Technology
Arts Sciences
TH Köln

Cologne University of Applied Sciences
Faculty of Information, Media and Electrical Engineering

Bachelor Thesis

Investigation of Slice Characteristics in 5G Campus Networks

Cologne University of Applied Sciences
Bachelor of Computer Engineering (B.Sc.)

Author: Arn Jonas Dieterich
Address: Nikolaus-von-Dorne-Str. 15, 50769 Köln
Matriculation number: 11135427

1st Examiner: Prof. Dr.-Ing. Andreas Grebe
2nd Examiner: Prof. Dr. René Wörzberger

Cologne, 29. September 2022

Bachelorarbeit

Titel: Untersuchung von Slice-Charakteristiken in 5G Campus-Netzwerken

Gutachter:
Prof. Dr.-Ing. Andreas Grebe (Technische Hochschule Köln)
Prof. Dr. René Wörzberger (Technische Hochschule Köln)

Zusammenfassung:
Die Einführung von 5G Network Slicing ermöglicht den Betrieb von virtualisierten Mobilfunknetzen mit unterschiedlichen Quality of Service (Qos) Eigenschaften. Diese These untersucht die verschiedenen QoS-Eigenschaften anhand des 5G Campus Netzwerkes im 5G Co:CreationLab in Köln. Diese Untersuchung basiert auf der Implementierung von mehreren 5G-fähigen Messproben, die die Netzwerk-Performance der unterschiedlichen Network Slices messen. Das verwendete Framework für die Messungen, Speicherung und Visualisierung der Resultate basiert auf einem TIG-Stack, der die Software-Komponenten Telegraf, InfluxDB und Grafana verwendet.

Stichwörter: 5G, 5G Network Slicing, Quality of Service, TIG-Stack

Datum: 29. September 2022

Bachelor Thesis

Title: Investigation of Slice Characteristics in 5G Campus Networks

Reviewers:
Prof. Dr.-Ing. Andreas Grebe (Cologne University of Applied Sciences)
Prof. Dr. René Wörzberger (Cologne University of Applied Sciences)

Abstract:
The introduction of 5G network slicing has enabled the implementation of virtualized cellular networks with disparate Quality of Service (QoS) characteristics. This thesis strives to investigate the different QoS-characteristics using the deployment of the 5G Campus Network at the 5G Co:CreationLab in Cologne. The means of achieving this, are based on the deployment of multiple 5G-enabled measurement probes, that will measure the performance of the network slices. The framework implementing the measurement collection, storage and visualization of results is based on a TIG-stack, that utilizes the software components Telegraf, InfluxDB and Grafana.

Keywords: 5G, 5G Network Slicing, Quality of Service, TIG-Stack

Date: 29st of September 2022

Contents

Introduction

This thesis strives to investigate and evaluate the various performance characteristics of Network Slices in 5G Campus Networks. The ever-increasing demand for cellular connectivity drives the development of next-generation cellular standards.

The most recent release of the 5G cellular standard seeks to meet and exceed these demands. In addition to laying the foundation for higher bandwidths and lower latency cellular environments, many more significant improvements have been introduced along with the 5G standard. One of these enhancements comprises 5G Campus Networks, which are self-contained private networks limited to a specific location facilitating various improvements regarding latency, coverage and security.

Furthermore, the concept of network slicing was introduced in 3GPP's Release 16. This concept allows for the formation of a cluster of disparate virtualized cellular networks, that operate on top of the same physical hardware. These networks however, do not offer the same set of services, but meet specific demands tailored to explicit audiences and application landscapes.

The transition to a new cellular standard and adoption of its new features and capabilities is a gradual but steady process. Nevertheless, the demand to utilize these advanced cellular capabilities is already here. In order for developers to gather an understanding of what is currently viable, a quantitative approach to evaluating and assessing the current potential of the 5G standard is required [12] [35].

1.1 Motivation

This thesis examines two specific segments of the 5G standard, namely 5G campus networks and network slicing. Campus networks have a huge potential in a wide range of industries. Various use-cases for modern and emerging technologies exist, that leverage the 5G cellular communication technology to operate. Examples range from robotics, to autonomous driving and even the medical sector. Furthermore, network slicing in the context of campus networks can aid in providing different types of devices and machines with a matching Quality of Service (QoS) level in order to operate with the highest degree of efficiency possible.

Even end-consumers with mobile devices can profit from network slicing, since the diversity of services and applications being utilized by mobile devices has changed immensely over time. Nowadays, end user's needs are not satisfied by providing simple messaging services. They want to stream videos in high resolutions, play mobile games, conduct live voice- and video telephony and even use augmented- and virtual reality applications in real-time. These types of applications stem a wide variety of QoS-requirements for the service provider. Therefore, segregating the users by application and placing them on a matching Network Slice, allows the service provider to meet the applications demands and fulfill the end-users needs by providing a smooth experience for them while interacting with their network-enabled device.

However, implementing network slicing and deploying operational campus networks at companies and various other institutions takes time. It is of utter importance to test current implementations of these recent features. This enables possible users and companies to conduct a feasibility study and compare these technologies with other competing solutions. Network connectivity with particular QoS-characteristics is a fundamental constituent for numerous applications and technologies, such as autonomous driving, Internet of Things, Smart Cities, cloud gaming and many more. In order to develop and operate software in these segments, developers and other involved parties need to know to what degree these network requirements are fulfilled.

Besides the business context, aiding research and education of modern cellular network technology is valuable. Many students will have had little exposure to the development of applications based on 5G cellular technology and can employ these capabilities in their projects and future endeavors. This can lead to an acceleration of the adoption of 5G cellular technology and improve the availability of high performance mobile broadband applications [89] [12].

1.2 Objective

This thesis aims to evaluate network slicing in 5G Campus Networks.

This objective implies several sub-goals, namely an operational hardware, network and software environment to conduct measurements, store and evaluate these. The goals in regards to the hardware utilized throughout the project, are to deploy several measurement probes in order to carry out simultaneous measurements or even execute stress-tests on the network. Therefore, creating a hardware platform with 5G-enabled modems is a foundational objective, that lays the basis for further configuration and testing.

Additionally, establishing access to an operational 5G Campus Network that utilizes network slicing is of essence. Once connectivity to the network is established, access to the configuration of the Network Slices and knowledge on how these Slices can be configured is required. Due to the ongoing development, implementation and deployment of the 5G standard in the 5G Co:Creation Lab, not all Network Slices that have been advertised or released in the 5G specification or by other service- and network providers have been implemented. Therefore, this thesis reviews what type of Slices are operational, how they can be configured and most importantly how they perform in terms of QoS-characteristics. Subsequent to finalizing the Network Slice configuration, measurement data is gathered over an extended period of time. This data comprises measurements on latency, upload- and download bandwidth and uptime. Lastly these measurements are evaluated per Network Slice configuration and location and compared to measurements or specifications from similar wireless- and cellular network media.

The objective in regards to the software entails developing a measurement framework based on a TIG-Stack. This so-called TIG-Stack comprises three components, namely Telegraf responsible for data aggregation, InfluxDB subject to storing the measurement results and Grafana for visualizing and organizing the results in the desired format. The deployment of these components takes place using Docker. This speeds up deployment, improves security and simplifies administration.

1.3 Thesis Structure

This section presents an overview of the structure and outline of this thesis. The first chapter outlines the aim of this thesis and provides an introduction into the concepts and technologies involved in this thesis. Furthermore, the section on motivation discusses why this subject is relevant today and how an deeper dive into investigating the 5G cellular standard and specifically network slicing and 5G Campus Networks aid in deploying future and emerging technologies. Furthermore, this segment discusses some of the use-cases that are of significance in today's applications landscape and can profit from 5G connectivity. Subsequently, a section covering the objectives of this thesis demonstrates what this thesis strives to achieve. This entails a brief description of the steps and sub-objectives involved in accomplishing this, as well as identifying the main goals relevant to investigate Network Slice characteristics in 5G Campus Networks. To conclude the introductory chapter, a demonstration of the thesis structure provides the reader with an elaborate overview of this report.

The second chapter depicts the technical background information on the topics networking, hardware and software. The networking aspects comprise details on the 5G standard, 5G network slicing and 5G Campus Networks. Furthermore, this section discusses the term Quality of service (QoS) and what QoS-characteristics are of significance. Thereafter, the section on hardware dives into component- and system requirements, needed to create a 5G-enabled measurement probe. Furthermore, the necessity and function of measurement-endpoints is put into context. The segment on software discusses the function of the individual TIG-stack components and how they can be utilized to conduct measurements. Moreover, the containerization platform Docker is covered that is used to deploy the software stack. In addition, the utilities employed for conducting measurements are outlined, as well as the operating system that will execute these tools.

The third chapter defines the solution concept of this thesis. This segment comprises the overall architecture of the software components, by depicting the interfaces they expose, as well as the interaction between the components. Furthermore, the concept of the hardware infrastructure is presented, involving the measurement probes and measurement end-points. Thereafter, a section on the testing strategy is examined. This section entails the methodology of the measurement process and depicts the necessary steps to achieve the desired measurements results. Thereafter, the key measurement characteristics are identified and explained. Subsequently, a quantitative definition of Key Performance Indicators (KPIs) is conducted, which are later used to evaluate the network performance.

The fourth chapter covers the implementation of the measurement setup and software. This entails the assembly of the required hardware components, including the installation of the elements enabling 5G-connectivity. Moreover, this process

comprises the implementation of the connection-establishment procedure to the 5G network slices. Thereafter, the implementation of the TIG-stack is discussed. The implementation involves the configuration of the software components, as well as the implementation of the measurements using the appropriate utilities and parameterising them accordingly. Subsequently, this implementation is demonstrated using a Proof of Concept (PoC) deployment. Furthermore, this section covers the final deployment of the measurement probes in the 5G Campus Network and illustrates the location of the probes with the cellular infrastructure.

The fifth chapter depicts the network results and evaluation thereof. The network results are illustrated per QoS-characteristic and explained accordingly. Thereafter, an evaluation of the achieved results is covered by comparing each QoS-characteristic for each slice with the defined KPIs. Lastly, outages and availability are discussed and assessed.

To conclude, the sixth chapter provides an overview of the overall process and results using a summary. At last, the outlook provides an outline of possible future developments and undertakings.

Technical Background

The following chapter depicts technical background information in regards to areas of networking, hardware and software. These explanations aid the reader in understanding the fundamental technologies and concepts employed throughout this work and form the groundwork of this thesis.

2.1 Network

The field of computer networking, especially cellular standards are omnipresent in this subject and lie at the core of this undertaking. At first, the 5G cellular standard is covered, after which more specific segments of 5G, namely 5G network slicing and 5G Campus Networks are discussed. To conclude this chapter, several aspects of QoS, including the measurement thereof are emphasized.

2.1.1 5G Standard

The most recent release of mobile cellular standards is the 5G standard and was released by the 3rd Generation Partnership Project's (3GPP). This standard supersedes last generation's 4G LTE cellular standard [15]. The 3GPP is the standardization body for mobile standards since 3G, and produces the technical specifications and reports that aid in the implementation of cellular standards by service- and network providers. The 3GPP unites telecommunications standardization bodies world wide, to from the basis for current and future mobile technologies [1].

The 3GPP works in so-called "Releases", with some releases being further divided into sub-releases named "Drops". Multiple Releases are worked on parallel to each other, in order to provide stable specifications, that can be implemented and further improved using stable future publications. These Releases can assume several different statuses, that indicate progress and finalization of that Release. The status "Open" indicates a Release that is in progress, hence the specifications is still being changed and improved. The status "Frozen" indicates that no further functionality can complement the current release, however final details can still be applied. The status "Closed" indicates the finalization of all specification, and

expresses the conclusion of that Release [3] [35].

As of now, there are five Releases concerning the 5G Standard, namely Release 15 through 19 [2]. The initial Release concerning the 5G standard, was Release 15 and was started in 2016. The first Drop comprised the specification for the 5G non-standalone (NSA) architecture. The NSA architecture constitutes 5G functionality that is based on the 4G core. Albeit this dependency on the 4G core, the NSA architecture improves network performance overall. Furthermore, this Drop introduced the specification for 5G New Radio (NR) technology, forming the basis for the 5G standard [35].

The second Drop of Release 15 added the 5G standalone architecture (SA), allowing the 5G technology to operate autonomously, hence without the support of the 4G core. This Drop therefore includes the introduction of the 5G core architecture. Moreover, a significant 5G network concept, namely the Ultra Reliable Low Latency Communication (uRLLC), constitutes this Drop (further discussion of uRLLC follows in Chapter 2.1.2 - 5G Network Slicing) [35].

The third and final Drop of Release 15 included the freeze of the physical layer and protocol specification. This formed the basis for commercial implementation and deployment by service providers to deliver 5G communication technology to the masses. Release 15 assumed the "Frozen" status in June of 2019 [35] [2].

The subsequent Release, Release 16, was initiated in March of 2019. This Release 16 introduced the concept of Network Slicing, which is crucial to this thesis. Furthermore, this Release published functionality regarding private 5G networks, such as 5G Campus Networks. These features are more relevant to enterprises, due to the advanced prerequisites for certain applications in the enterprise context. Release 16 assumed the status "Frozen" in July of 2020 [2] [35].

Release 17 began in June of 2018. This Release further improved upon features from Release 15 and 16, such as Dynamic Spectrum Sharing (DSS) and further enhancements to private 5G networks. New features include 5G satellite access including the compatibility of 5G NG radio for satellite communication. The Release was frozen in June of 2022 [2] [35].

2.1.2 5G Network Slicing

The concept of Network Slicing is new to cellular communication standards and was introduced with Release 16 by the 3GPP [35]. It offers unprecedented flexibility in tailoring network characteristics to application specific demands. This is achieved by enabling the operation of multiple virtualized mobile networks, hosted on the same physical infrastructure. These virtualized mobile networks are referred to as "Network Slices". Slices can offer entirely different QoS-characteristics, that enable the serving of a specific use case. This flexibility is essential, due to the tremendous differences in network characteristics required for an application to operate and behave as intended. An example for these disparate network demands, is high resolution video streams compared to autonomous driving. While high resolution on-demand video streams require a large bandwidth, they are less dependent on packet loss (reliability of the channel) and latency, due to the ability to buffer a certain amount of the transmitted content. In contrast, autonomous driving requires less bandwidth to transmit sensor information, however requisites immaculate latency and reliability characteristics, as tardy arrival of sensory information or even total loss of transmitted packets can lead to fatal accidents. Due to the vast contrast of application requirements in the context of computer networks, Slices can improve the application performance and lead to resource-savings for the network provider. Furthermore, with a growing interconnection of appliances and machines, such as in the context of Internet of Things, Industry 4.0, Smart Cities and autonomous driving, Network Slicing can fulfill the particular QoS demands and enable these segments to thrive [5] [89].

The International Telecommunications Union Radio Communication Sector (ITU-R) defines three main services that can benefit from the service-oriented approach of Network Slicing. These services are defined using certain capabilities according to the ITU-R, namely Peak data rate, User experienced data rate, Latency, Mobility, Connection density, Energy efficiency, Spectrum efficiency and Area traffic capacity. Further capabilities that can enhance future service definitions regard Security and privacy, Operational lifetime, Resilience and Reliability. The main service categories currently defined are ultra Reliable Low Latency Communication (uRLLC), massive Machine Type Communications (mMTC) and enhanced Mobile Broadband (eMBB) [88].

The service segment most familiar to the common end-user is called enhanced Mobile Broadband (eMBB). This type of service is utilized by end-user to browse on their phone, use messenger services, stream music or videos and play online games. The most significant capabilities this service should imbed are proficient peak data

rates, high user experience data rate, a large area traffic capacity, great mobility and considerable coverage. Further use-cases regarding eMBB are Cloud Gaming, Cloud Office Services and Augmented-, Virtual- and Mixed reality applications [88] [57] [89].

Services categorized into the uRLLC segment, have high demands in regards to latency and availability. Possible use-cases and areas of application for this slice are autonomous driving, Industry 4.0, Smart Medicine, Tactile Internet as well as Augmented-, Virtual- and Mixed reality applications. The most relevant capabilities for this service according to the ITU-R are latency, reliability, mobility and mobility interruption time. However, resilience, security and privacy capabilities take on an important role in this field, as for example a remote surgery and mission critical applications utilizing 5G Network Slicing should consider redundancy and security aspects as essential [88] [57] [89] [74].

Services classified by mMTC cover the interconnection of a large number of machines and devices in a limited geographical area. This segment finds application in the field of Internet of Things, Smart Cities, Smart Mobility and traffic telemetry. The essential capabilities required for these services to operate, comprise a substantial connection density and high network energy efficiency. For example, monitoring agricultural or industrial plants using microcontrollers with a limited battery to collect sensory information, can benefit exactly from this set of capabilities, to interconnect and aggregate information to identify disruptions and raise the production efficiency [88] [57] [89] [74].

A further technology called Low Latency Low Loss Scalable Throughput (L4S) developed by the Internet Engineering Task Force (IETF) introduces the ability to server applications with low latency and high throughput demands. This technology has similarities with uRRLC, however strives to achieve higher throughput performance, which can benefit applications in the area of online gaming as well as in the fields of Augmented-, Virtual- and Mixed reality applications. L4S achieves these characteristics by utilizing the so-called Explicit Congestion Notification (ECN) in the IP or TCP header, to signal congestion to the network. This signalling prevents packets from being dropped and leads to an improved reliability of the network overall. ECN is used in combination with a scalable congestion control algorithm, such as Self-Clocked Rate Adaptation for Multimedia (SCReAM), to process the congestion signals at the sender- and receiver end. An implementation of this technology by Ericsson has yielded excellent results, with latencies in the range of 1ms to 50ms and a guaranteed delivery level of 99.9% up to 99.999%. The introduction of this technology has the capability to drastically change the ability to deploy and operate modern and emerging time-critical applications [14] [82] [78].

2.1.3 5G Campus Networks

5G Campus Networks are private cellular networks restricted to a certain geographical area. This allows for the deployment of custom-tailored network characteristics that match the application's demands deployed within the infrastructure. Benefits provided by these type of networks include improved latency, throughput, coverage, availability and connection density. Furthermore, the complete decoupling of the Campus Network from the public cellular infrastructure, yields great security and privacy benefits. This isolation from public infrastructure includes the deployment of a dedicated Network Slice with a private a Access Point Name (APN)/ Data Network Name (DNN). Moreover, 5G Campus Networks possess a dedicated frequency spectrum, which is assigned by the authorized agency of that country, for example the Federal Network Agency in Germany. The spectrum assigned for Campus Networks within Germany comprises a 100 MHz band from 3.7Ghz to 3.8Ghz, and can be leased for a certain period of time within a restricted geographical area. This allows for the private deployment of Campus Networks by a large number of businesses and production facilities.

The main field of application for 5G Campus Networks lies in the industrial sector. The flexibility of wireless communication with custom QoS-characteristics and high service quality standards, allows for a more effective and adaptable manufacturing environment. Quantitatively speaking, some of the figures that have been tested under laboratory conditions by T-Systems offer peak data rates of up to 20Gbit/s, while keeping the latency below 20ms. Furthermore, 5G Campus Networks can pack one million devices per square kilometer, while assuring service quality requirements [34].

Campus Networks are not restricted to industrial production environments, the logistic sector, maintenance and repair, AR applications as well as company premises such as employee offices can benefit from 5G Campus Networks. Additionally, besides the deployment and operation of a private network, companies such as the Telekom improve the coverage of the public cellular infrastructure, when deploying their 5G Campus Network solution [12] [33].

2.1.4 Quality of Service

The term Quality of Service (QoS) describes the general performance of a network. Some of the most significant characteristics that define QoS are latency, throughput, jitter, reliability and availability.

Latency describes the sum of transmission, propagation, queuing and processing delays across the network. When using tools such as "ping" from our user-equipment (UE) of choice, we measure the round-trip-delay, hence the time it takes for a packet or alternative Protocol Data Unit (PDU) to travel across the network and back to

the sender. Throughput is a measure of the amount of bits that successfully arrive while travelling the path from source to a destination. This measure is most often expressed in Mbit/s and Gbit/s. The term jitter describes the variance in latency across the network. This characteristic is mostly measured in μs. Availability describes the overall uptime of a network. This uptime can concern a certain set of application layer service, or refer to lower level protocols such as ICMP on the network layer. This measure is mostly expressed as a percentage. The characteristic reliability refers to the error rate of data transmissions. Thus, the amount of times data do not successfully arrive at the destination. In the 5G standard QoS-characteristics are expressed using the 5QI standard. [4] [76] [73] [75].

2.2 Hardware

The subsequent section provides an overview of the hardware requirements and type of hardware necessary to facilitate this thesis. The main components involved in the investigation of Network Slicing in 5G Campus Networks comprise the measurement probe host, a 5G cellular modem with antennas, as well as measurement end-points hosted in public or private infrastructure.

2.2.1 Measurement Probe Host

The main system powering the modem along with the software necessary to aggregate measurement results lies at the core of the measurement setup. Various types of computer-systems can fulfill this role, ranging from low-powered microcontrollers, to single-board computers such as a Raspberry Pi, to open-source router-boards powered by LibreCMC and OpenWRT, to industrial routers or even normal desktop computers and laptops. Each of these host systems have advantages and trade-offs regarding their performance, energy consumption, heat dissipation and mobility with respects to the size and weight and ability to run off a battery. Furthermore, the number and type of hardware interfaces these systems offer are of importance, due to high bandwidth and redundancy requirements as well as the ability to establish network connections using different layer 1 and layer 2 technologies (Open Systems Interconnection (OSI) model).

