = psutil.Process(os.getpid())
if os_used == "win32":
    process.nice(psutil.REALTIME_PRIORITY_CLASS)
elif os_used == "linux":
    pid = os.getpid()
    os.system("sudo renice -n -19 -p " + str(pid))
else:
    process.nice(20)


# command shell
command = f"sudo ping -c {COUNT} -i {INTERVAL} -s {SIZE} {HOSTNAME}"

# analysis variables
searchword = "min/avg/max/mdev"
minimum = [0] * CYCLES
average = [0] * CYCLES
maximum = [0] * CYCLES
mdev = [0] * CYCLES
per = [0] * CYCLES

# program start
start_time = time.perf_counter()
for j in range(CYCLES):


    response = os.popen(command).read()

    strings = response.split()
    packetErrors = 0

    if searchword in strings:
        position = strings.index(searchword)
        stats = strings[position + 2]
        per[j] = strings[position - 6]

    index = 0
    k = 0
    delimiter = [0] * 3
    for i in range(len(stats)):
        position = stats.find("/",index)
        if(position!= -1):
            index = position+1
            delimiter[k] = position
            k = k + 1

    minimum[j] = stats[0:delimiter[0]]
    average[j] = stats[delimiter[0]+1:delimiter[1]]
    maximum[j] = stats[delimiter[1]+1:delimiter[2]]
    mdev[j] = stats[delimiter[2]+1:len(stats)]

stop_time = time.perf_counter()

# data processing
rttData = pd.DataFrame({'minimum': minimum,
                        'average': average,
                        'maximum': maximum,
                        'mdev': mdev,
                        'per': per
                        })

print("\n\rprocess time: " + str(stop_time - start_time))
rttData.to_excel('/home/factory/rtde/ws/data/ping_response.xlsx', sheet_name='ping_data')
```

Listing 7.5: Code for ICMP testing with hping3

```
# name:      hping3_tester
# author:    m.sonnberger
# date:      02.06.2023
# use:       ICMP tester for measurements of RTT, jitter and PER with
#            fixed interval

#           sends COUNT ICMP packets with a size of SIZE  and with a fixed
#           sending INTERVAL to a network device HOSTNAME and measures RTT

import os
import sys
import numpy as np
import pandas as pd
import sys
import os
import psutil
import math

# target
HOSTNAME = "10.10.0.200"

# measurement configurations
COUNT = 10000
INTERVAL = "u40000"
SIZE = 86                       #128

# set scheduler priority of code to REALTIME (highest)
os_used = sys.platform

# Set highest priority for the python script on the CPU
process = psutil.Process(os.getpid())
if os_used == "win32":
    process.nice(psutil.REALTIME_PRIORITY_CLASS)
elif os_used == "linux":
    pid = os.getpid()
    os.system("sudo renice -n -20 -p " + str(pid))
else:
    process.nice(20)

# command shell
command = f"sudo hping3 -1 -c {COUNT} -i {INTERVAL} -d {SIZE} -t 2000 {HOSTNAME}"

# analysis variables
rtt = np.zeros([COUNT])
rttUnfiltered = np.zeros([COUNT])
firstRttData = 18
nRttData = 7
stringOffset = 13

# program start
```

```python
process = os.popen(command)
preprocessed = process.read()

strings = preprocessed.split()
receivedData = math.ceil((len(strings)-stringOffset)/7)

for i in range(receivedData):

    rttUnfiltered = strings[firstRttData + i*nRttData]
    rtt[i] = rttUnfiltered[4:8]

# data processing
rttData = pd.DataFrame({'rtt': rtt,
                        })

rttData.to_excel('/home/factory/rtde/ws/data/hping3_response.xlsx', sheet_name='hping3_data')
```

Listing 7.6: Code for TCP testing client

```python
# name:      tcp_tester_client
# author:    m.sonnberger
# date:      12.04.2023
# use:       TCP client for measurements of RTT, jitter and PER

#           sends packets to server and waits till packets
#           arrive from server, measures time till reception

import socket
import time
import numpy as np
import statistics
import pandas as pd

import sys
import os
import psutil

# traget
HOST = '10.10.0.200'
PORT = 30004

SLEEPTIME = 1/1000

#set scheduler priority of code to REALTIME (highest)
```

```python
os_used = sys.platform

# Set highest priority for the python script on the CPU
process = psutil.Process(os.getpid())
if os_used == "win32":
    process.nice(psutil.REALTIME_PRIORITY_CLASS)
elif os_used == "linux":
    pid = os.getpid()
    os.system("sudo renice -n -19 -p " + str(pid))
else:
    process.nice(20)

# packet payload in byte
payload_size = 1024

# number of packets
packets = 10

# number of bursts that send n number of packets
bursts = 500

payload = b'0' * payload_size

# rtt mean per burst
rtt_burst = np.zeros(bursts)

# error rate per burst
error_rate_burst = np.zeros(bursts)

# jitter mean per burst
jitter_burst = np.zeros(bursts)

print("\n\rtesting\n\r")

# loops the number of bursts
for j in range(bursts):

    # sets rtt,errors and jitter to zero for every burst
    rtt = np.zeros(packets)
    errors = 0
    jitter = np.zeros(packets-1)

    # create socket and connect
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(30000)
    sock.connect((HOST, PORT))

    # loops the number of packets in every burst
    for i in range(packets):

        payload = i.to_bytes(1,'big') * payload_size

        # measuring time before and after S/R
        start_time = time.perf_counter()

        # send and receive payload
        send = sock.send(payload)
        receive = sock.recv(payload_size)

        end_time = time.perf_counter()

        # rtt in ms
        rtt[i] = ((end_time - start_time) * 1000)

        # if sent packet is different from received packet --> error
        if payload != receive:
            errors = errors + 1

        # jitter = delta of rtt1 & rtt2
        if i != 0:
            jitter[i-1] = abs(rtt[i-1] - rtt[i])
            # print(jitter[i-1])

        time.sleep(SLEEPTIME)

    # take mean of rtt from every burst and store it in vector
    rtt_burst[j] = statistics.mean(rtt)

    # take mean of jitter from every burst and store it in vector
    jitter_burst[j] = statistics.mean(jitter)

    # store PER from every burst into vector
    error_rate_burst[j] = ((errors / packets) * 100)


    # dispaly progress
```

```python
    print(" %d%%" % ((j / bursts)*100), end="\r")

    # close socket between bursts
    sock.close()

# write data to xlsx file
headline = pd.DataFrame({'rtt': rtt_burst,
                         'jitter': jitter_burst,
                         'PER': error_rate_burst})
headline.to_excel('/home/factory/rtde/ws/data/tcp_tester_data.xlsx',
                  sheet_name="TCP Tester Data")

# for readability
for j in range(bursts-1):
    print("%.6lf %.6lf %.6lf" % (rtt_burst[j], jitter_burst[j], error_rate_burst[j]))
```

Listing 7.7: Code for TCP testing server

```python
# name:      tcp_tester_server
# author:    m.sonnberger
# date:      12.04.2023
# use:       TCP server for measurements of RTT, jitter and PER

#           receives packets and routes them back

import socket
import sys
import os
import psutil

# target
HOST = "192.168.1.58"
PORT = 30004

BUFFERSIZE = 1024

# set scheduler priority of code to REALTIME (highest)
os_used = sys.platform

# Set highest priority for the python script on the CPU
process = psutil.Process(os.getpid())
if os_used == "win32":
    process.nice(psutil.REALTIME_PRIORITY_CLASS)
elif os_used == "linux":
    pid = os.getpid()
    os.system("sudo renice -n -19 -p " + str(pid))
else:
    process.nice(20)


with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
   sock.bind((HOST, PORT))
   sock.listen()
   while True:
        conn, addr = sock.accept()
        data = conn.recv(BUFFERSIZE)

        while data:
            conn.send(data)
            data = conn.recv(BUFFERSIZE)

        conn.close()
```

Statement of Affirmation

I hereby declare that all parts of this thesis were exclusively prepared by me, without using resources other than those stated above. The thoughts taken directly or indirectly from external sources are appropriately annotated. This thesis or parts of it were not previously submitted to any other academic institution and have not yet been published.

[illegible]

Dornbirn, 20.08.2023 Michael Sonnberger