In terms of must-haves, the host system must include a M.2 Key-B slot or extension that can house the 5G cellular modem for 5G connectivity. Furthermore, the M.2 slot and surrounding components must offer enough space for the rather large modem, with the dimensions of 52mm x 30mm x 2,3mm. The performance and interfaces of the system must allow for bandwidths north of 2.5 Gbit/s to be processed in real-time, in order to prevent a hardware related bottleneck during the measurements. In addition, a second network connectivity option for administration and data transfer must be incorporated in the setup host system [70] [71].

2.2.2 5G Cellular Modem

The cellular modem enables 5G connectivity of the host system. Due to the specific network environment, the modem must support the 5G standalone (SA) mode, as this modus operandi is utilized in the 5G Campus Network, where the measurements are conducted. Furthermore, the modem must support the frequency bands for 5G cellular technology in Germany, in particular the frequency bands for 5G Campus Networks that range from 3.7Ghz to 3.8Ghz [77] [12].

Moreover, the modem must support operation using open-source firmware, due to an improved flexibility when choosing a host operating system as well as better

deployment options and independence from closed-source software that must be purchased or cannot be acquired readily.

2.2.3 Antennas

The modem requires further hardware for operation, namely antennas. The antennas enable the receipt and propagation of radio waves to facilitate the 5G wireless communication. As mentioned in the prior sub-section regarding the modem, the 5G Campus Networks operate in a specific frequency band, which must be supported by the antennas as well. Additionally, these antennas must be connected to the modem using MHF4 pigtails and be held by an encasing or other structure to allow for proper propagation of the radio waves [36].

2.2.4 Measurement Endpoints

In order to conduct network performance tests and gather measurement results, a measurement end-point must to be defined and setup. This end-point needs to host the appropriate software to perform the measurements and be capable of communicating using the required network protocols utilized by the tests. This includes the provisioning of the appropriate network stack and configuring the according firewall rules. In addition, the adequate provisioning of computer resources such as compute, storage and network resources must be fulfilled. Especially network resources, such as network interface (NIC) bandwidth must be sufficient to avoid bottlenecks on the measurement end-point.

The setup and access to the requisite hardware can occur using hosting providers in the public-, or private cloud with dedicated hardware. Furthermore, hosting a measurement end-point within the network that tests are conducted against, as well as using different hosting providers and private solutions hosted in disparate network segments, aids in providing a enhanced overview of the general performance of the network.

2.3 Software

The ensuing section covers the software frameworks and technologies employed throughout this thesis. Moreover, a brief section depicts operating system and firmware details applicable to the testing setup.

2.3.1 TIG Stack

The principal software components enabling the measurements in terms of aggregation, storage and visualization are Telegraf, InfluxDB and Grafana. The following sections dive into each component and describe their relevance and function regarding the network measurements.

Telegraf

The open-source software Telegraf functions as the aggregation and processing agent on the measurement probe hosts. The agent is plugin based with a diverse and active community that has developed over 300 plugins serving various segments, such as the collection of metrics on IoT sensors, host performance, web services, orchestration tools and database operation. The tool is written in Go and can be installed as a single binary without requiring any further dependencies.

The operation of Telegraf is divided in to four major steps. Firstly, "Input plugins" collect metrics. The second step involves "Process plugins" to filter and adapt the collected data to yield the desired format. The third steps covers "Aggregator plugins", allowing the application of mathematical accumulation operations, such as calculating the minimum and maximum on a set of results. The last step deals with "Output plugins" and performs the final write operation to the desired data source, such as a database [46] [45] [47].

InfluxDB

InfluxDB serves as the Database Management System (DBMS) for storing measurements, as well as the source for visualizing the results. Furthermore, it is utilized for monitoring the systems and networks involved in conducting the measurements. The DBMS was developed by InfluxData Inc. and is open-source software published on GitHub. One of the main advantages of this platform is the optimized storage of timestamped data. This characteristic causes all data points to receive time labels, and allows for faster and more efficient processing of the measurement results, in comparison to common relational databases. Furthermore, InfluxDB offers a variety of Application Programming Interfaces (APIs), that can be employed for querying and storing measurements. Theses APIs allow for a more distributed architecture,

and enable the measurement hosts to communicate with the database from other network segments [44] [50].

Grafana

Grafana is the platform facilitating the visualization and analysis of the measurement data. This software offers a modern GUI with numerous options for illustrating time series data. The adaptability and customizability allows for an insightful depiction of the gathered data, to help a user understand the current state as well as analyse the root cause of a failure or predict future developments.

Grafana offers a seamless and secure integration with InfluxDB, to retrieve, aggregate and visualize the results. The advanced aggregation capabilities for time series data, allow for a rapid evaluation and monitoring of the subject infrastructure. Possible aggregation options for the dashboards include, the calculation of the sum, average and minima or maxima of a set of results. The dashboards are comprised of panels, which can be grouped together for an improved oversight. Panels can illustrate data using various types of constructs, such as graphs, histograms, geomaps, gauges, heatmaps, lists and many more. Therefore, the tailored approach to time series data, flexibility regarding the depiction of data and enhanced integration with InfluxDB makes this platform ideal for the illustration of network performance in the context of 5G Campus Networks [63] [62] [60].

2.3.2 Docker

The deployment of a TIG stack involves the integration of various package sources, installation of packages, configuration of the operating system environment and the software itself. In order to speed up and automate this process a containerization tool can aid in this undertaking. Docker offers a simple yet powerful solution to achieve this goal and is therefore used throughout this thesis.

Docker enables the creation, deployment and management of resource efficient and platform independent containers. Due to the light-weight nature of process-based containers, images take up less disk space and require less computing-resources than ordinary Virtual Machines (VMs). Moreover, the container abstraction adds an additional layer of security to the host system, making privilege-escalation attacks more complex, as well as improving availability due to the enhanced redundancy capabilities and easy re-deployment of the containers. This leads to an acceleration of the development and a more secure operating environment of the desired application [23] [93] [37].

Dockerfile

A Dockerfile comprises the definition of the container, including the base image, dependencies, environment variables, files, volumes, networks, exposed port and entry-point for the application. The commands in a Dockerfile are standardized and similar to Linux-based OS-commands. The Docker Engine utilizes this Dockerfile to create the image, automating the image creation procedure. Various applications already have pre-assembled Docker images that can be downloaded from public registries online, such as Docker Hub [23] [93] [37].

Docker-Compose

Docker Compose is a container orchestration tool that enables the deployment and management of applications with multiple containerized components. Many applications today can benefit from multiple isolated environments, as the segregation of components improves aspects such as security, scaling, redundancy and speed of deployment. For example, in the context of continuous integration code is regularly changed and integrated with the production environment, a segregated containerized approach leads to to less re-deployments as only the improved components are affected. Docker Compose utilizes a Yet Another Markup Language (YAML) file called *docker-compose.yml*, that comprises the definition of the various containers, in form of a reference to their respective Dockerfile, as well as additional parameters such as networks, volumes, exposed ports and deployment sequence. The TIG stack environment offers a suitable application for container orchestration with Docker Compose, due to several distinct yet dependent components working concurrently alongside one other [23] [38].

2.3.3 Network Performance Tools

The measurement and evaluation of network performance requires a suitable toolset to collect the desired metrics. These tools must be capable of interacting with a cellular network interface, to collect the network and interface metrics used to evaluate QoS characteristics. Moreover, these tools must offer a parsable output format or offer an output that is already easily processable (as in the case of JavaScript Object Notation (JSON) and Extensible Markup Language (XML)) for aggregation and storage of the measurements.

These tools can be ready-to-use binaries or in the case of Telegraf, plugins that can be easily incorporated within the TIG-stack. Telegraf also offers the ability to execute Operating System (OS)-commands directly from within the configuration file. This allows for the utilization of tools that stem from a package repository of a Linux-based OS or otherwise installed binaries. Linux repositories offer a wide range of tools for network performance assessments, such as "ping", "hping3", "iperf3"

and many more. The growing list of plugins for Telegraf in regards to network performance and OS metrics, is further employed for the collection of measurements throughout this thesis. The combination of OS-tools and plugins shapes the core of data collection and aggregation, to assess QoS-characteristics such as latency, throughput, jitter and application-layer specific performance such as Domain Name System (DNS) response-time. The choice of tools is essential for creating a meaningful data pool to evaluate network performance [46] [81] [59] [85].

2.3.4 Operating System and Firmware

The choice of OS and firmware play a vital role in creating a stable platform for cellular connectivity and operation of the TIG-stack. Due to the solid integration and simple installation of the TIG-stack on a GNU's Not Unix! (GNU)/Linux OS, a large set of network tools in the package repositories and pre-installed open source firmware for cellular modem support, the OS choice is narrowed down to a GNU/Linux OS [98] [81] [97].

In terms of firmware choice, an open-source solution is favorable, due the prior integration with modern Linux kernels. Closed-source firmware may require registration with a vendor and is not compatible with all modems, requiring a more rigorous installation and platform setup procedure [97].

Further firmware is required for Universal Serial Bus (USB)-connectivity, as well as for the protocols used to communicate with the modems. Next to the ATtention (AT)-command set, also known as Hayes command set, there are various modern protocols that aid in the interaction with cellular modems. Most modern modems support at least on of the following protocols, namely Qualcomm MSM Interface (QMI) and Mobile Broadband Interface Model (MBIM). In order to enable the communication with these protocols additional software and firmware is required [13] [80] [51] [29] [54] [94] [19].

Solution Concept

This chapter demonstrates the concepts and architecture surrounding the measurement setup and procedure. At first, some of the preliminary work achieved thus far is presented. The subsequent section illustrates the architecture of the measurement setup and cellular infrastructure. Consequently, the testing strategy is presented, along with the methodology, key aspects of what is measured and how performance is assessed.

3.1 Preliminary work

As part of the curriculum of the Bachelor of Computer Engineering at the Cologne University of Applied Sciences, students conduct a practical phase that encompasses the execution of a project based assignment, in collaboration with an industry partner. This practical phase was utilized for creating a platform for conducting QoS measurements in 5G Campus Network, having the title "Data Measurement Framework (DMF) Probe for QoS Measurements in 5G Campus Networks". Throughout this practical phase a platform employing a Raspberry Pi 4, along with a Hardware attached on top (HAT), enabling 5G connectivity, was used to test QoS characteristics of the existing 5G Campus Networks at the campus Deutz based in Cologne.

The software framework includes a custom dashboard and collection and aggregation components to aid in the performance measurements and evaluation thereof. These components have limited capabilities in regards to the visualization of the data. Furthermore, the collection and processing of the data is not optimized for time series data. This limits the ability to scale the data collection with more clients and achieve the same level of performance. Furthermore, the Raspberry Pi 4 along with the HAT are not suitable for high bandwidth applications, due to the limited processing capability and interface bandwidth. However, the gained experience from interacting with cellular modems and establishing connections to the 5G Campus Network, forms the foundation for establishing 5G connectivity with the new platform and software stack in this thesis.

3.2 Architecture

This sections defines and clarifies the solution concept of the software and hardware infrastructure and illustrates how these integrate as a whole. The software architecture comprises three main components, namely a collection and aggregation, a storage and a visualization component.

Figure 3.1: TIG-Stack component overview and interaction [31]

The collection and aggregation component, Telegraf, is installed on the measurement probes to gather metrics in the 5G Campus Networks. Telegraf employs a configuration file that specifies at which interval, what metrics are collected, how they are processed and aggregated and where they are sent to for storage purposes. There are numerous plugins developed in Go with the goal of gathering statistics on network performance, network interfaces and OSs activity. Generic plugins exist that allow for the execution of OS commands, allowing tools such as "ping" (installed from the package manager of a GNU/Linux OS) to be executed at regular intervals as well transferring the collected results to a specified remote data source [45] [46].

The storage component, InfluxDB, is responsible for accepting metrics from the measurement probes, as well as handling query requests of the stored measurements. Furthermore, InfluxDB stores and manages the saved metrics on the physical hardware. This DBMS is specifically optimized for time-series data, allowing for an efficient and scalable operation in the context of network monitoring and performance assessment. The Representational state transfer (REST)-APIs exposed by InfluxDB allow remote measurement clients to save their collected metrics, and allow for the

retrieval of those metrics by the visualization dashboard. These REST-APIs are secured with a credential authentication mechanism, known as "Hypertext Transfer Protocol (HTTP) basic access authentication". Further use of Hypertext Transfer Protocol Secure (HTTPS) employing Transport Layer Security (TLS) secures the communication by facilitating integrity, authentication and confidentiality [50] [24] [20].
The visualization component, Grafana, retrieves the measurements from a specified data source in form of a InfluxDB database and displays them in a custom format. Various types of databases and other kinds of data sources such as logging servers have already been incorporated in the visualization framework. This aids in the integration of the desired data source, as only data source specific parameters have to be specified, in order to access the data. Data sources are decoupled from the visualization dashboards, thus one data source can be utilized for several dashboards or one dashboard can retrieve the relevant data from multiple data sources at once. The dashboards are highly customizable and allow for the integration of various panels comprising tables, lists, charts and graphs of various kinds. This allows for the depiction of all essential aspects surrounding QoS characteristics, such as bandwidth (Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)), latency, jitter, availability and application layer performance such as time to first byte (TTFB) and DNS query times. Several measurements of a specific QoS characteristics with different measurement endpoints can be grouped together using rows, leading to a more comprehensible depiction of the overall measurement performance [64] [61].

The hardware infrastructure depicts the hardware hosts that operate the software components and how these are distributed across the infrastructure. The collection and aggregation component is hosted on the measurement clients and is responsible for performing the measurements, aggregating and filtering the results and transmitting these to the storage destination. This task implies 5G-connectivity and the ability to execute the measurements without creating a processing bottleneck. Furthermore, these clients are deployed in the vicinity of the radio interfaces of the 5G Campus Network, hence must comply to certain mobility constraints.

The storage and visualization components are hosted independently from the measurement clients, but can be hosted together on the same server infrastructure. This segregation is advantageous, since the measurement clients are operated with comparatively low computational power and storage space. Hosting the storage and visualization component with a professional service provider allows for a more scalable and high availability approach. Employing virtualized machines allows for flexible resource allocation in regards to compute, storage and bandwidth needs, as well as for a quicker deployment. Moreover, many hosting providers offer simple backup and replication abilities. Next to having disaster recovery options, the vir-

tual machines are operated on virtual clusters that are constantly monitored and have built-in hardware redundancy, such as two or more processors, several hard disk drives (HDDs) operating in a redundant array of independent disks (RAID) setup and multiple network interface controllers (NICs). This hardware redundancy increases availability and leads to a strengthened resilience in case of hardware failures [92] [32] [87]

3.3 Testing Strategy

This section details the testing strategy concept, entailing the methods utilized to conduct measurements, the QoS-characteristics measured and an examination of the metrics that are key to assess to the overall performance of a 5G Campus Network.

3.3.1 Methodology

The methodology of this thesis comprises the depiction of the quantitative approach of data collection and how this data is analysed, in order to evaluate the performance of different slices in 5G Campus Networks.

In the first step the target 5G Campus Network for the QoS measurements is selected. This process entails choosing the location where access to the Radio Access Network (RAN) of the 5G network can be established. Consequently, the available slices of that network are identified along with their parameterization. In the next step, the slices to be measured are chosen. Thereafter, it is determined how many measurement clients are setup, what hardware specification is utilized for each client and how these clients can be deployed in the respective location. In the next step, the measurement probes along with the cellular modem are deployed. This comprises the installation of the OS and firmware, as well as setup of the cellular modem with the antennas. Thenceforth, the connection establishment to the respective 5G network slice is tested for each client.

Once successful network connectivity has been verified, the means of measuring the QoS characteristics are identified. This comprises the identification of the relevant metrics and what data needs to be collected for each metric. In order to accumulate a representative set of measurements for each metric, the number of measurement endpoints is determined, how often data is gathered per measurement cycle and how many measurement cycles are conducted per collection phase. These findings are then implemented using a Telegraf configuration file. The Telegraf agent is installed on the clients with the appropriate configuration and operability verified. The testing and verification of the collection agent comprises establishing an authenticated communication channel with the data source and correct collection of metrics. Furthermore, it is verified that the measurements are transferred

successfully to the respective database.

Thereafter, the data collection phases are executed by the clients. The duration of these phases varies, however, a longer data collection phase provides more data and hence, insights on how the network performs over an extended period of time. Therefore, some phases only cover a period of a few hours, while the long-term measurements might span several days up to a few weeks.

Once the data collection phase concludes, the overall network performance is evaluated. At first, the Grafana dashboards are adapted to illustrate an overview of all the metrics gathered. Thereafter, the data displayed in the panels is further customized by utilizing various aggregation functions offered by the query language of the database, Influx Query Language (InfluxQL), and the Grafana framework itself. The visualization framework is further utilized to generate and export graphs and other visual depictions of the gathered metrics, to illustrate the performance of the network for a specific time interval.

The pre-processed and refined depictions of the measurements are then utilized in the evaluation. The evaluation process entails the comparison of the collected data with the parameterization of the respective slice. Furthermore, the data is compared with the specification of the 5G standard itself and prior cellular standards. Moreover, performance measurements gathered by the Internet Service Provider (ISP), in this case Deutsche Telekom AG, are compared with the measurements collected throughout the data collection phase. Further references that serve in the evaluation, are measurements conducted with a mobile User Equipment (UE). These are one-time measurements conducted with a 5G-enabled phone or tablet using an independent broadband measurement application for evaluating the connection quality of an ISP, such as Speedtest by Ookla and Broadband Measurement by zafaco GmbH [66] [102].

3.3.2 Measurement Key Characteristics

In order to objectively evaluate the performance of 5G Campus Networks and their according slices, it is essential to select significant and meaningful QoS characteristics.

The key characteristic, throughput, determines the number of bits that successfully travel across the network. This characteristic can be utilized to evaluate the bandwidth, thus capacity of a network. Due to the high bandwidth demands of today's applications, such as high resolution video streams, it is fundamental to assess this characteristic. The measurements are conducted on the transport layer and employ two essential transport layer protocols, the connection-oriented TCP and the connectionless protocol UDP. TCP is essential for application layer protocols such as HTTP, HTTPS, Remote Desktop Protocol (RDP), Secure Shell (SSH), Sim-

ple Mail Transfer Protocol (SMTP) and Internet Message Access Protocol (IMAP). These protocols enable web-browsing, remote administration and e-mail messaging. UDP is crucial for application protocols such as DNS, Dynamic Host Configuration Protocol (DHCP), Voice over IP (VoIP). These protocols facilitate mapping domain names to Internet Protocol (IP)-addresses, the dynamic allocation of IP-addresses to hosts and internet telephony. Furthermore, both TCP and UDP play an indispensable role in video streaming, for example when using streaming services such as Netflix, Youtube and Amazon Prime. The tools "iperf3" and "speedtest-cli" as well as further Telegraf plugins allow for the measurement of throughput to and from a specified destination server [8] [84] [46] [72].

The next characteristic, latency describes the delay across the network. This characteristic can be further divided into the sum of propagation, transmission, queuing and processing delays that arise when a packet or other PDU travels from source to destination. This characteristic is crucial in real-time applications, such as video gaming, internet telephony or video conferencing and is therefore assessed in the measurements of the 5G Campus Networks. This characteristic is evaluated on the Network layer (layer 3) of the OSI model using the Internet Control Message Protocol (ICMP) protocol. There are several tools available in the package repositories of GNU/Linux OSs, such as "ping" and "hping3", as well as plugins available for Telegraf, that enable the measurement of network delay using ICMP [73] [10] [46].

The characteristic jitter describes the fluctuation in latency. Jitter negatively affects the quality and experience in a voice call, video conferencing, online games and other real-time and interactive applications. Therefore, it is vital to investigate the degree to which jitter arises in a network, to enable the use of interactive applications and meet QoS requirements. This characteristic is measured using tools, such as "iperf3", My Traceroute (MTR) and "Wireshark" [17] [76] [21] [11].

There are various factors to assess in regards to the availability and reliability of a network. The availability is first of all rated in terms of successful connection-establishment to the network, which includes the correct allocation of an IP-address. Furthermore, the availability and reliability is judged using the network and transport layer of the OSI model, employing multiple different network protocols. On the network layer (layer 3), IP-connectivity and ICMP reachability is measured. This is achieved by utilizing the tools "ping" and MTR, which indicate the availability of IP-connectivity, and in case of erroneous connection attempts, where on the route to the destination the connection fails, and what percentage of packets are dropped. On the transport layer (layer 4), the protocols TCP and UDP are assessed, by evaluating the number of TCP retransmissions and UDP datagram drops. The error rate on the transport layer is measured using the tool "iperf3".

Furthermore, the characteristic DNS-query time is assessed. This metric is mea-

sured on the application layer (layer 8) of the OSI model. The query-time is significant due to the widespread use of domain names in various applications. For example, accessing a website using a domain name requires the execution of a DNS-query to retrieve the IP-address of the appropriate web server. Moreover, sending an e-mail requires querying the responsible mail server of that domain. Therefore, establishing an overview of how query times fluctuate to various domains using different DNS-servers, aids in assessing the general performance of a network. The DNS-query time can be assessed using a Telegraf plugin [49].

3.3.3 Key Performance Indicators

This section covers the definition of KPIs used to evaluate network performance. The KPIs outline the same QoS-characteristics mentioned in the previous section, namely throughput, latency, jitter, error rate and DNS-query time. However, this section depicts specific magnitudes for these characteristics, in order to fulfill a certain level of QoS.

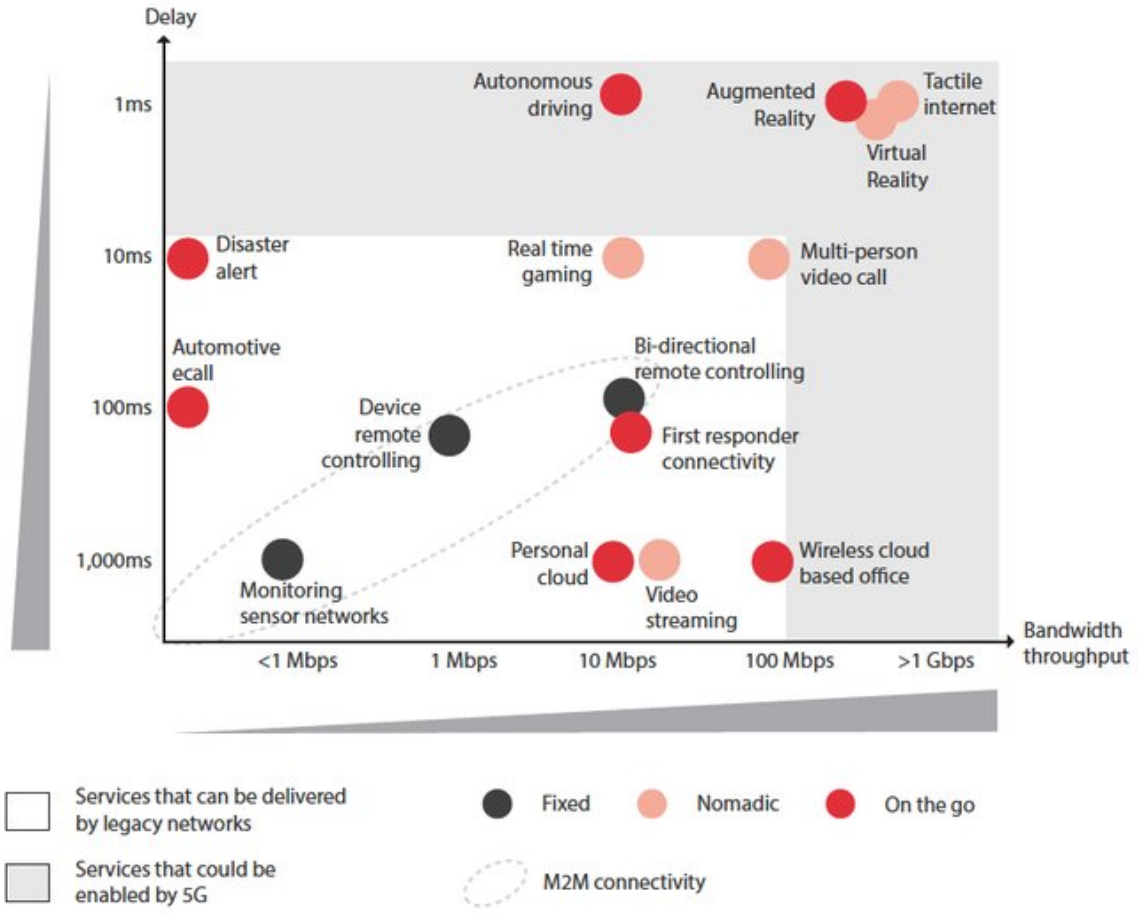


Figure 3.2: QoS-requirements for different applications. [55]

The throughput in the download direction is set at 80Mbit/s or above. The throughput in the upload direction is set at 30Mbit/s or more. These specific in-

equalities are chosen in order to allow for conference video calls and interactive online gaming applications. Furthermore, this criteria fulfills the bandwidth requirements for high definition video streams and VoIP calls. The latency in terms of round-trip time (RTT) is set at 50ms or below. This interval is set to enable interactive applications, such as online gaming, video- and voice telephony, without affecting the experience substantially. The jitter is set to 30ms or less. This is especially important in VoIP and other interactive applications such as gaming. The packet loss is set to below one percent and the TCP-retransmission rate to below two percent. These magnitudes are chosen in order to prevent applications from failing and to reduce the effect on the user-experience. The DNS-query latency is set to less than 100ms, as this latency is feasible using public name servers and does not impact the user-experience considerably, when for example browsing the internet [52] [41] [101] [16] [7] [99] [103].

KPI Name	**Value/Range**	**Protocol**
Throughput (Download)	>80Mbit/s	TCP and UDP
Throughput (Upload)	>30Mbit/s	TCP and UDP
Latency	<50ms	ICMP
Jitter	<30ms	ICMP and UDP
TCP-Retransmissions (Error Rate)	<2%	TCP
Packet Loss (Error Rate)	<1%	ICMP and UDP
DNS-query latency	<100ms	DNS

Table 3.1: Key Performance Indicators

Implementation

This chapter covers the implementation of all aspects surrounding the hardware measurement setup, 5G-connectivity to the Campus Network and software utilized for data collection and aggregation, storage and visualization of the measurements.

At first, the hardware measurement setup is outlined, including all components necessary to establish connectivity to the 5G cellular network. Subsequently the interaction and communication with the 5G modem is presented, including the required firmware and protocols to communicate with the modem. Thereafter, the process of establishing and closing connections to and with the 5G network is illustrated. Consequently, the TIG-stack configuration and deployment is discussed. This section dives deeper into the tools and plugins utilized for the measurements and how these measurements are stored and retrieved for visualization purposes. At last, the locations and networks utilized for the measurements are depicted. This includes an illustration of the cellular infrastructure as well as the configuration of the network slices available in the 5G Campus Network.

4.1 Measurement Tool Setup

The measurement tool setup represents the hardware and OS, that enables 5G connectivity to the Campus Network. Furthermore, this setup facilitates the execution of the software components requisite to conduct all measurements operations.

The host system is based on a Fujitsu Esprimo Mini PC Q-Series with an Intel Core i5 Processor and 8GB of Random-Access Memory (RAM). This specification provides enough processing power to operate the software components, as well as for processing the amounts of data generated during the measurements. Additionally, this system offers two USB 3.0 interfaces, that allow for a high bandwidth connection, capable of supporting the bandwidths of the 5G cellular modem.
The host system is powered using the OS Ubuntu 22.04.1 LTS (Desktop OS). This operating system integrates package repositories that allow for the installation of most software components utilized in the measurement procedure using the package manager. Furthermore, the firmware required to control and communicate with the modems, "qmi_wwan", is already integrated in the kernel version, 5.15.0-41-generic,

used by Ubuntu 22.04.1 LTS [65] [67] [22] [97].

In order to connect a 5G cellular modem, that connects using the M.2 Key B interface, to the host system, an adapter for USB-A to M.2 Key B is required. Furthermore, the modem requires a Subscriber Identity Module (SIM)-card to operate, as the SIM-card facilitates authentication with the Campus Network. The "DeLOCK 63166 interface-card/adapter" fulfills both these functions, and is therefore utilized in the measurement setup. The SIM-cards are supplied by the operator of the network, Deutsche Telekom AG. Additionally, the modem requires antennas for receipt and transmission of the radio signals. The antennas need to support the frequency band of 5G Campus Networks in Germany, that ranges from 3.7Ghz to 3.8Ghz. The antennas employed in this measurement setup are the broadband omni-directional antennas "PWB-BC3G-38-RSMAP" from Panorama Antennas and adhere to the requirements mentioned above. In order to connect the antennas featuring SubMiniature version A (SMA)-connectors with the modem, antenna cables with MHF4-connectors on one end, and Reverse-polarity-SMA (RP-SMA) connectors on the other, are required. The cables utilized throughout this project are the "DELOCK 90398 High frequency (HF) antenna cables". Furthermore, to allow for an adequate propagation of the radio waves, the antennas are held in a holder. The enclosure utilized in this setup is custom designed and 3D-printed holder created by the Makerspace [77] [12] [6] [68] [83] [58].

The fundamental component enabling 5G-connectivity is the modem. The modem must support the 5G Standalone architecture (SA), 5G Campus Networks and the according frequency bands in Germany. There are several modems utilized throughout this project.

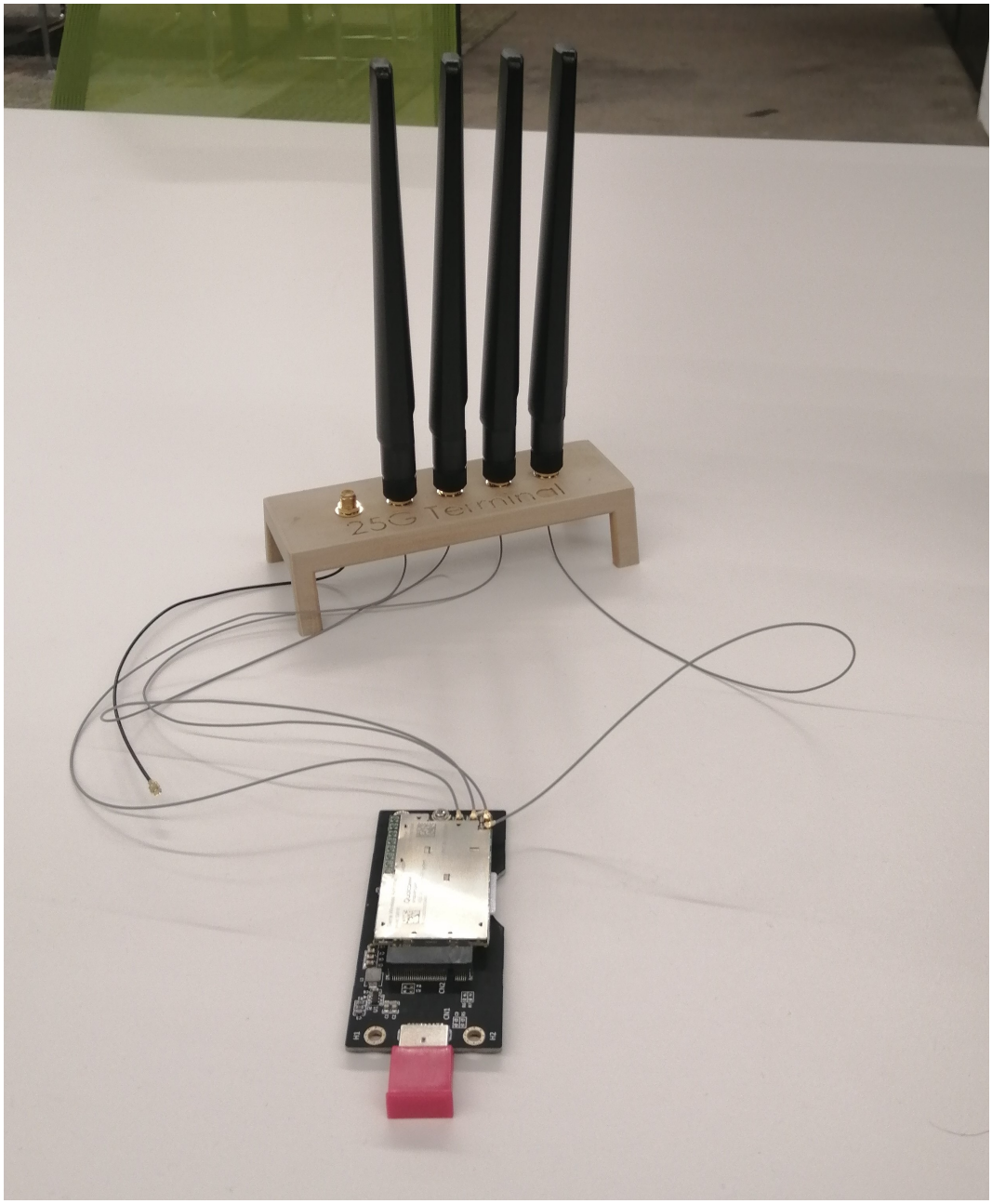


Figure 4.1: Illustration of the modem setup with antennas.

Name	Manufacturer	SA/NSA Support	Upload/Download Bandwidth (Gbps)	Interface	Frequency Bands (5G NR)
RM500Q-GL	Quectel	Yes/Yes	0.90/2.10 (SA) 0.65/2.50 (NSA)	M.2 Key B	n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n48, n66, n71 (SA) n38, n41, n77, n78, n79 (NSA)
EM9190	Sierra Wireless	Yes/Yes	3.00/5.00	M.2 Key B	n1, n2, n3, n5, n7, n8, n12, n20, n25, n28, n38, n40, n41, n48, n66, n71, n77, n78, n79 mmWave: n257, n258, n260, n261
FM150-AE	Fibocom	Yes/Yes	0.90/2.10 (SA)	M.2 Key B	n1, n3, n5, n8, n20, n28, n38, n41, n77, n79

Table 4.1: Overview of the 5G modems employed in the measurement setups [69] [40] [39] [53].

Furthermore, the measurement setup includes multiple measurement end-points. Next to publicly available end-points, three additional systems are implemented in the form of virtual servers. In order to provide an objective result when collecting data, all systems are hosted on disparate infrastructure.

The first server is hosted on the infrastructure of Hetzner Online GmbH in Nuremburg, Germany. This system is powered using three Virtual Central Processing Units (vCPUs) with 4GB of RAM. The server is running the OS Ubuntu 22.04 LTS. The second virtual server is hosted on the infrastructure of Host Europe GmbH, which is a subsidiary of GoDaddy Incorporated. The system is located in Strasbourg, France. This measurement end-point has two vCPUs along with 2GB of RAM and is running on the OS CentOS 7. The last end-point is a virtual machine hosted on a dedicated server running the hypervisor VMware ESXi version 7.2. This server is deployed within the network of the DN.Lab at the campus Deutz of Cologne University of Applied Sciences. The virtual server possesses eight vCPUs and 16GB of RAM. This virtual machine is also running on the OS Ubuntu 22.04 LTS. All three measurement end-points allow communication using the ICMP protocol. Furthermore, all systems are running an iperf3-server, that can handle throughput measurements to and from the server, using the protocols TCP, UDP and Stream Control Transmission Protocol (SCTP). The iperf3-server is implemented as a "Systemd Service", in order to simplify administration and automatically react to a failure or halt of the service and restart it [18].

The measurement end-points are publicly accessible via DNS, in order to account for potential infrastructure changes including an IP-address change. The domain utilized in this process is called "netzmessungen.de", and is registered with Host Europe. The server hosted at Hetzner is connected using the subdomain "srv1.netzmessungen.de. The system hosted at the DN.Lab in the Cologne University of Applied Sciences' network is reachable via the subdomain "srv2.netzmessungen.de". The system hosted at Host Europe is reachable via the subdomain "srv3.netzmessungen.de".

4.2 Modem Interaction

In order to interact with the modem, the modem is connected to the host computer using a USB-3.0 interface with the measurement setup mentioned above. Communication with the modem via Ubuntu's command-line interface (CLI) is required, in order to gather information on the operation of the modem and to establish a connection with the 5G Campus Network. The 5G modems employed during the measurements expose several interfaces, that allow a user to interact with the modem.

```
  System
      device: /sys/devices/platform/scb/fd500000.
      pcie/pci0000:00/0000:00:00.0/0000:01:00.0/usb2/2-1
              drivers: option1, qmi_wwan
              plugin: quectel
              primary port: cdc-wdm0
              ports: cdc-wdm0 (qmi), ttyUSB0 (qcdm),
                       ttyUSB1 (gps), ttyUSB2 (at),
                       ttyUSB3 (at), wwan0 (net)
```

Listing 4.1: ModemManager - Modem Ports

The discovery of the modem is conducted using the ModemManager, the syslog, kernel log and device list of the operating system. Upon connecting the modem with the host, the ModemManager is started in the debug mode. Subsequent to the initialization of the ModemManager, information is gathered on the operability and properties of the modem and SIM-card [86].

```
#!/bin/bash
# Check correct Loading of QMI-WWAN Driver
lsmod | grep -i qmi_wwan

# Start ModemManager with Log-Level DEBUG
ModemManager --debug

# Change Log-Level (optional)
mmcli --set-logging=[ERR|WARN|INFO|DEBUG]

# Scan for WWAN Interfaces
ls -l /var/log/syslog | grep -i cdc-wdm

# Scan for USB devices
ls -l /dev | grep -i usbtty
lsusb -t

# Display Kernel Log USB Information
dmesg | grep -i usb

```

```
# Scan for Modems
mmcli --scan-modems

# List available Modems
mmcli --list-modems

# List SIM information
mmcli --sim 0

# List Modem Information
mmcli --modem 0
```

Listing 4.2: Modem Discovery

The most significant tool and library used for interacting with the modems are the ModemManager and libqmi.

The ModemManager provides an abstraction layer on top of the actual protocols utilized to directly interact with the modem. This simplifies communication and provides a standardized approach for interacting with a modem. Next to being able to issue commands to establish and close connections with cellular networks, the ModemManager is essential for enabling the modem, authenticating with the SIM-card using a Personal identification number (PIN) and querying information on modem state and operation.

```
#!/bin/bash
# Modem reset
mmcli --modem 0 --reset

# Enable modem
mmcli --modem 0 --enable

# Authenticate SIM with PIN
mmcli --modem --sim=0 --pin=XXXX

# Monitor modem state
mmcli --modem 0 --monitor-state

# Scan for available networks
mmcli -m 0 --3gpp-scan
```

Listing 4.3: ModemManager - Modem Interaction

Further operations provided by the ModemManager are interaction with location related services, Short Message Service (SMS) operations, firmware queries, ability to send AT-commands and all commands surrounding the management of network connections and network bearers [9] [30] [27].

Next to the ModemManager, the library libqmi is crucial for communicating with QMI-enabled modems. This library is used to start and terminate cellular network

connections, as well as query information on various aspects of the network and modem itself. A user-space implementation utilizing libqmi is the program "qmicli" [26] [28].

```
#!/bin/bash
# Query Signal Info
qmicli --device=/dev/cdc-wdm1 --device-open-proxy \
--nas-get-signal-info

# Query Signal Strength
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--nas-get-signal-strength

# Query Data Format of Host System
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--get-expected-data-format

# Set Data Format of Host System to "raw-ip"
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--set-expected-data-format=raw-ip

# Query WWAN Interface Name
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--get-wwan-iface

# Query Capabilities of the Modem
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--dms-get-capabilities

# Query the Home Network
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--nas-get-home-network

# Query the Serving System
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--nas-get-serving-system

# Query SIM-Card Status
qmicli --device=/dev/cdc-wdm1 --device-open-proxy  \
--uim-get-card-status
```

Listing 4.4: libqmi - Modem Interaction

There exist various further qmicli commands, that aid in information retrieval and troubleshooting modem operation, however this is out of scope of this thesis [25].

4.3 5G Cellular Connectivity

This section demonstrates the connection-establishment process with a 5G Campus Network slice.

The first step involves resetting the modem, ModemManager and NetworkManager, in order to start the required hardware and tools in an orderly state. Moreover, it is recommended to initially disconnect the modem from the power supply and then reconnect it, to cold start the modem and perform a hardware reset. Furthermore, for troubleshooting purposes, the ModemManager and NetworkManager are started with a more detailed log-level to increase visibility of potential errors. The software applicable steps are depicted in the following code excerpt [96].

```
#!/bin/bash
# Reset the Modem
mmcli --modem 0 --reset

# Stop ModemManger and NetworkManager
systemctl stop ModemManager
systemctl stop NetworkManager

# Start ModemManger and NetworkManager with DEBUG log-level
ModemManager --log-level=DEBUG &> /dev/null &
NetworkManager --log-level=DEBUG &

# Wait for ModemManger and NetworkManager to Initialize
sleep 30

# Check NetworkManager Connections
nmcli device status
```

Listing 4.5: Initialization for Connection Establishment

Subsequently, once the ModemManager and NetworkManager have been initialized correctly, the connection to the cellular network is established. At first, the modem is enabled, in order to send commands and interact with the modem. Thereafter, the name of the WWAN-interface is queried. Due to the properties of the 5G Campus Network, the "raw-ip" mode is set for the WWAN-interface. Consequently, the connection is established using the desired Access Point Name (APN). Each APN offered by the Campus Network represents a disparate network slice. Thenceforth, an IP-address is assigned, by requesting the configuration from a responsible DHCP server. The implementation of this process is illustrated in the following code excerpt [95].

```
#!/bin/bash
# Enable the Modem
mmcli --modem 0 --enable

# Query WWAN Interface Name
qmicli --device=/dev/cdc-wdm1 --device-open-proxy --get-wwan-iface

# Stop WWAN Interface
ip link set dev wwan0 down

# Query Data Format of Host System
qmicli --device=/dev/cdc-wdm1 --device-open-proxy --get-expected-
    data-format

# Set Data Format of Host System to "raw-ip"
qmicli --device=/dev/cdc-wdm1 --device-open-proxy --set-expected-
    data-format=raw-ip

# Start WWAN Interface
ip link set dev wwan0 up

# Connect to 5G Network Slice "jumpiot" using IPv4
qmicli --device=/dev/cdc-wdm1 --device-open-proxy --wds-start-
    network="ip-type=4,apn=jumpiot" --client-no-release-cid

# Request an IP-address from Networks's DHCP Server
udhcpc -q -f -n -i wwan0

# Check WWAN Interface for correct IP-assignment
ifconfig wwan0

# Check NetworkManager Connections
nmcli device status

# Test Layer 3 Connectivity
ping -I wwan0 -c 5 <IP-address/Domain>
```

Listing 4.6: Connection Establishment with 5G Campusnetwork

There exist various means of monitoring the network connection, radio interface and network interface. These indicators aid in judging the quality of the network connection and reveal potential errors. The most significant means of monitoring these aspects are depicted in the excerpt below [25] [27].

```
#!/bin/bash
# Monitor Modem State
mmcli --modem 0 -w

# Query WWAN Interface Statistics (every 5s)
watch -n5 ifconfig wwan0

# Query Signal Info (every 5s)
watch -n5 qmicli --device=/dev/cdc-wdm1 --device-open-proxy --nas-
    get-signal-info

# Query Signal Strength (every 5s)
watch -n5 qmicli --device=/dev/cdc-wdm1 --device-open-proxy --nas-
    get-signal-strength
```

Listing 4.7: Interface and Network Monitoring

Once the measurements have been conducted successfully, the connection is terminated [25].

```
#!/bin/bash

# Terminate Network Connection
qmicli --device=/dev/cdc-wdm1 --device-open-proxy --wds-stop-
    network=<HANDLE> --client-cid=<CID>

# Check NetworkManager Connections
nmcli device status
```

Listing 4.8: Connection Termination with 5G Campusnetwork

Troubleshooting is conducted using the log entries created by the ModemManager and NetworkManager. As the log-level was increased to "DEBUG", extensive information on various aspects regarding the modem state and network connection is recorded [96].

```
#!/bin/bash
grep -E 'ModemManager|NetworkManager|systemd|dbus-daemon|dhclient'
    /var/log/syslog | less
```

Listing 4.9: Troubleshooting using Syslog

4.4 TIG Stack

This section presents the implementation of the TIG-stack utilized for QoS-measurements and Network Performance Monitoring (NPM) in 5G Campus Networks. Firstly, the setup and configuration of the software components is covered. Subsequently, the deployment of these components using the containerization platform Docker is demonstrated. At last, a PoC depicts the operability of this implementation in a non-cellular network environment.

4.4.1 Configuration

This segment covers the configuration of the TIG-stack components, addressing each component individually, due to the application specific configuration. At first the composition of the Telegraf configuration is covered. Thereafter, the setup of the DBMS InfluxDB is examined. At last, the configuration of Grafana is addressed, including the integration of data sources and design of the dashboards for data visualization.

Telegraf

The component Telegraf is central to the collection and aggregation of all measurement data from the measurement clients. The main concern of the setup of this component is the Telegraf configuration file, located under the path `/etc/telegraf/telegraf.conf` on GNU/Linux OSs.

This configuration file comprises all the plugins and OS commands with their according parameters, utilized in gathering the performance metrics. Furthermore, this file specifies the desired remote location and database, where measurements are stored. In addition, the configuration specifies means of authentication with the remote database, using the authentication method of choice. The authentication employed in this configuration, uses "HTTP basic access authentication" [24].

The Telegraf component interacts with cellular interface via the host system to gather measurments, as well as with the database InfluxDB, that receives the measurements via a REST API.

The latency measurements using the protocol ICMP are conducted with the Telegraf plugin "inputs.ping" [48].

```
[[inputs.ping]]
urls = ["google.com", "reddit.com", "twitter.com",
        "www.dn.th-koeln.de", "th-koeln.de", "94.130.106.172",
        "1.1.1.1", "8.8.8.8", "he.de"]
count = 4
ping_interval = 1.0
timeout = 10.0
```

Listing 4.10: Ping Plugin Configuration

This Telegraf plugin conducts latency measurements to various destinations, located in the array named "urls". Multiple different destinations are chosen, in order to build a broader data set. The parameter "count" determines the number of ICMP-requests sent, which are set to four, to gather several measurements per cycle. The parameter "ping_interval" determines how many seconds pass between requests, and the variable "timeout" determines the amount of seconds to wait before the measurement is regarded as failed.

The next plugin utilized in the measurements, determines the latency of DNS requests. DNS is essential for mapping domains names to IP-addresses. Hence, DNS requests are executed whenever a service makes a request to a remote location, specified with a domain name, that is not specifically named in the OS's hosts file [49].

```
[[inputs.dns_query]]
  servers = ["8.8.8.8"]
  domains = ["google.com", "twitter.com", "th-koeln.de",
             "netzmessungen.de", "telekom.de", "he.de"]
```

Listing 4.11: DNS Plugin Configuration

The configuration of the DNS plugin specifies the DNS server utilized for the queries, using the variable "servers". The DNS server chosen for the requests is a publicly available DNS server operated by Google LLC. The domains to be queried are listed in the array "domains". Multiple domains are listed in this array, in order to gather data on several disparate domains.

The collection of jitter measurements is conducted using the tool MTR. This tool has the ability to determine jitter to a particular destination, as well as output the average and worst jitter values from these measurements [11].

```
# Jitter Monitoring  - TH Network
[[inputs.exec]]
  commands=["mtr --report -c 5 srv2.netzmessungen.de \
             -o JMX --json"]
  interval = "13m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "jitterTh"
```

Listing 4.12: Jitter Exec-Plugin Configuration

The "current Jitter" is specified using the output field "J", the "Jitter Mean/Avg." is specified using the field "M" and "Worst Jitter" is specified using the field "X".

Since this tool is executed using the OS, the "inputs.exec" plugin is utilized for this measurement. The frequency at which this measurement is executed, is specified using the variable "interval", in this case 13 minutes. Furthermore, the field "timeout" determines the time limit before a measurement is regarded as failed.

Throughput measurements are conducted using two utilities installed on the OS of the measurement host. The first tool is the CLI "speedtest-cli" of "speedtest.net" operated by Ookla. The second tool employed for throughput measurments is "iperf3" [59] [72].

The utility speedtest-cli conducts measurements using TCP, with HTTP as a backup option, in case measurements using TCP are restricted. The tool measures both upload and download throughput, by dynamically allocating the amount of threads necessary to produce a more accurate result [79].

```
[[inputs.exec]]
  commands = ["speedtest --secure --json"]
  interval = "5m"
  timeout = "2m"
  name_suffix = "_speedtest"
  data_format = "json"

  tag_keys = [
    "client_isp",
    "client_ip",
    "server_id",
    "server_name",
    "server_country",
    "server_cc"
  ]
```

Listing 4.13: Throughput Speedtest Exec-Plugin Configuration

The speedtest configuration illustrates the user space call of the package "speedtest-cli", using the parameter "–secure" to minimize potential blockage of throughput tests by the network operator. This measurement is conducted using the plugin "inputs.exec". The "interval" and "timeout" parameters have the same definition as in the previous configuration of the jitter measurements. Further parameters specify the format and naming scheme of the data in the database.

Furthermore, the tool "iperf3" is utilized in conducting throughput measurements. This utility is more customizable than "speedtest-cli", and offers the ability to execute measurements using TCP and UDP among others [59].

```
# iPerf3 Monitoring with 4 Threads TCP - TH Network
[[inputs.exec]]
  commands=["iperf3 -c srv2.netzmessungen.de -p5201 -b 1000M -P 4
   --json"]
  interval = "9m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "iperf3th"
```

Listing 4.14: Throughput Iperf3 TCP Exec-Plugin Configuration

The configuration above depicts a TCP throughput measurement to "srv2.netzmessungen.de" using the CLI call to "iperf3" with the client option ("-c") and port 5201 ("-p"). The default protocol utilized by "iperf3" is TCP, therefore the protocol is not explicitly stated in the command. The parameter "-b" indicates the target bitrate, in this case 1Gbps. The bitrate was set to 1Gbps, as the initial throughput measurements yielded results below this value. Adjusting the thread count in throughput measurements can aid in utilizing the maximum channel bandwidth to achieve more accurate results, therefore this option is increased to four threads, using the parameter "-P".

```
# iPerf3 Monitoring with 4 Threads UDP - Hetzner
[[inputs.exec]]
  commands=["iperf3 -c srv1.netzmessungen.de -p5201 -u -b 1000M -P
   4 --json"]
  interval = "23m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "iperf3hetznerudp"
```

Listing 4.15: Throughput Iperf3 UDP Exec-Plugin Configuration

The configuration above depicts a UDP throughput measurement to "srv1.netzmessungen.de" using the CLI call to "iperf3" using the plugin "inputs.exec". The protocol UDP has to be explicitly stated in the command using the "-u" option. The other parameters are equal to the TCP measurements using "iperf3".

The appendix includes a full sample configuration of one of the measurement clients, utilizing multiple measurement end-points for each metric.

InfluxDB

The configuration of the DBMS, InfluxDB, comprises the activation of the REST interface and authentication thereof. Furthermore, a user for the Telegraf agents, as well as an admin account for administration of the DBMS is created. In addition, the creation of the required databases is performed.

```
# InfluxDB Configuration - Section [http]
[http]
    # Enable REST Interface
    enabled = true

    # Enable Authentication
    auth-enabled = true

    # Port to bind to
    bind-address = ":8086"

    # Enable Logging
    log-enabled = true

    # Parameters for Debugging and Monitoring
    write-tracing = false
    pprof-enabled=true
    pprof-auth-enabled=true
    debug-pprof-enabled=false
    ping-auth-enabled = true
```

Listing 4.16: InfluxDB Configuration - HTTP Endpoint

The excerpt above depicts the configuration of InfluxDB's HTTP-endpoint. At first, the HTTP end-point is activated, along with HTTP-authentication. Thereafter, the port for the end-point is defined and logging enabled, in order capture information on all requests to the end-point. At last, several options in regard to logging and monitoring are set. These parameters aid in the debugging process as well as authenticate the retrieval of that information. Further, options are disabled to limit the amount collected debugging data [43].

```
# User Creation
$ influx
influx> CREATE USER admin WITH PASSWORD 'XXXXX'
        WITH ALL PRIVILEGES;
influx> CREATE USER telegraf WITH PASSWORD 'XXXXX';

# Test User Creation
influx> SHOW USERS;
user     admin
----     -----
admin    true
telegraf false
```

Listing 4.17: InfluxDB Configuration - User Creation

The code fragment above, illustrates the creation of two users. The "admin" user is granted all privileges for administration purposes. The "telegraf" user is given permissions on an on-demand basis, per database that is utilized for the measurements [42].

```
# Database Creation
$ influx
influx> CREATE DATABASE 5gmessung1;
influx> GRANT ALL ON 5gmessung1 TO telegraf;

# Test Database Creation
influx> SHOW DATABASES;
name: databases
name
----
_internal
5gmessung1

influx> USE 5gmessung1;
Using database 5gmessung1
influx> SHOW MEASUREMENTS;
name: measurements
name
----
dns_query
exec_speedtest
iperf3godaddy
iperf3godaddyudp
....
ping
```

Listing 4.18: InfluxDB Configuration - Database Creation and Permissions

The creation of databases is illustrated above. At first the database "5gmessung1" is setup and the user "telegraf" is granted privileges for this database. Subsequently, several measurements from the individual Telegraf plugins are stored in the database by a measurement client. These are displayed by selecting the database and displaying the measurements.

Grafana

The visualization interface, Grafana, requires the configuration of two main components in order to visualize measurement data. The first part entails the configuration of a data source. Various types of data sources are already integrated into Grafana, including the DBMS InfluxDB. The creation of a new InfluxDB data source in Grafana, comprises the specification of a name, HTTP Uniform Resource Locator (URL) of the server hosting the database, authentication mechanism with credentials and lastly the InfluxDB database name, user and password.

Parameter	**Value**
Name	TIGClient1-jumpiot
URL	http://srv1.netzmessungen.de:8086
HTTP Auth (User)	telegraf
HTTP Auth (Password)	XXXXXXX
InfluxDB (DB Name)	5gmessung1
InfluxDB (User)	telegraf
InfluxDB (Password)	XXXXXXX

Table 4.2: Grafana Data Source - Sample Configuration

Subsequently, the data source is utilized by a dashboard. Dashboards consist of numerous panels that are individually customized to fit the type of data illustrated. Once a dashboard has been created with all the panels required to fulfill one's objective, the entire dashboard can be exported and imported for further reuse with other data sources. The following two figures demonstrate examples of different type of panels and how panels are grouped together in rows.

Figure 4.2: Grafana - "Stat" panels illustrating maximum latency to multiple different locations.

Figure 4.3: Grafana - "Timeseries" panel depicting throughput (upload) over time to a server hosted with Host Europe.

The data displayed in the panels is provided by one of the data sources configured in Grafana. In order to query and aggregate the data correctly, the query language InfluxQL is used. Two sample queries for gathering measurement data are displayed below.

```
# InfluxQL Query
# Query Figure 4.2 - Stat Panel
SELECT max("maximum_response_ms") FROM "ping"
    WHERE ("url" = 'hosteurope.de') AND $timeFilter
    GROUP BY time($__interval) fill(null)

# Query Figure 4.3 - Timeseries Panel
SELECT "sum_sent_bits_per_second" FROM "iperf3godaddy"
    WHERE $timeFilter
```

Listing 4.19: Grafana Configuration - InfluxQL Query Examples

The queries in the figure above depict the retrieval of data from a specific "MEASUREMENT" of an InfluxDB database. The first query retrieves the ICMP latency measurements from the "MEASUREMENT" "ping" of the "inputs.ping" Telegraf plugin. The "WHERE" clause is used to specify the measurements in a certain time interval to a specific domain, in this case "hosteurope.de". The aggregator utilized in this query, "max()", retrieves the maximum value from the time range specified in the global variable *&timeFilter* configured in the dashboard. The second

query retrieves the throughput in upload direction of a certain time range, using the metric "sum_sent_bits_per_second" from the "MEASUREMENT" "iperf3godaddy", collected with the tool "iperf3".

4.4.2 Deployment

The deployment of the TIG-stack is achieved using on the containerization platform Docker. The measurement clients run single Docker instance of Telegraf with a specific Telegraf configuration, that specifies the metrics and interval at which measurements are collected.

Deploying the Telegraf container using Docker requires the adapted configuration file and a working Docker installation on the measurement host. Deployment of the Telegraf container is depicted in the subsequent excerpt.

```
# Dockerfile
FROM telegraf:1.12

# Install Dependencies of Measurement Tools
RUN apt-get update && apt-get install -y --no-install-recommends \
    dnsutils speedtest-cli iperf3 iputils-ping hping3 mtr-tiny \
    traceroute nano && rm -rf /var/lib/apt/lists/*

# End Dockerfile

# CLI Commands to Build and Run Docker Container
# Build Telegraf Docker Image
docker build -t telegraf-tigclient - < telegraf-tigclient.docker

# Run Telegraf Docker Image
docker run --name telegraf --rm -v \
    $PWD/telegraf.conf:/etc/telegraf/telegraf.conf \
    telegraf-tigclient
```

Listing 4.20: Docker Deployment - Telegraf

The Dockerfile specifies the base image, which is the official Telegraf Docker image published on Docker Hub. In order to run the plugins and measurement tools correctly, all missing dependencies are installed. Thereafter, the Docker image is built and run from the CLI of the measurement host. The commands are executed from the local directory that contains the custom Telegraf configuration and Dockerfile, in order for the container to mount the configuration file correctly.

The InfluxDB and Grafana instances are hosted together on a server at Hetzner Online GmbH. In order to operate both platforms using a Docker environment, a container orchestration tool called Docker Compose is utilized for this purpose [38].

```
# Docker Compose - Container Definition of InfluxDB and Grafana
version: '3.9'

networks:
    tig-net:
        driver: bridge

volumes:
    tig-data:

services:
    influxdb:
        image: influxdb:1.8.6
        container_name: influxdb
        ports:
            - 8086:8086
        environment:
            INFLUXDB_HTTP_AUTH_ENABLED: "true"
            INFLUXDB_DB: "telegraf"
            INFLUXDB_ADMIN_USER: "admin"
            INFLUXDB_ADMIN_PASSWORD: "XXXXX"
        networks:
            - tig-net
        volumes:
            - tig-data:/var/lib/influxdb

    grafana:
        image: grafana/grafana:7.5.9
        container_name: grafana
        ports:
            - 3000:3000
        environment:
            GF_SECURITY_ADMIN_USER: admin
            GF_SECURITY_ADMIN_PASSWORD: XXXXX
        networks:
            - tig-net
        volumes:
            - tig-data:/var/lib/grafana
        restart: always

# Test opreation of InfluxDB and Grafana
curl -G "http://localhost:8086/query?pretty=true" \
     -u admin:XXXXX --data-urlencode "q=show databases"
curl -I "http://localhost:3000/login"
```

Listing 4.21: Docker Deployment - InfluxDB and Grafana [56]

The Docker Compose file exhibits the definition and configuration of the InfluxDB and Grafana containers. The containers both utilize the official application images as their base image. In order for them to communicate with each other, a Docker network named "tig-net" is created. The respective ports that expose the applications are defined using the parameter "ports". For the purpose of providing persistent storage, a Docker volumes is defined, that stores the application related data and configurations on the host system. Initial configuration of the applications is conducted using environment variables, that allow for the creation of an application user along with a password. Furthermore, an initial database is created and the REST interface and authentication thereof is enabled. The operation of the containers is verified using "curl" and making calls to the respective HTTP end-point. Further configuration of the applications is conducted by interacting with the CLI of the containers and by accessing the configuration files, using the location of the Docker volume on the host's file system.

Proof of Concept

The operation of the TIG-stack is verified by deploying the software on a host system in a non-cellular environment. The same host hardware and operating system are utilized in this setup, to replicate the production environment as closely as possible, however without the cellular interface for 5G-connectivity.

The host system is instead interconnected using Gigabit Ethernet (IEEE 802.3 1000BASE-T), offering speeds of up to 1000Mbit/s. The system is deployed in the university network at the DN.Lab. The Telegraf agent is deployed with a configuration file equivalent to the configuration files deployed on the measurement clients in the 5G Campus Network environment. The verification measurement is conducted for a period of one week [100].

The collected metrics are adequate and reflect the results when collecting measurements using the same tools and parameters, without the utilization of the TIG-stack as the measurement platform. Furthermore, the measurements in the Gigabit Ethernet environment did not lead to a processing bottleneck of the host system, that might affect the cellular results collected in the 5G Campus Network. The throughput measurements using "iperf3" in the download direction depict values between 268Mbit/s and 931Mbit/s, with the majority of those values centering around the 920Mbit/s mark. The measurements using "speedtest" range from 67Mbit/s to 409Mbit/s, with the vast majority of data points accumulating at the 409Mbit/s mark. The throughput measurements using "iperf3" in the upload direction range from 271Mbit/s and 935Mbit/s, with the majority amassing at the 930Mbit/s mark. The "speedtest" throughput results in the upload direction yielded substantially lower results, ranging from 9.44Mbit/s to 152Mbit/s, with the vast majority centering around 152Mbit/s. The discrepancy of the values yielded using "iperf3" in

comparison to "speedtest", stems most likely from the dynamic allocation of the measurement end-points in comparison to the self-hosted measurement end-points used with "iperf3". This dynamic allocation leads to the utilization of different measurement end-points, with disparate capacities, load and geographic locations. Furthermore, "speedtest" employs HTTP as fallback mechanism in case TCP is restricted along the path to the end-points. This leads to different measurement results, when comparing both tools.

The latency results collected using the "inputs.ping" plugin of Telegraf produced results ranging from 6.17ms to 26.0ms. The results vary due to the different locations of the end-points that tests are conducted against. For example, the end-point at GoDaddy is hosted in Strasbourg, France, while the instance from Hetzner is located in Nuremburg, Germany.

The jitter results collected using MTR, range from 0.016ms to 0.186ms. These results indicate excellent performance for VoIP, video calls or gaming, as the jitter is almost negligible in human interactive applications. The DNS-query latency gathered with the Telegraf plugin "inputs.dns_query", ranges from 6.56ms to 20ms, with the vast majority of results lying between 6.6ms to 14.0ms.

The results of the throughput measurements are illustrated below. The measurements of latency, jitter and DNS-query latency are listed in Appendix C.

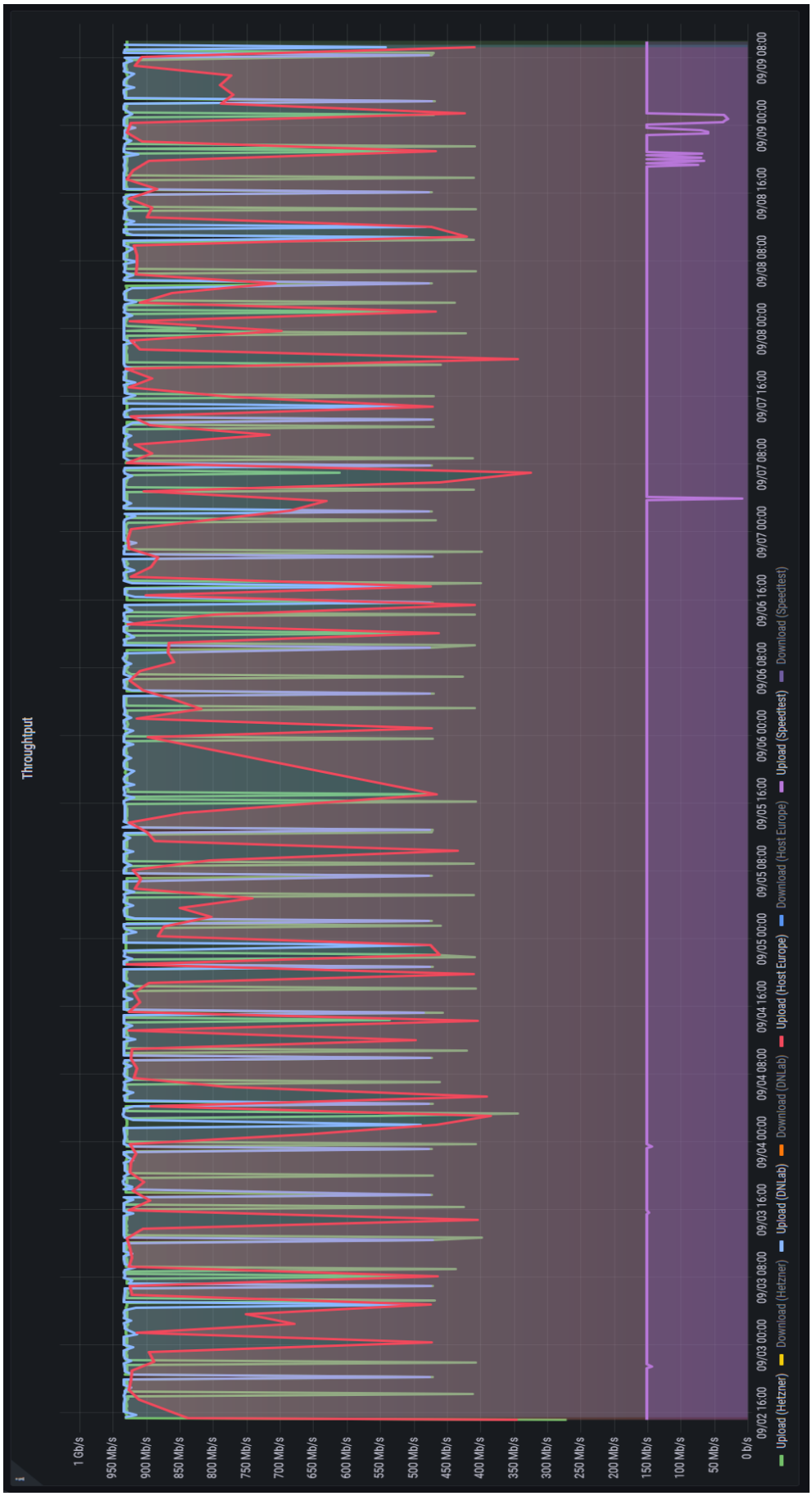


Figure 4.4: PoC - Throughput (Upload)

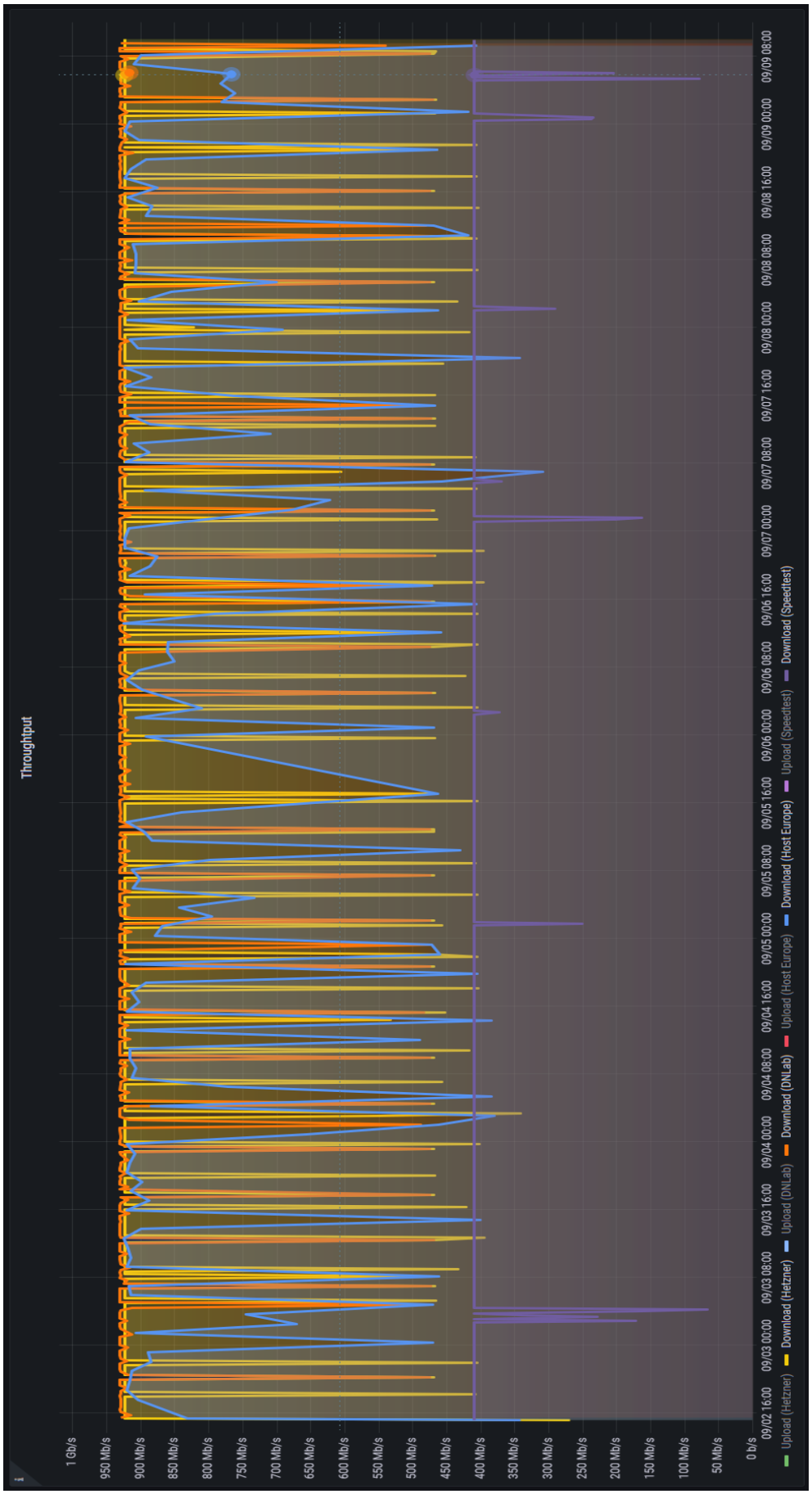


Figure 4.5: PoC - Throughput (Download)

4.5 Testing Locations

The locations utilized for the measurement are located at the campus Deutz of the Cologne University of Applied Sciences. There are three premises where measurement clients are deployed, namely the Makerspace, the Gateway Gründungsservice and the Cologne Cobots Lab. All three premises are in the vicinity of the 5G New Radio (5G NR) equipment, required for connectivity with the 5G Co:CreationLab's 5G Campus Network. In total there are two antennas including a Remote Radio Unit (RRU). One antenna is installed in the premises of the Gateway Gründungsservice, the other is installed in the space of the Cologne Cobots Lab.

Figure 4.6: 5G Antenna at Gateway Gründungsservice (side view)

Figure 4.7 depicts the locations where the clients are deployed, as well as the installation of the antennas and RRUs. The subsequent figure, figure 4.8, depicts the cellular infrastructure of the Co:CreationLab at the campus Deutz and connection to the core network at Deutsche Telekom AG.

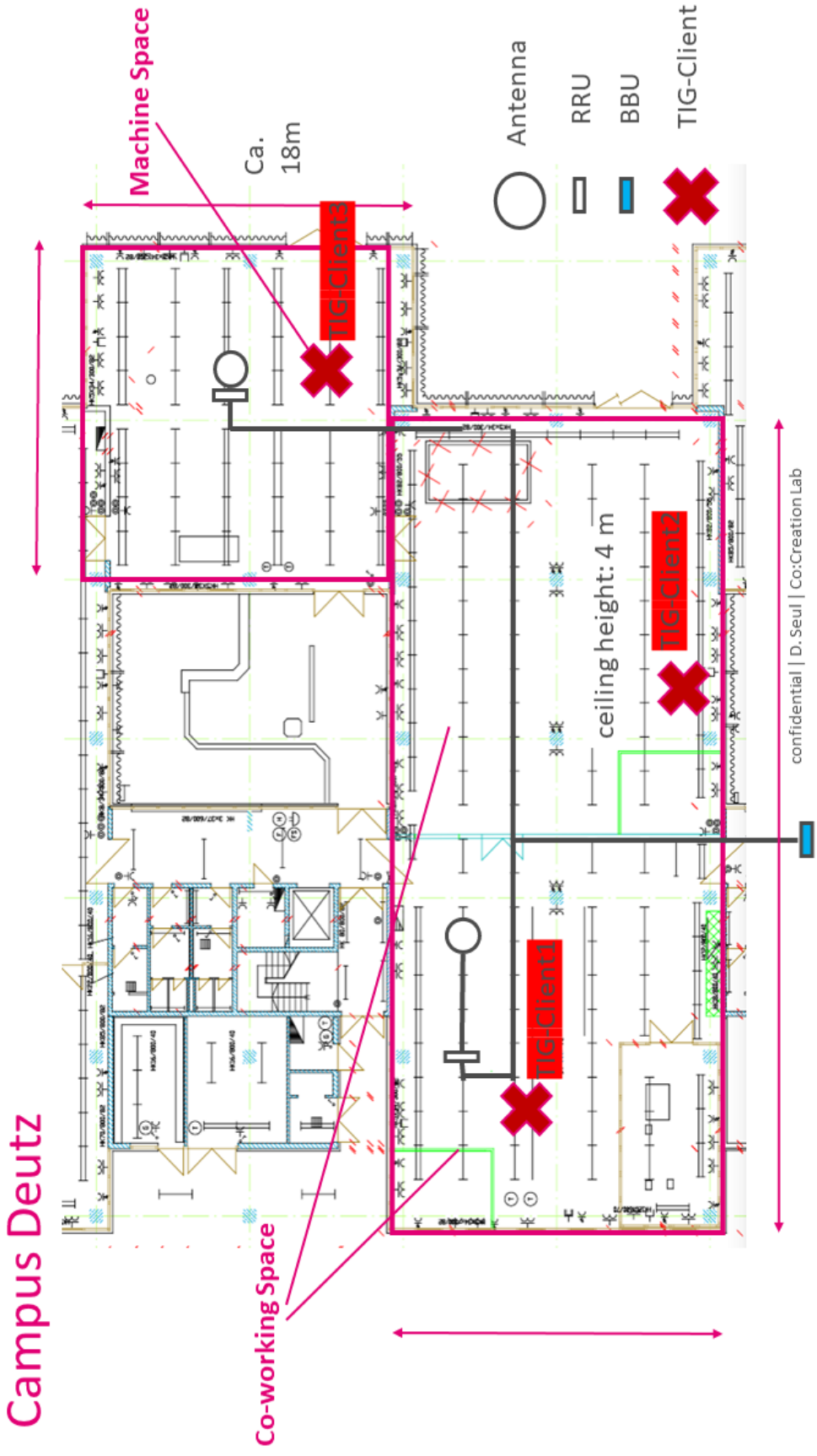


Figure 4.7: Location of Measurement Clients - Co:CreationLab Campus Deutz [91]

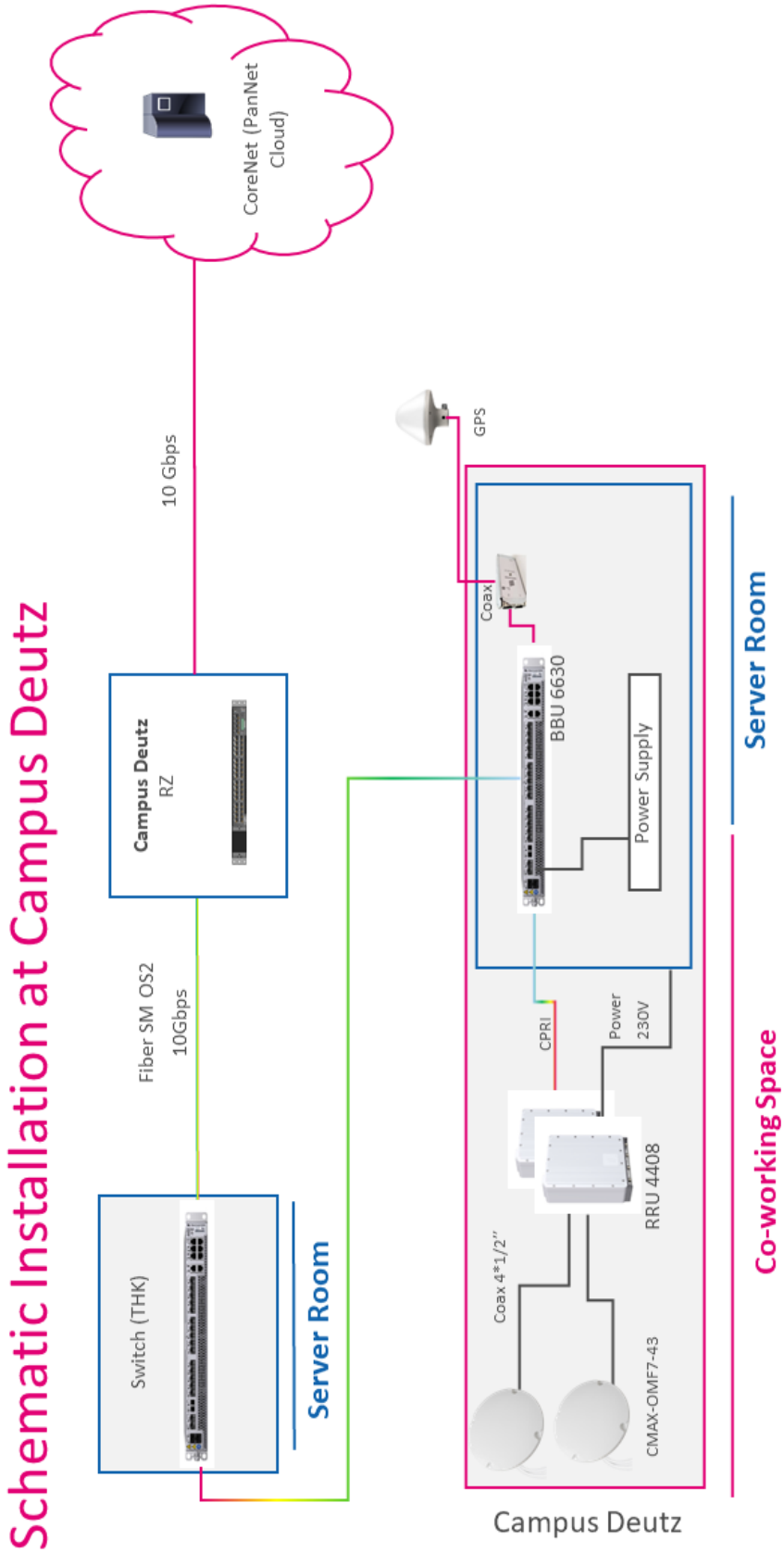


Figure 4.8: Schematic Installation - Co:CreationLab Campus Deutz [91]

Network Test Results and Evaluation

This chapter illustrates the final results of the measurements conducted on the network slices of the 5G Campus Network. Furthermore, the measurements are compared with the network's specification and assessed using the defined KPIs. Moreover, outages and availability are evaluated, as well as difficulties clarified, that arose during the process of conducting measurements.

5.1 Results of Network Tests

This section demonstrates all significant QoS-measurements on the disparate slices. The key performance aspects measured are throughput, latency, error rate, DNS-query latency and jitter using measurements conducted over varying time intervals.

5.1.1 Throughput

This section covers the throughput measurements conducted using "iperf3" and "speedtest". The metrics gathered are based on TCP throughput in the upload and download direction. There are three measurement end-points running an "iperf3"-server, namely a system at Hetzner, at the DNLab and at GoDaddy. Furthermore, "speedtest" conducts throughput measurements by dynamically allocating a measurement end-point.

The results depicted in Figure 5.1 illustrate the throughput measurements in the upload direction. These metrics were collected using the network slice "jumpiot" using TIGClient2 over a period of 10 days. The findings gathered using the "iperf3" utility vary between 3.31Mbit/s and 84.50Mbit/s, with the majority of the measurements focusing around 75Mbit/s. The tool "speedtest" achieved results up to 108Mbit/s, with the lowest measurement reaching 3.31Mbit/s. The majority of results are based around the 90Mbit/s mark.

The metrics shown in Figure 5.2 depict the throughput measurements in the download direction, collected using TIGClient2 on the network slice "jumpiot" over

a period of 10 days. The results gathered using the "iperf3" utility range from 2.98Mbit/s up to 76.8Mbit/s. The majority of data points is centered around the 70Mbit/s mark. The results gathered using "speedtest" range from 32Mbit/s to 409Mbit/s. The majority of results are centered around the 400Mbit/s mark.

The throughput measurements differ in terms of the symmetry of the results. The upload and download metrics collected using "iperf3" are comparatively symmetrical. In contrast, the results gathered using "speedtest" are asymmetrical, with download to upload ratio being approximately four to one.

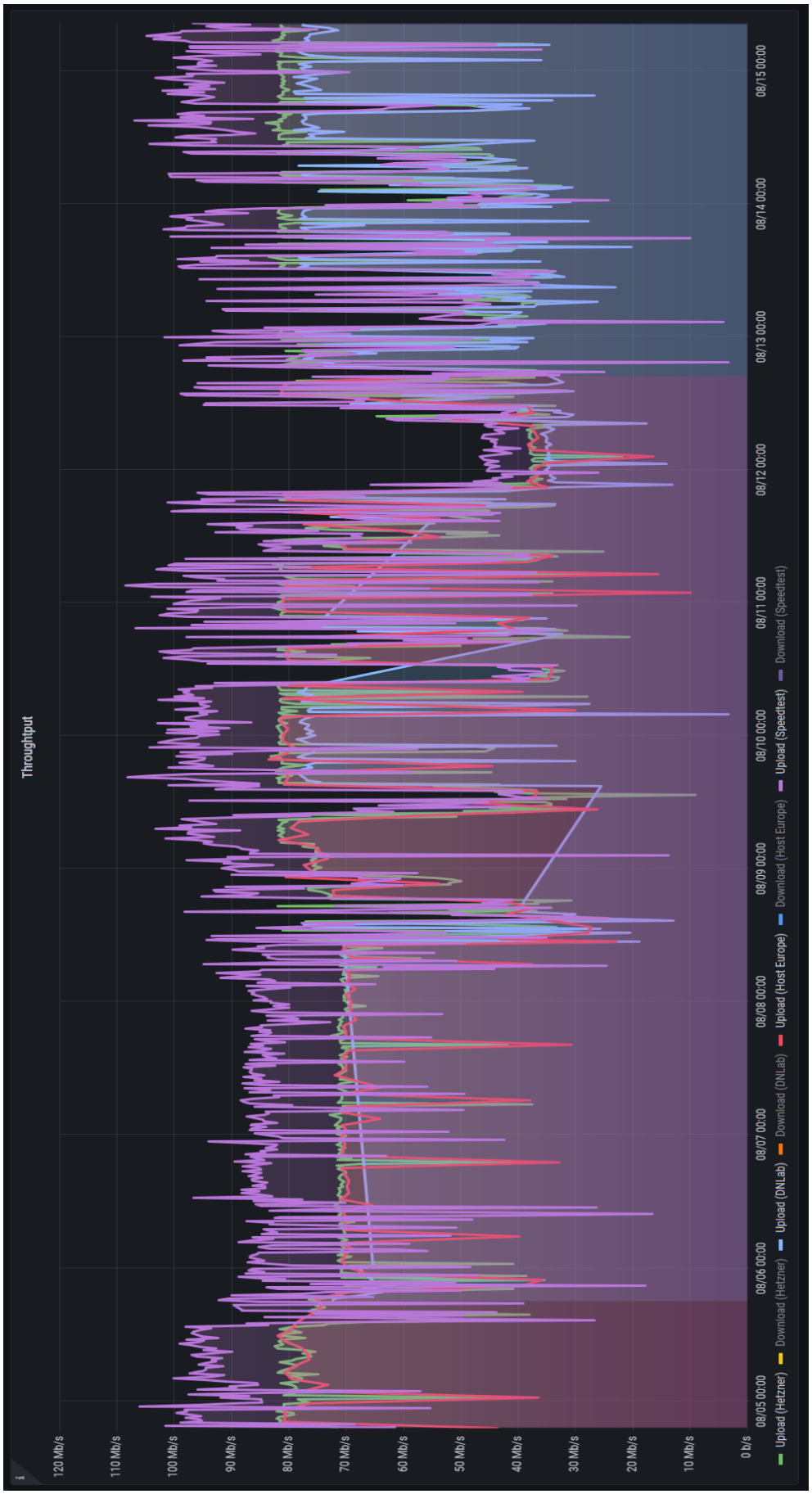


Figure 5.1: Throughput Upload (TCP) - Slice Jumpiot on TIGClient2

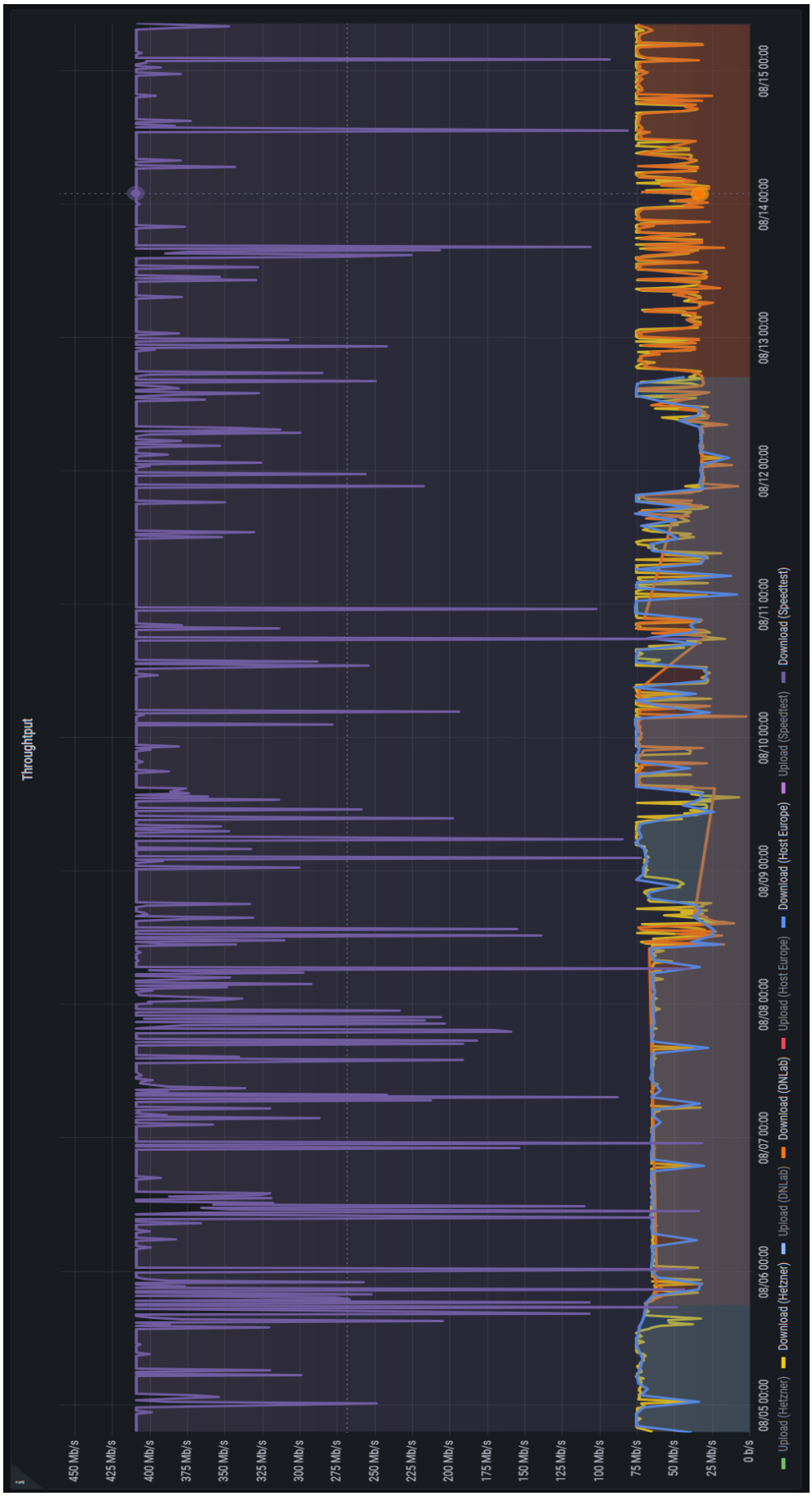


Figure 5.2: Throughput Download (TCP) - Slice Jumpiot on TIGClient2

5.1.2 Latency

This section covers the latency metrics, collected using the "inputs.ping" Telegraf plugin. These measurements depict the RTT to specific destination, thus the time taken for an ICMP Echo-request to receive the Echo-reply. The destination utilized for the measurements are the three measurement end-points at Hetzner, the DN.Lab and GoDaddy, as well as "google.com", "twitter.com", "reddit.com" and Cloudflare's "1.1.1.1".

The results depicted in Figure 5.3 show the latency measurements gathered using TIGClient2 on the network slice "jumpiot" over a period of 11 days. The results are limited to 150ms, since the overwhelming majority of data points lie below that value and including the spikes leads to an unclear depiction of the results. The full collection of results is depicted in Appendix E (Measurement Results) E.1. The vast majority of metrics lies below the 40ms mark. Furthermore, the figure depicts the latency differences when comparing the individual destinations. The latency measurements to the server at the DN.Lab mostly focused between 30ms and 40ms, while the other destinations depict the majority of measurements below the 30ms mark.

The metrics illustrated in Figure 5.4 depict the latency measurements collected using TIGClient1 on the slice "jumpmediaupload" measured over a period of two hours. Similarly to the previous measurement, the results are bound to a certain magnitude for clarity purposes, in this case 50ms. The complete depiction of all data points is referenced in Appendix E (Measurement Results) E.2. The majority of results lie below the 40ms mark, with a clear disparity regarding the distinct destinations. The measurements to the server at the DN.Lab produced most results between 34ms and 40ms. The outcome of measurements to the destinations "google.com" and the server at Hetzner are mainly between 23ms and 32ms. Lastly, the majority of results from "reddit.com", "twitter.com" and Cloudflare focus between 14ms and 26ms.

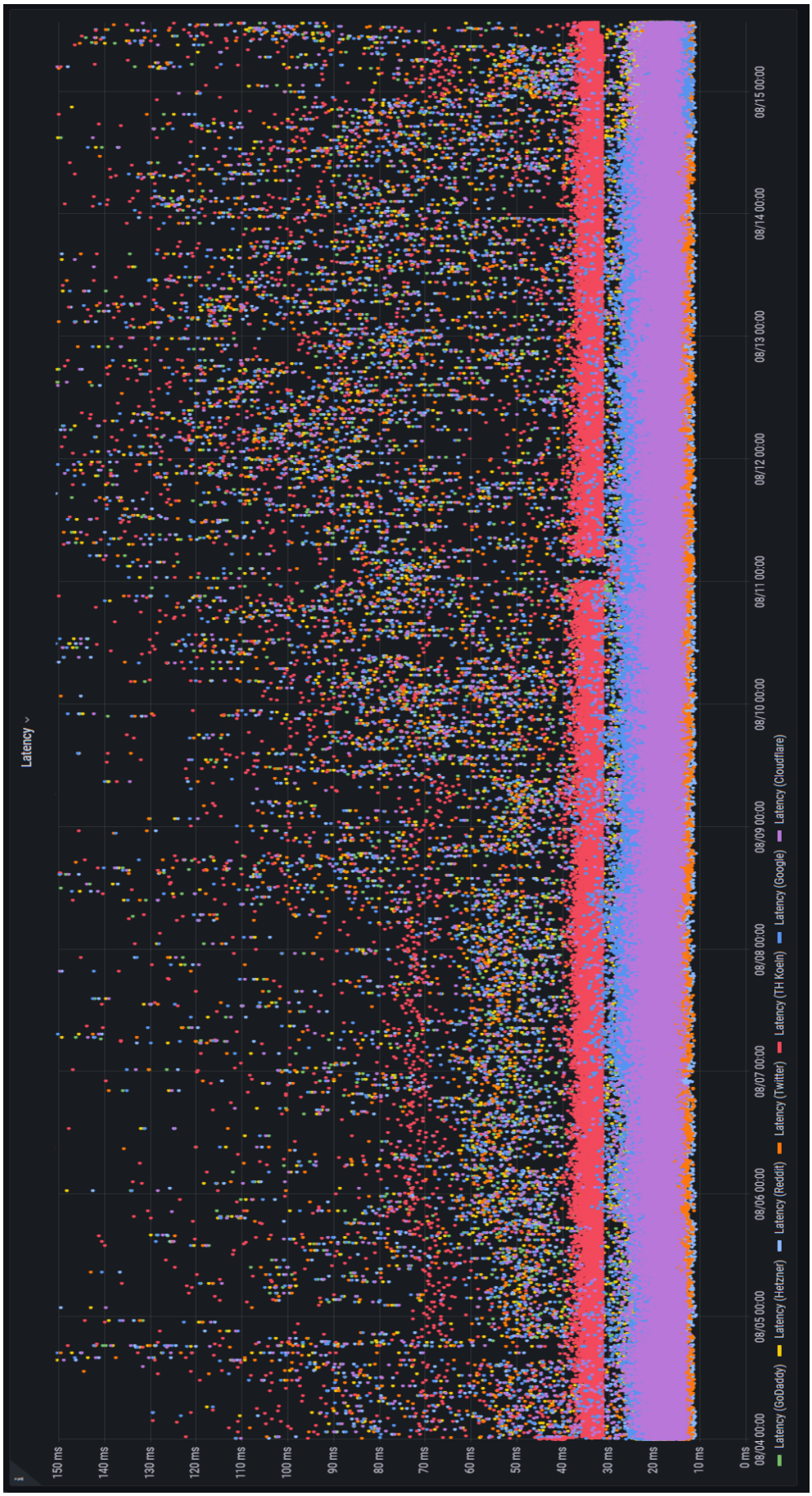


Figure 5.3: Latency (ICMP) up to 150ms - Slice Jumpiot on TIGClient2

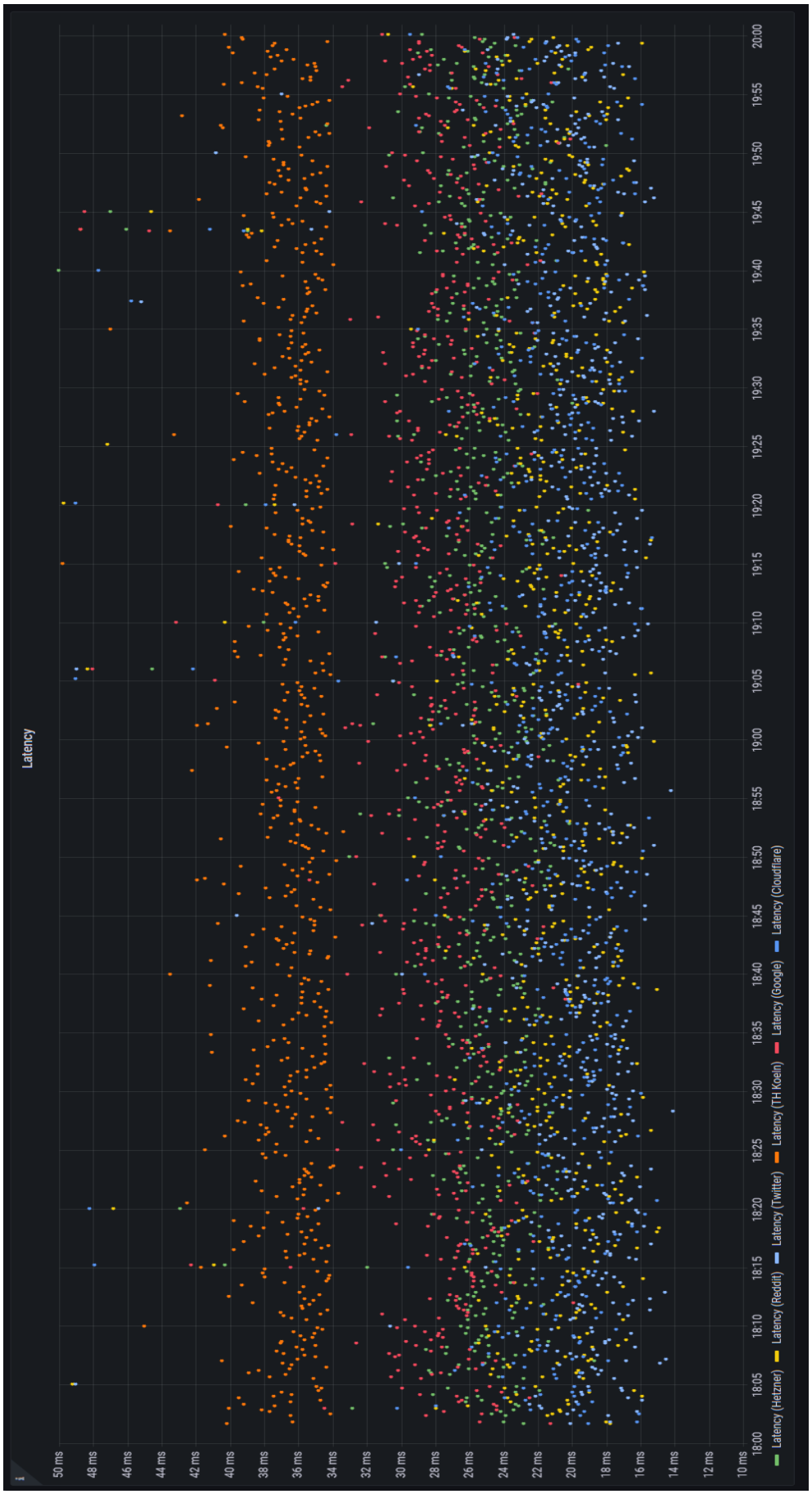


Figure 5.4: Latency (ICMP) up to 50ms - Slice Jumpmediaupload on TIGClient1

5.1.3 Error Rate

This section covers results regarding the error rate of the 5G Campus Network slices. Two aspects are measured, namely the number of TCP retransmissions on the transport layer (layer 4), as well as packet loss on the network layer (layer 3) of the OSI-model using the protocol ICMP. The utility employed for measuring TCP retransmissions is "iperf3". The end-points utilized for measuring TCP retransmissions are the server at Hetzner, GoDaddy and the system at the DN.Lab. The packet loss is measured using the Telegraf plugin "inputs.ping" and gathers metrics on the same destinations utilized in the previous segment on latency.

The results depicted in Figure 5.5 illustrate the TCP retransmissions collected using TIGClient2 on the slice "jumpiot" measured over a period of 11 days. The vast majority of TCP retransmissons are below five per measurement cycle. There are only few exceptions compared to the number of data points, however the number of retransmissons reach up to a magnitude of 344 during one measurement cycle.

The metrics presented in Figure 5.6 depict the percentage of packets lost during one measurement cycle with the plugin "inputs.ping". The measurements were conducted with TIGClient1 on the network slice "publicinternet" over a period of 80 minutes. The figure illustrates that given the network is available, packet loss is mostly zero and only in few cases 20% or above. However, due to the periods of unavailability, four time intervals of around five minutes depict 100% packet loss.

The results illustrated in Figure 5.7 present the percentage of packets lost during one measurement cycle. The metrics were collected using TIGClient2 on the slice "jumpiot" over a period of 11 days. The results demonstrate that high availability is given during the measurement period and only a thousandth of measurements experienced 10% packet loss or more.

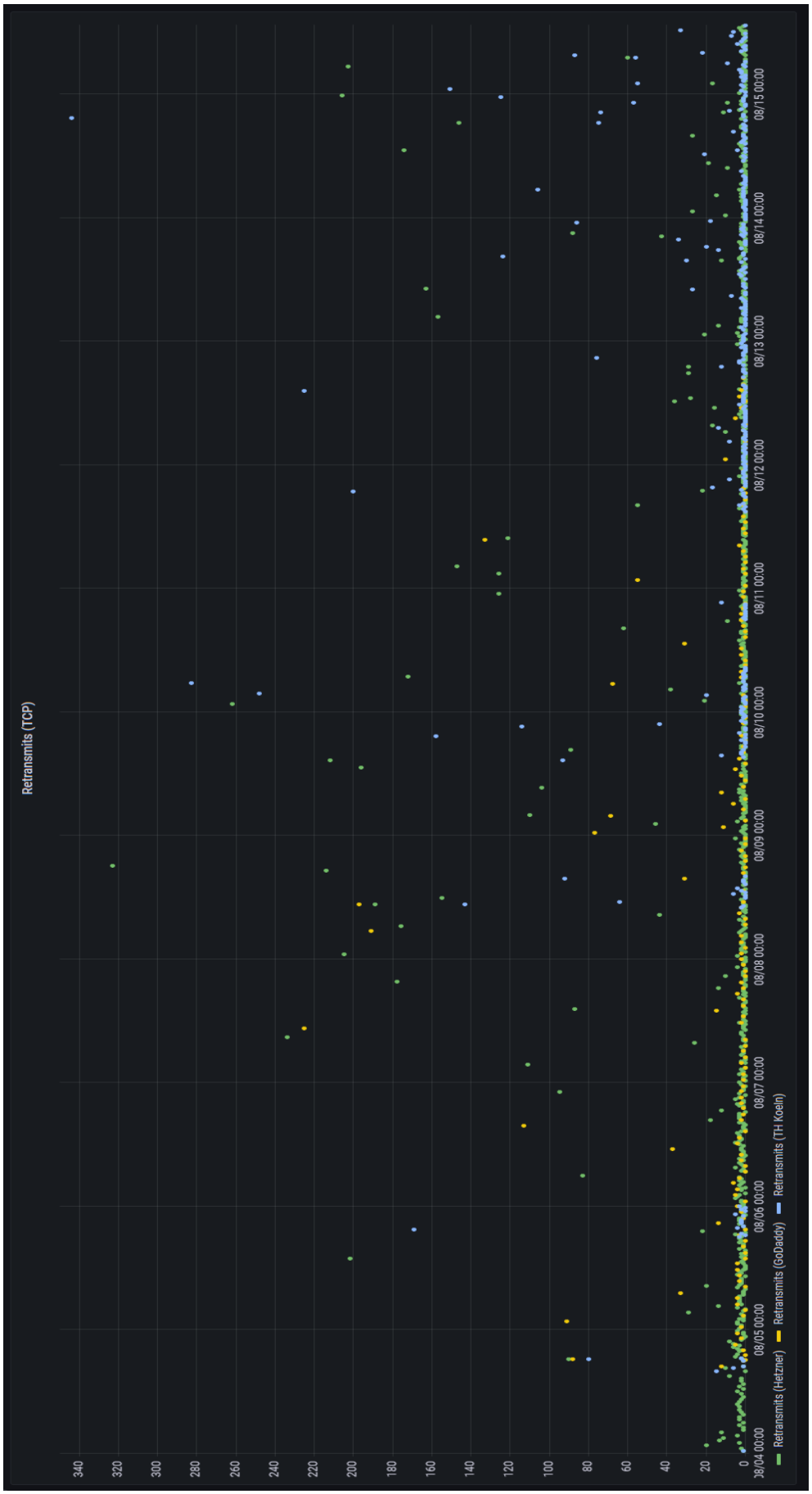


Figure 5.5: Retransmits (TCP) - Slice Jumpiot on TIGClient2

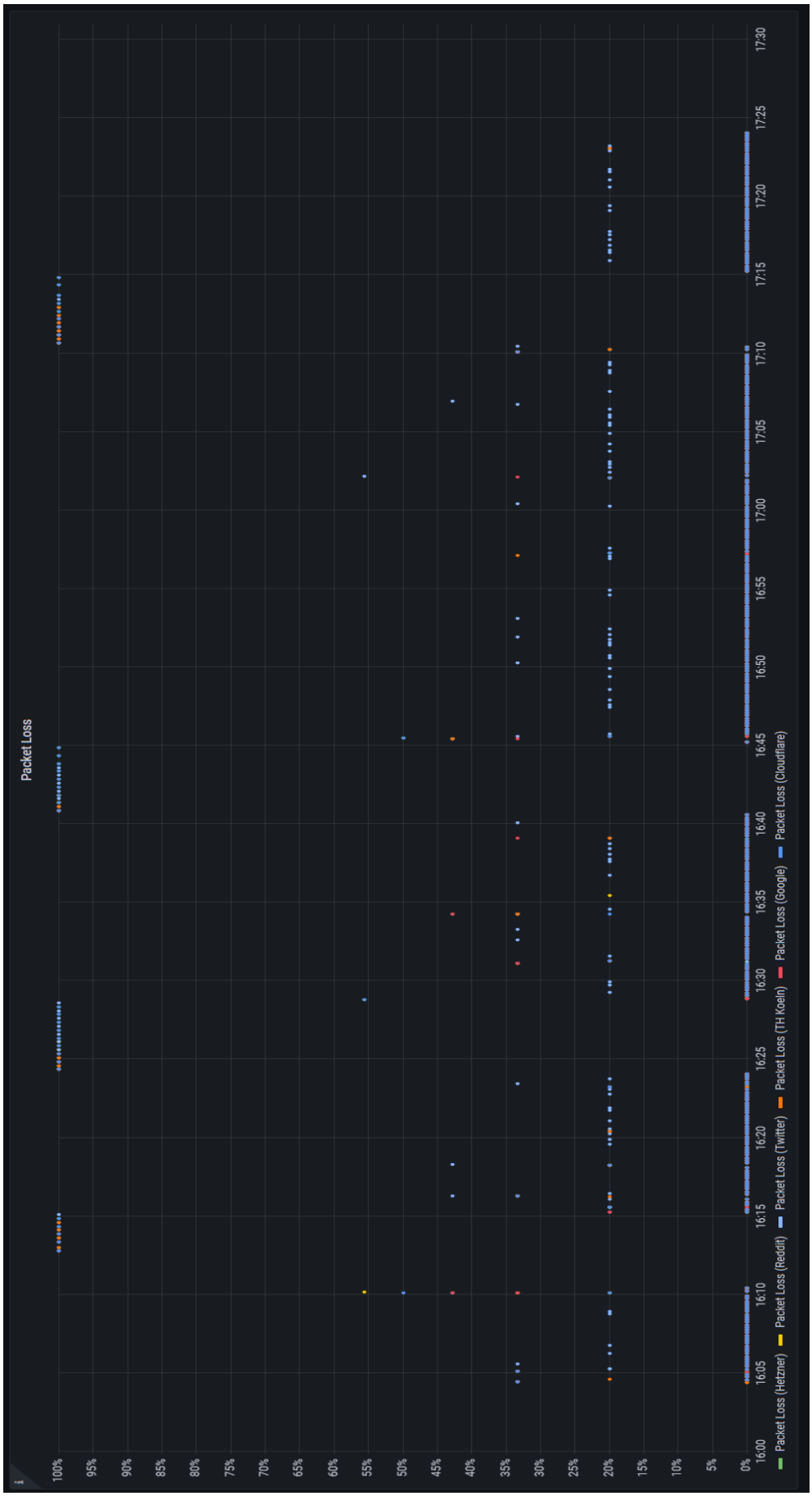


Figure 5.6: Packet Loss (ICMP) - Slice Publicinternet on TIGClient1

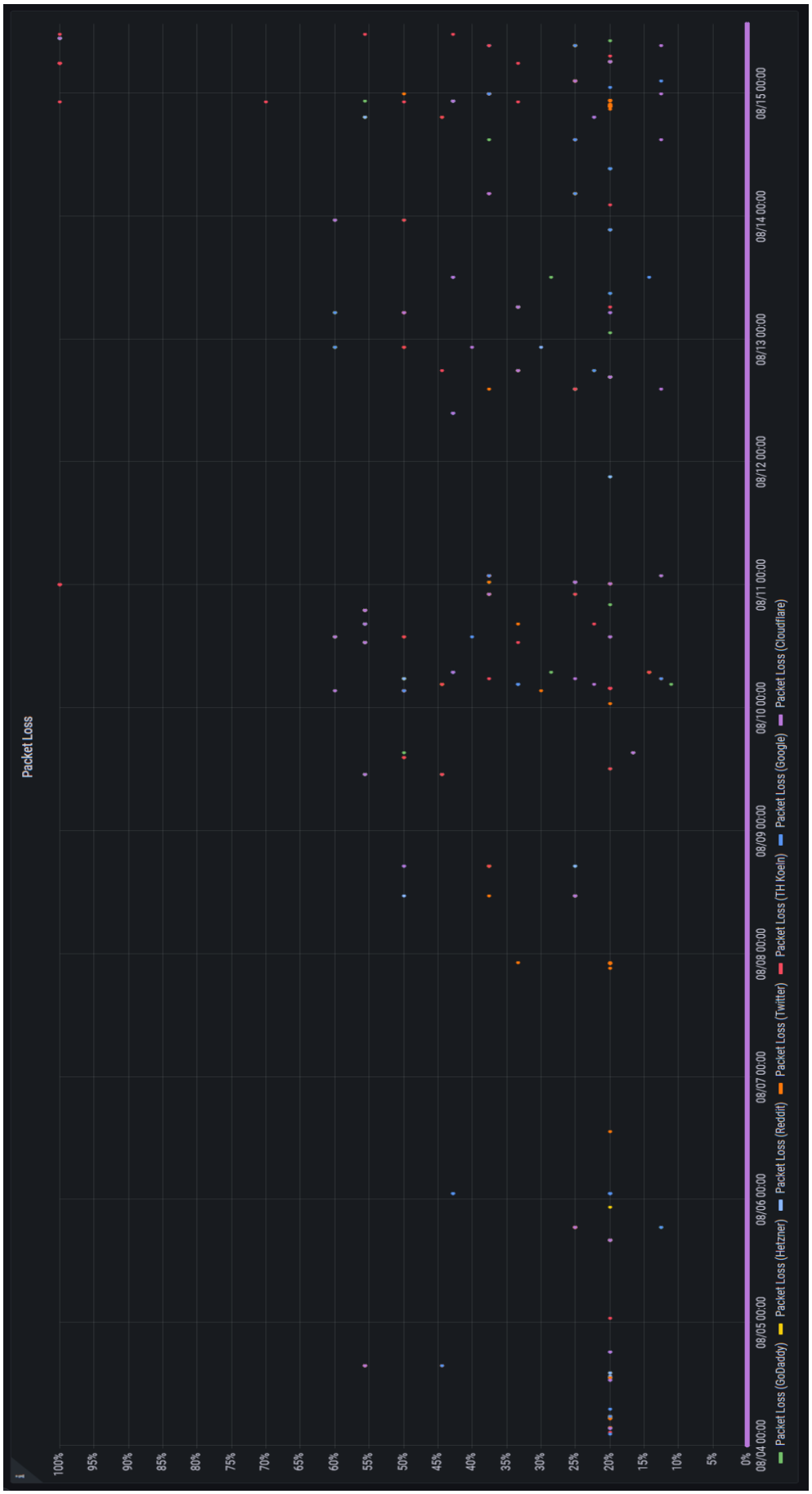


Figure 5.7: Packet Loss (ICMP) - Slice Jumpiot on TIGClient2

5.1.4 DNS-Query Latency

This section presents the DNS-query latency of the 5G Campus Network slices. The metrics are collected using the Telegraf plugin "inputs.dns_query" and measure the time it takes for a DNS-query to be answered. The (sub-)domains for which records are queried are "th-koeln.de", "www.dn.th-koeln.de", "google.com", "reddit.com" and "twitter.com".

The results presented in Figure 5.8 depict DNS-query times for the (sub-)domains mentioned above. The metrics were gathered using TIGClient2 on the network slice "jumpiot" over a period of 11 days. The measurements illustrate that the majority of query times range from 20ms to 35ms. The first day of the measurements depicts a clear difference in query time per domain. The average query time for "twitter.com" of around 21ms is the fastest, followed by "google.com" averaging around 22ms. The query latency for "reddit.com" averages approximately 24ms, and the query times for "th-koeln.de" and "www.dn.th-koeln.de" average about 28ms. The results per domain vary for the remainder of the measurement period, however, the fastest queries in this time interval are achieved for the domain "google.com".

The measurements depicted in Figure 5.9 demonstrate DNS-query times to the selection of (sub-)domains mentioned above. The metrics were collected using TIG-Client1 over a time period of two hours. The query times per domain vary considerably and no domain presents consistently faster query times than another. The vast majority of results is concentrated between 20ms and 35ms.

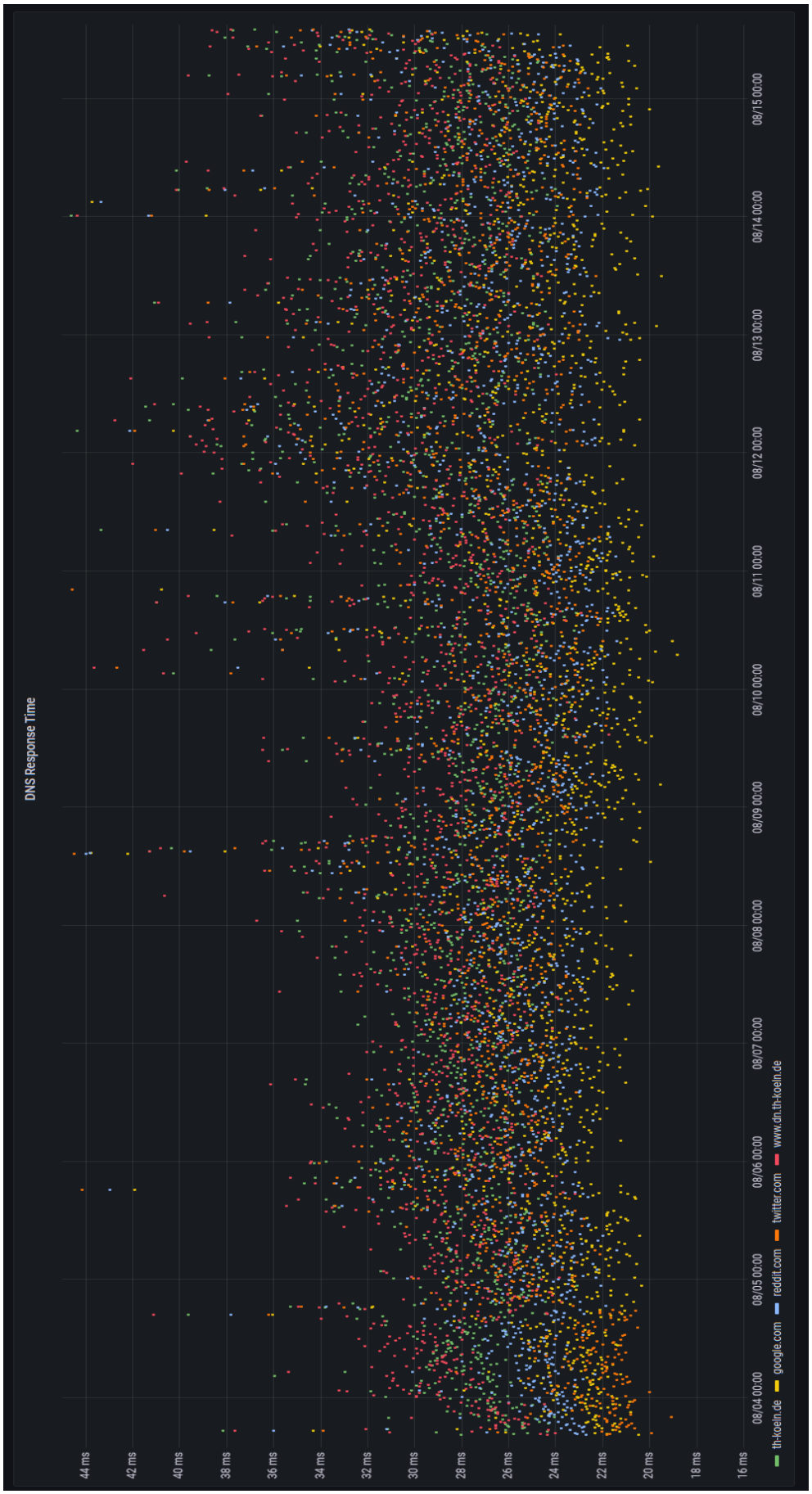


Figure 5.8: DNS-Query Latency - Slice Jumpiot on TIGClient2

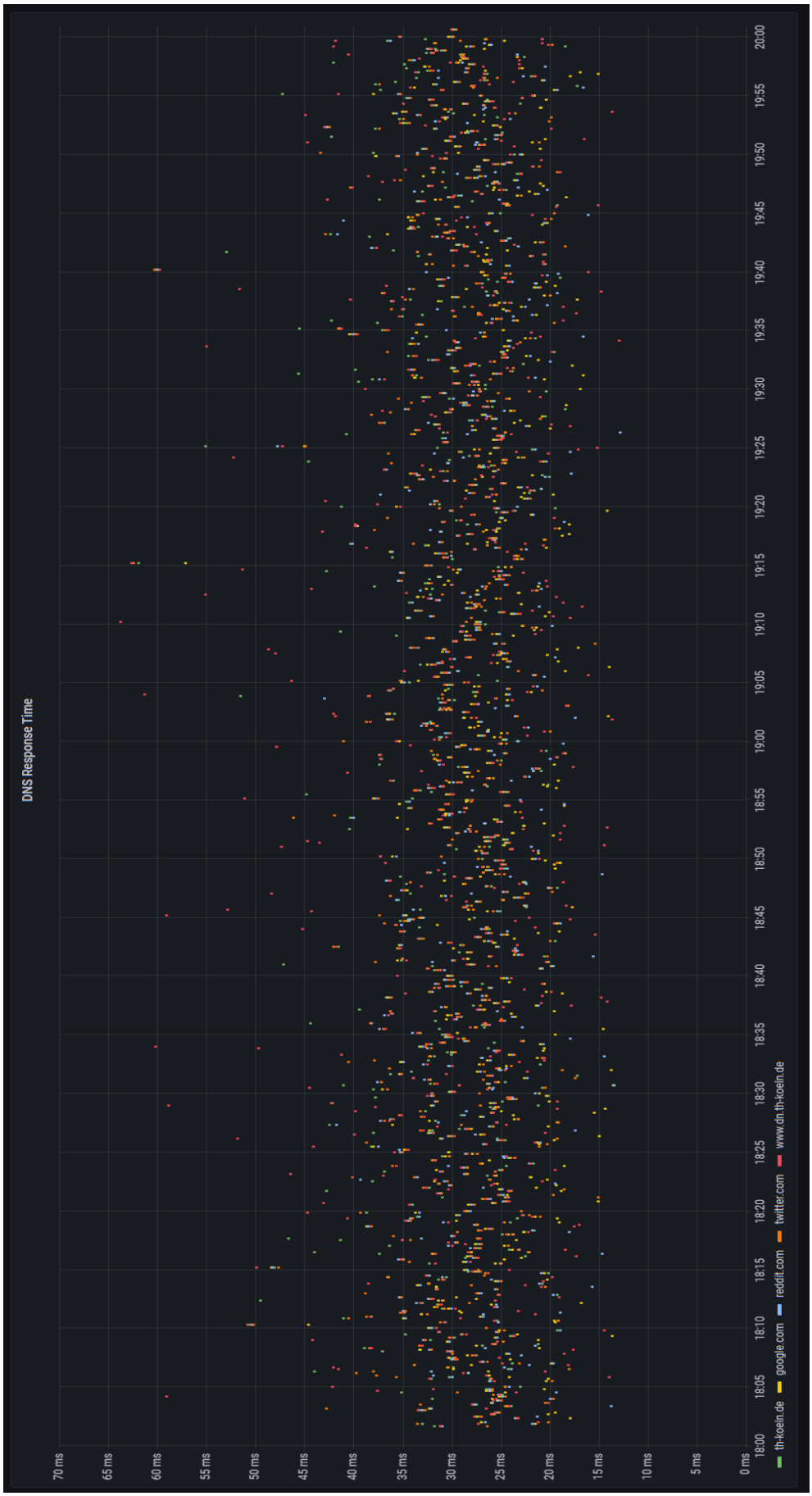


Figure 5.9: DNS-Query Latency - Slice Jumpmediaupload on TIGClient1

5.1.5 Jitter

This section demonstrates the results of the jitter measurements. The metrics are collected using the utility "iperf3" and measure the variance in the packet arrival times using the protocol UDP. The destinations that jitter is measured against, are the servers at the DN.Lab, Hetzner and GoDaddy.

The results depicted in Figure 5.10 present the measurements gathered using TIGClient2 on the network slice "jumpiot" over a period of five hours. The measurements range from 0.467ms to 1.70ms. The majority of results are centered around the 0.6ms mark. The jitter measurements to the server at the DN.Lab yielded predominantly better results than the measurements to Hetzner, with jitter results of largely below 0.6ms.

The metrics illustrated in Figure 5.11 demonstrate the jitter results gathered using TIGClient1 on the slice "publicinternet" over a time interval of 80 minutes. The measurements vary between 0.566ms and 1.67ms, with results focusing around the 0.75ms mark.

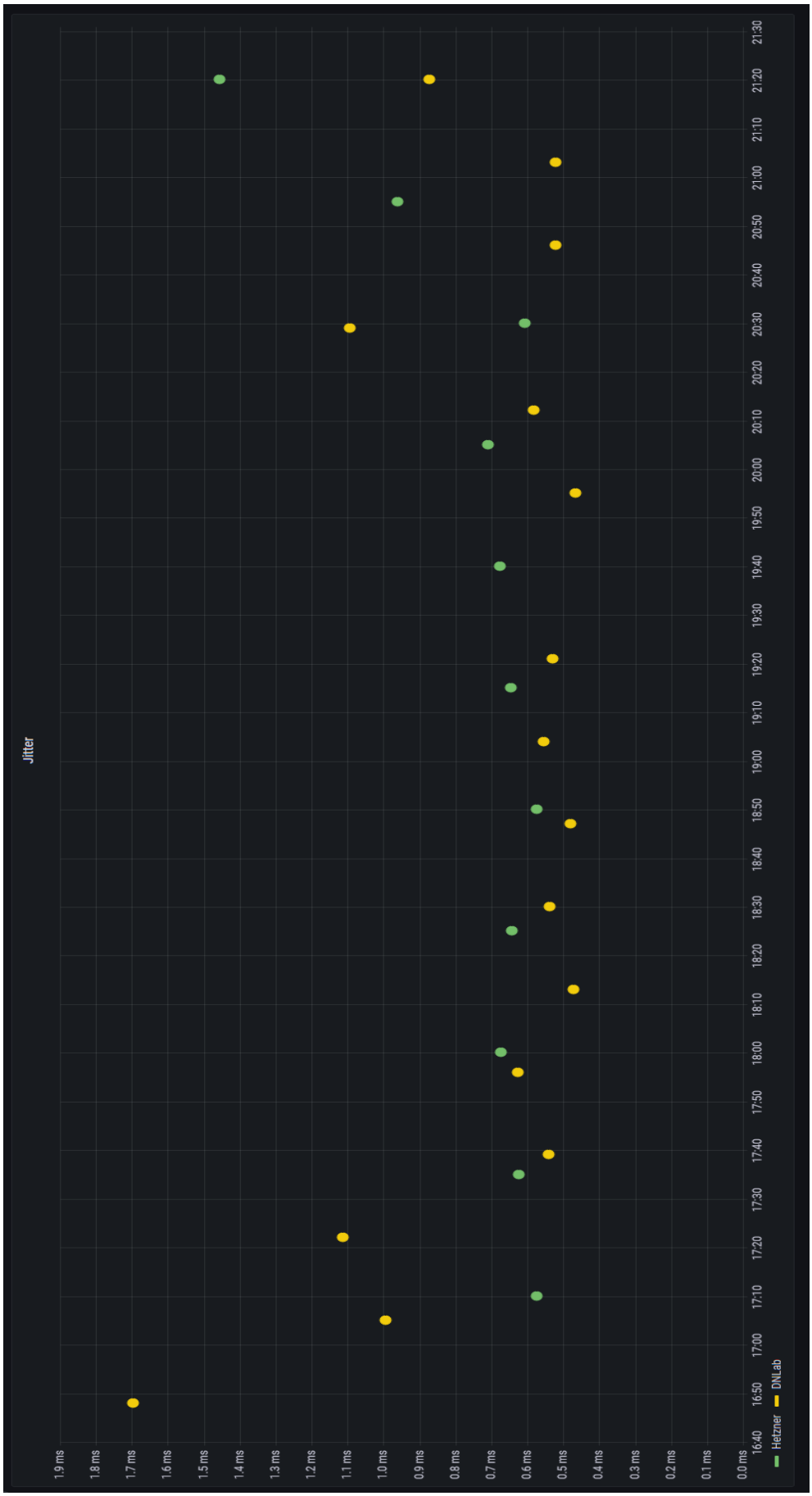


Figure 5.10: Jitter (UDP) - Slice Jumpiot on TIGClient2

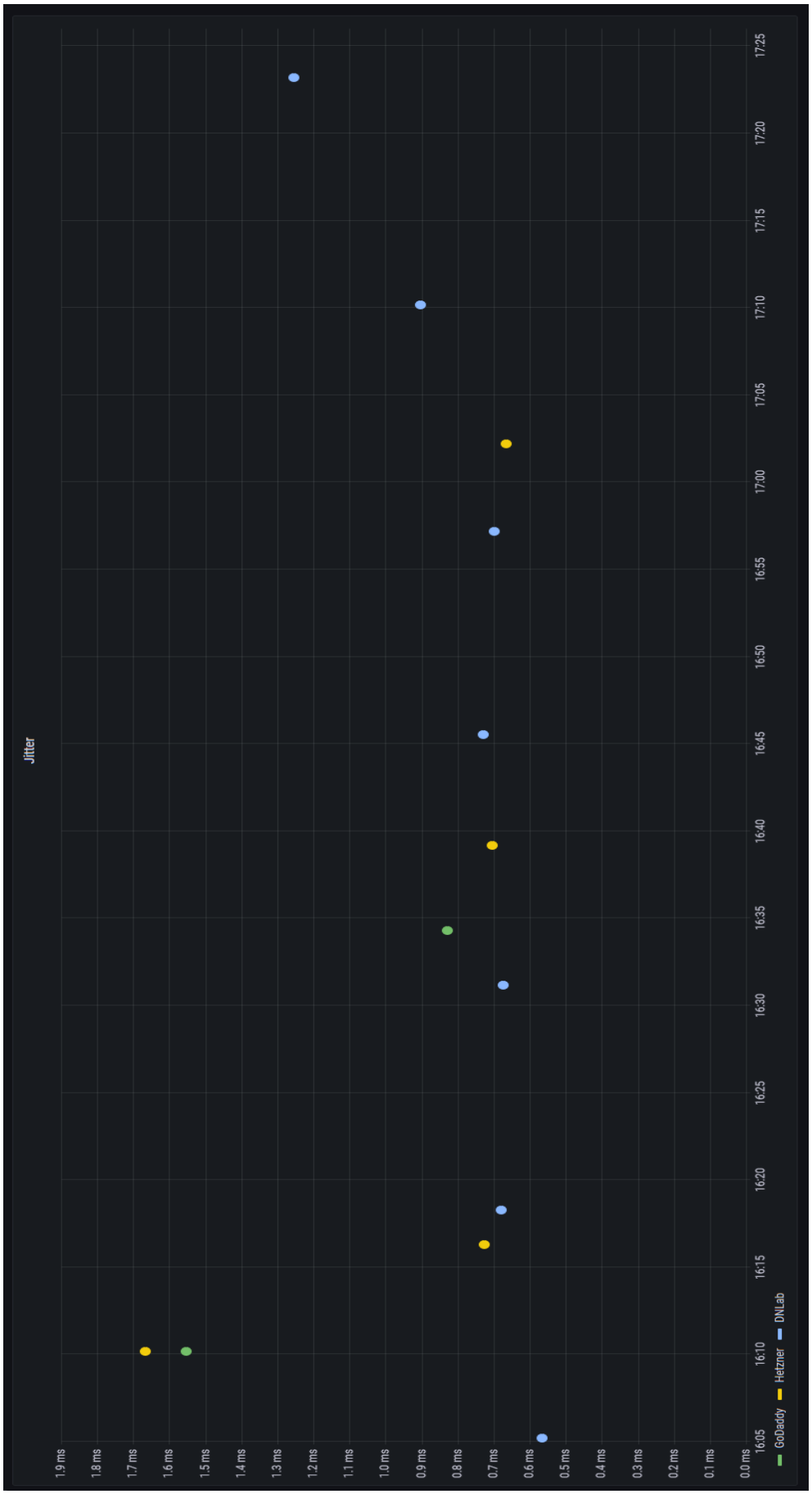


Figure 5.11: Jitter (UDP) - Slice Publicinternet on TIGClient1

5.2 Evaluation Network Tests

This chapter covers the evaluation of the collected results. The evaluation comprises an assessment of the performance of the network slices, using the gathered metrics and KPIs defined previously. Furthermore, the outages and availability of the 5G Campus Network are discussed.

5.2.1 Network Slice Performance Assessment

This section examines the measurement results in the context of the defined KPIs and determines whether these are satisfied. Each KPI is assessed on an individual basis, after which the results are examined as whole. The metrics utilized for the evaluation are found on the data medium included with this thesis.

The first KPI regards the QoS-characteristic throughput. The throughput in the download direction is defined to be at least 80Mbit/s. The measurements conducted on the slice "jumpiot" yielded disparate results concerning the different measurement tools. The tool "speedtest" consistently achieved speeds of 400Mbit/s over a period of 10 days. However, the measurements conducted using "iperf3" to the defined measurement end-points only achieved an average throughput of 70Mbit/s. This contrasts significantly with the throughput results of "speedtest", however is not far off the defined KPI of 80Mbit/s. The throughput results for "publicinternet" in the download direction achieved around 50Mbit/s using "iperf3" and 300Mbit/s using "speedtest" over a period of 80 minutes. Although the results gathered with "iperf3" do not meet the standard set by the KPI's definition, the metrics gathered with "speedtest" more than fulfill the KPI's requirements. The measurements for the slice "mediaupload" were gathered for a period of two hours and achieved around 50 Mbit/s using the utility "iperf3" and 200Mbit/s using "speedtest". This classifies the measurement results gathered using "iperf3" as not meeting the definition of the KPI, while "speedtest" meets the requirements by a large margin.

The next KPI regards the throughput measurements in the upload direction and is set to be at least 30Mbit/s. The throughput metrics for the slice "jumpiot" cover a time period of 10 days. The measurements collected using "iperf3" yielded approximately 80Mbit/s in the upload direction, while "speedtest" achieved results around the 90Mbit/s mark. Therefore, the throughput measurements adhere to the standard set by the KPI, since both tools achieved results of more than twice the defined magnitude. The measurements conducted on the slice "publicinternet" span a time period of 80 minutes. The metrics gathered by "iperf3" yield results around the 50Mbit/s mark, while "speedtest" produced measurements around the 70Mit/s mark. Therefore, the results for the slice "publicinternet" fulfill the requirements of the KPI. The throughput measurements on the slice "jumpmediaupload" cover

a period of two hours. The metrics gathered using "iperf3" yield results focusing at the 55Mbit/s mark, while "speedtest" accomplished speeds around 68Mbit/s. Consequently, the measurements meet the standard set by the KPI.

The subsequent KPI concerns the latency measured on layer 3 of the OSI model using the protocol ICMP. The KPI defines the latency to be under 50ms. The measurements on the network slice "jumpiot" span a time period of 11 days. The overwhelming majority of metrics are located below the 40ms mark, thereby adhering to the KPI's standard. The measurements collected on the slice "jumpmediaupload" span a time interval of two hours. The metrics collected using the Telegraf "inputs.ping" plugin depict the majority of data points below 40ms, with all destinations besides the server at the DN.Lab, largely scoring results below 30ms. Therefore this slice also complies with the KPI's definition. The results collected on the slice "publicinternet" cover a period of 80 minutes. The bulk of results for this slice depict latency metrics of below 40ms. In addition, the results for the server at the DN.Lab accumulate between 34ms and 38ms and the other destinations accomplish most results below the 30ms mark. As a result, this slice meets the requirements defined by the KPI.

The next characteristic applies to the jitter measurements, collected using the utility "iperf3". The KPI's definition assess measurements below 30ms as sufficient. The metrics gathered on the network slice "jumpiot" cover a period of five hours. The results for this slice focus around the 0.6ms mark. Therefore, complying to the definition of the KPI by a considerable margin. The measurements collected using the slice "publicinternet" span a period of 80 minutes. The majority of results are centered around 0.75ms, thereby adhering to the KPI's constraints. Due to complications regarding network availability, the slice "jumpmediaupload" is not assessed for this QoS-characteristics. Further information regarding the availability and outages of the network are covered in a later section.

The ensuing KPIs regard the error rate of the disparate network slices. The KPI concerning packet loss defines this metric to be below 1%. The other KPI applies to TCP-retransmissions and states the upper limit as 2%. The metrics collected for the slice "jumpiot" cover a period of 11 days. The overwhelming majority of TCP-retransmissions accumulate below five per measurement cycle, placing the TCP-retransmission rate far below 1%. Furthermore, the vast majority of ICMP measurements did not experience packet loss, leading to an overall packet loss of below 1%. Therefore, both KPIs are met for the slice "jumpiot". The measurements gathered for the slice "jumpmediaupload" span a time interval of two hours. The majority of TCP-retransmissions lie below 12 per measurement cycle. This leads to a TCP-retransmission rate of far below 1%. Moreover, the metrics collected on packet loss depict few measurements with any packet loss at all. Consequently, the packet loss is below 1% by a large margin. Therefore, the slice "jumpmediaupoad"

complies to both KPIs concerning the error rate. The measurements gathered on the slice "publicinternet" span over a time interval of 80 minutes. The measurements regarding TCP-retransmissions illustrate that the majority of data points are at 12 retransmissions or below for one measurement cycle. This places the TCP-retransmission rate unequivocally below 1%. The metrics on packet loss depict several periods of unavailability during the duration of the measurement. Therefore, the packet loss is extremely high compared to the other network slices. The packet loss for this measurement is 25%. Therefore, the KPI concerning TCP-retransmission rate is fulfilled, while the KPI regarding packet loss is missed by a large margin.

The last KPI concerns the DNS-query latency and is set to below 100ms. The metrics collected for the slice "jumpiot" depict the query latency over a period of 11 days. The majority of values range from 20ms to 35ms, hence placing far below the 100ms mark. On these grounds, the KPI is fulfilled. The measurements on the network slice "jumpmediaupload" span a period of two hours. The majority of metrics accumulate between 20ms and 35ms, thereby adhering to the constraints of the KPI. The measurements collected for the slice "publicinternet" cover a time period of 80 minutes. The large majority of metrics range between 14ms and 40ms. Consequently, the measurements comply with the requirement of KPI.

Overall, all three network slices illustrated performant QoS-characteristics and principally adhere to the defined KPIs. The exceptions include the throughput results in the download direction collected using "iperf3". All three slices failed to consistently reach the 80Mbit/s mark. However, the results gathered using "speedtest" fulfilled the constraints of the KPI. Moreover, the slice "publicinternet" experienced high packet loss during the measurement interval, causing the requirements of the KPI to be missed.

Network Slice Configuration

The configuration of the network slices is fundamental their operation and QoS-characteristics. The network slices in the 5G Co:CreationLab at the Cologne University of Applied Sciences do not yet offer vastly disparate configurations. This leads to the measurement of similar results across all slices. The slice "publicinternet", "jumpmediaupload" and "jumpiot" are configured with an up-link and down-link bandwidth of 4Gbits/s, which in terms of the present infrastructure indicates a maxing out of the bandwidth capabilities. Furthermore, the latency is configured similarly, only the slice "jumpmediaupload" employs two different 5G QoS-profiles. The mapping of these QoS-profiles to the 5QI-standard indicates that with one profile the latency is similar to the other slices, with a packet delay of 150ms and with the second profile, the packet delay can reach up to 300ms. The configuration of the slices is depicted in Table 5.1 [4].

DNN	IP-Version	Latency	Uplink	Downlink	QoS-Profile	Session
publicinternet	IPv4	300ms (best effort)	4Gbps (unlimited)	4Gbps (unlimited)	9	Preserved
jumpmediaupload	IPv4	300ms/ 150ms (best effort)	4Gbps (unlimited)	4Gbps (unlimited)	4/ 2	Preserved
jumpiot	IPv4	300ms (best effort)	4Gbps (unlimited)	4Gbps (unlimited)	9	Preserved

Table 5.1: Overview of the 5G network slices configuration. [90]

5.2.2 Outages and Availability

This sections covers availability and connection issues, in regards to the network infrastructure and the measurement setup. During the course of conducting measurements, various outages occurred that substantially influenced the measurement process. These outages include planned maintenance as well as unplanned disruptions. The majority of the outages are of unplanned nature. The longest outage caused the measurement clients to be disconnected from the network for a period of over 10 days. Every disruption requires the manual reset and restart of the measurement probes leading to significant delays.

Furthermore, the measurement clients experience frequent disconnects. This leads to various measurements being aborted after less than an hour, even though the infrastructure is up an running. The cause of these disconnects could not yet be fully uncovered, however possible issue regard faulty session closure and a policy mismatch with the network according to the measurement probe's logs.

Moreover, the measurement probe TIGClient3 deployed in the Cologne Cobots Lab, was not operational throughout the entire measurement process. All connection-establishment attempts to all three slices failed. These challenges influenced the volume of data that was gathered, as well as the duration of the measurements.

Outlook and Summary

This chapter concludes the thesis with a summary covering the most significant aspects and insights, as well as an outlook on future developments and undertakings.

6.1 Summary

This thesis covered the investigation of slice characteristics in 5G Campus Networks by developing a measurement framework based on the TIG-stack and deploying this framework in the realms of the 5G Co:CreationLab at the Cologne University of Applied Sciences.

The segment on the technical background covered the necessary technologies involved in this process. These technologies comprise 5G cellular networks, especially in regards to 5G network slicing and Campus Networks. Furthermore, the chapter depicts the hardware components that make up a measurement host with cellular connectivity. In addition, this chapter illustrated the requirements and function of the measurement-endpoints.

The subsequent chapter covered the solution concept, that entails the architecture of the software and hardware components involved in the measurement process. This segment details the interfaces between these software components and explains their respective function and interaction with each other. Furthermore, the testing strategy outlined the methodology that elaborates on how to proceed in order to successfully collect and evaluate measurement results. Moreover, this section identified the key measurement characteristics used to assess the 5G network slices. These are namely throughput, latency, jitter, error rate and DNS-query latency. The last section of this chapter quantitatively defined the magnitudes of these measurement characteristics in order meet a certain QoS-level for various widespread applications.

The chapter on implementation dives into the setup of the measurement probe, including all the components employed to enable 5G-connectivity, such as the 5G modem. Furthermore, the implementation of the measurement-endpoints and domain connection thereof is laid out. Thereafter, the interaction of the modem on the CLI is covered. Subsequently, the implementation of the connection-establishment procedure is explained, as well as the process of extracting relevant information on

the operation of the modems. Additionally, the implementation, configuration and deployment of the TIG-stack was clarified. This includes the parameterisation of the specific utilities and plugins used to measure the defined KPIs. Consequently, the functioning of this implementation was demonstrated using a PoC deployment, including the successful collection of measurement results. Lastly, the measurement locations were presented, depicting the specific locations where the measurement probes are deployed and what the cellular infrastructure consists of.

The next section covered the network measurement results and evaluation thereof. At first, the measurements on each QoS-characteristic were presented, covering measurements on throughput, latency, jitter, error rate and DNS-query latency using disparate network slices. The measurements were presented using detailed figures and an explanation thereof. The last section of this chapter covered the evaluation of results, by addressing each QoS-characteristic and comparing the measurements for each network slice to the defined KPIs. The results illustrated that the measurements complied with the constraints of the KPIs with few exceptions. Thereafter, the similarity of the measurement results was elaborated using the network slice configuration of the 5G Co:CreationLab. At last, outages and availability concerns were discussed that affected the measurement process.

6.2 Outlook

There a number of desirable undertakings regarding the further investigation of 5G network slicing. At first, the measurement of significantly disparate slice configurations will aid in creating an understanding, of how network slicing can be utilized to provide QoS-characteristics tailored to specific applications. These network environments can often not be provisioned due to contradicting requirements, preventing future applications to emerge. By deploying multiple virtual networks with differing QoS-characteristics, user-experience can be vastly improved and emerging applications be trialed.

Furthermore, conducting measurements on the recently deployed Low Latency Low Loss Scalable Throughput (L4S) technology at the 5G Co:CreationLab will be fascinating to investigate. This technology offers unprecedented QoS-characteristics regarding latency and reliability and will enable modern technologies such as autonomous driving and Augmented Reality (AR) applications to be based on 5G cellular technology.

Appendix

ModemManager - Modem Listing

```
-----------------------------------
  General
  dbus path: /org/freedesktop/ModemManager1/Modem/0
  device id: 5de19236291f39ba537800aa3b406ef2ea9e0f68
  -----------------------------------
  Hardware |            manufacturer: Quectel
           |                   model: RM500QGL_VH
           |       firmware revision: RM500QGLABR10A02M4G
           |          carrier config: Commercial-DT-VOLTE
           | carrier config revision: 0A011F1F
           |            h/w revision: 20000
           |               supported: gsm-umts, lte, 5gnr
           |                 current: gsm-umts, lte, 5gnr
           |            equipment id: 863305040260159
  -----------------------------------
  System   |
      device: /sys/devices/platform/scb/fd500000.
      pcie/pci0000:00/0000:00:00.0/0000:01:00.0/usb2/2-1
           |                 drivers: option1, qmi_wwan
           |                  plugin: quectel
           |            primary port: cdc-wdm0
           |            ports: cdc-wdm0 (qmi), ttyUSB0 (qcdm),
                                ttyUSB1 (gps), ttyUSB2 (at),
                                ttyUSB3 (at), wwan0 (net)
```

Listing A.1: ModemManager - Modem Information

Sample Telegraf - Configuration

```
# Telegraf Config

# Input plugins

# Ping plugin
[[inputs.ping]]
urls = ["google.com", "reddit.com", "twitter.com",
        "www.dn.th-koeln.de", "th-koeln.de",
        "94.130.106.172", "1.1.1.1", "8.8.8.8", "he.de"]
count = 4
ping_interval = 1.0
timeout = 10.0

# DNS plugin
[[inputs.dns_query]]
  servers = ["8.8.8.8"]
  domains = ["google.com", "twitter.com", "th-koeln.de",
             "netzmessungen.de", "telekom.de", "he.de"]

# Throughput Monitoring using Speedtest by Ookla
[[inputs.exec]]
  commands = ["speedtest --secure --json"]
  interval = "5m"
  timeout = "2m"
  name_suffix = "_speedtest"
  data_format = "json"

  tag_keys = [
    "client_isp",
    "client_ip",
    "server_id",
    "server_name",
    "server_country",
    "server_cc"
  ]

# Jitter Monitoring  - TH Network
```

```
[[inputs.exec]]
  commands=["mtr --report -c 5 srv2.netzmessungen.de
            -o JMX --json"]
  interval = "13m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "jitterTh"

# Jitter Monitoring  - Hetzner
[[inputs.exec]]
  commands=["mtr --report -c 5 srv1.netzmessungen.de
            -o JMX --json"]
  interval = "13m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "jitterHetzner"

# Jitter Monitoring  - GoDaddy
[[inputs.exec]]
  commands=["mtr --report -c 5 srv3.netzmessungen.de
            -o JMX --json"]
  interval = "13m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "jitterGodaddy"

# iPerf3 Monitoring with 4 Threads TCP - TH Network
[[inputs.exec]]
  commands=["iperf3 -c srv2.netzmessungen.de -p5201
            -b 1000M -P 4 --json"]
  interval = "9m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "iperf3th"

# iPerf3 Monitoring with 4 Threads UDP - TH Network
[[inputs.exec]]
  commands=["iperf3 -c srv2.netzmessungen.de -p5201 -u
            -b 1000M -P 4 --json"]
  interval = "13m"
  timeout = "2m"
  data_format = "json"
  json_query = "end"
  name_override = "iperf3thudp"
```

```
86
87 # iPerf3 Monitoring with 4 Threads TCP - Hetzner
88 [[inputs.exec]]
89   commands=["iperf3 -c srv1.netzmessungen.de -p5201
90              -b 1000M -P 4 --json"]
91   interval = "17m"
92   timeout = "2m"
93   data_format = "json"
94   json_query = "end"
95   name_override = "iperf3hetzner"
96
97 # iPerf3 Monitoring with 4 Threads UDP - Hetzner
98 [[inputs.exec]]
99   commands=["iperf3 -c srv1.netzmessungen.de -p5201 -u
100             -b 1000M -P 4 --json"]
101  interval = "23m"
102  timeout = "2m"
103  data_format = "json"
104  json_query = "end"
105  name_override = "iperf3hetznerudp"
106
107
108 # iPerf3 Monitoring with 4 Threads TCP - GoDaddy
109 [[inputs.exec]]
110   commands=["iperf3 -c srv3.netzmessungen.de -p5201
111              -b 1000M -P 4 --json"]
112   interval = "61m"
113   timeout = "2m"
114   data_format = "json"
115   json_query = "end"
116   name_override = "iperf3godaddy"
117
118 # iPerf3 Monitoring with 4 Threads UDP - GoDaddy
119 [[inputs.exec]]
120   commands=["iperf3 -c srv3.netzmessungen.de -p5201 -u
121              -b 1000M -P 4 --json"]
122   interval = "67m"
123   timeout = "2m"
124   data_format = "json"
125   json_query = "end"
126   name_override = "iperf3godaddyudp"
127
128 # Output Plugin InfluxDB
129 [[outputs.influxdb]]
130   database = "5gmessungX"
131   urls = ['http://<IP>:8086']
132
133 ## HTTP Basic Auth
```

```
134   username = "telegraf"
135   password = "XXXXXXXXX"
```

Listing B.1: Telegraf - Sample Configuration of a Measurement Client

Proof of Concept - Results

Figure C.1: PoC - Latency

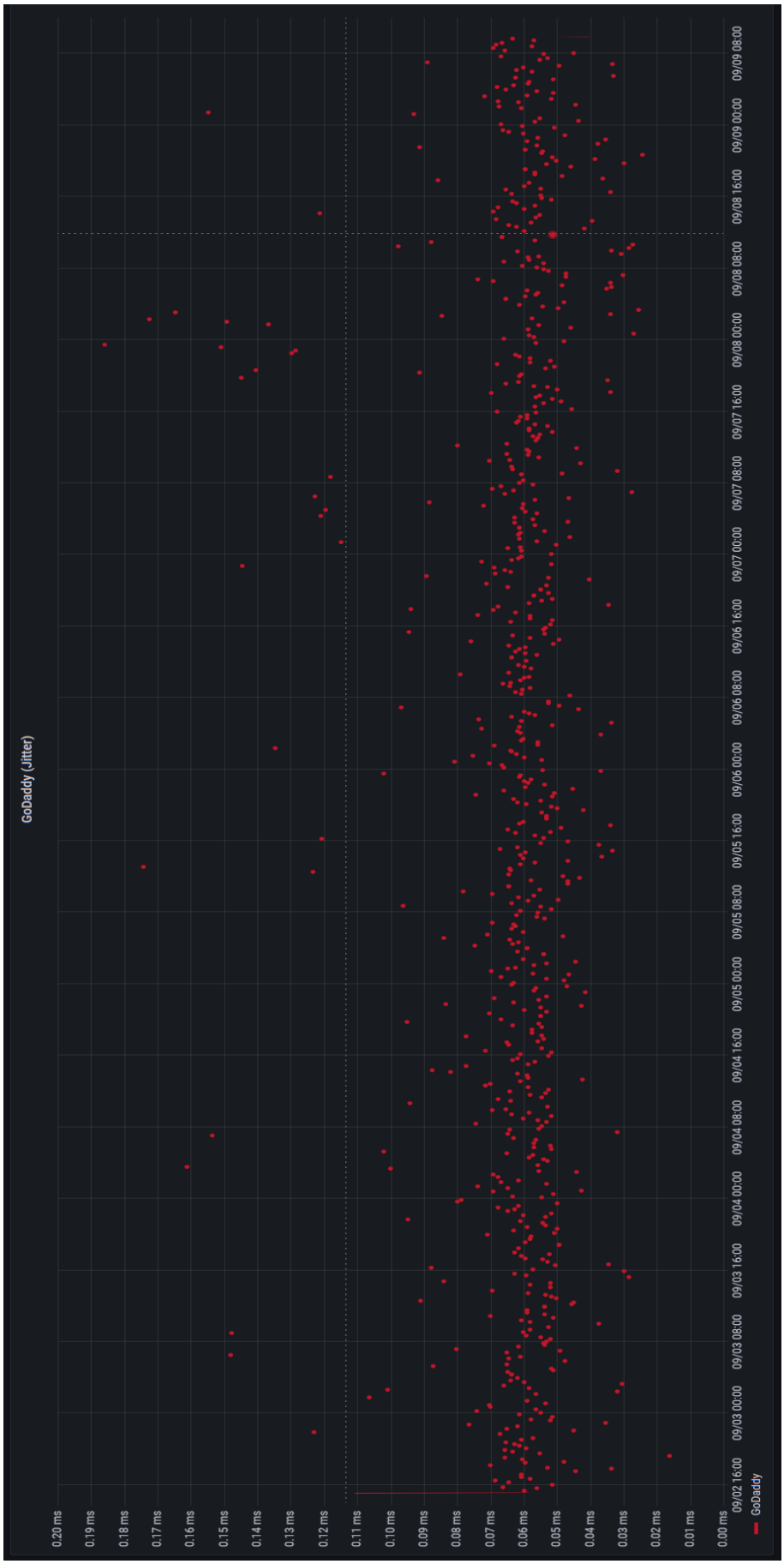


Figure C.2: PoC - Jitter

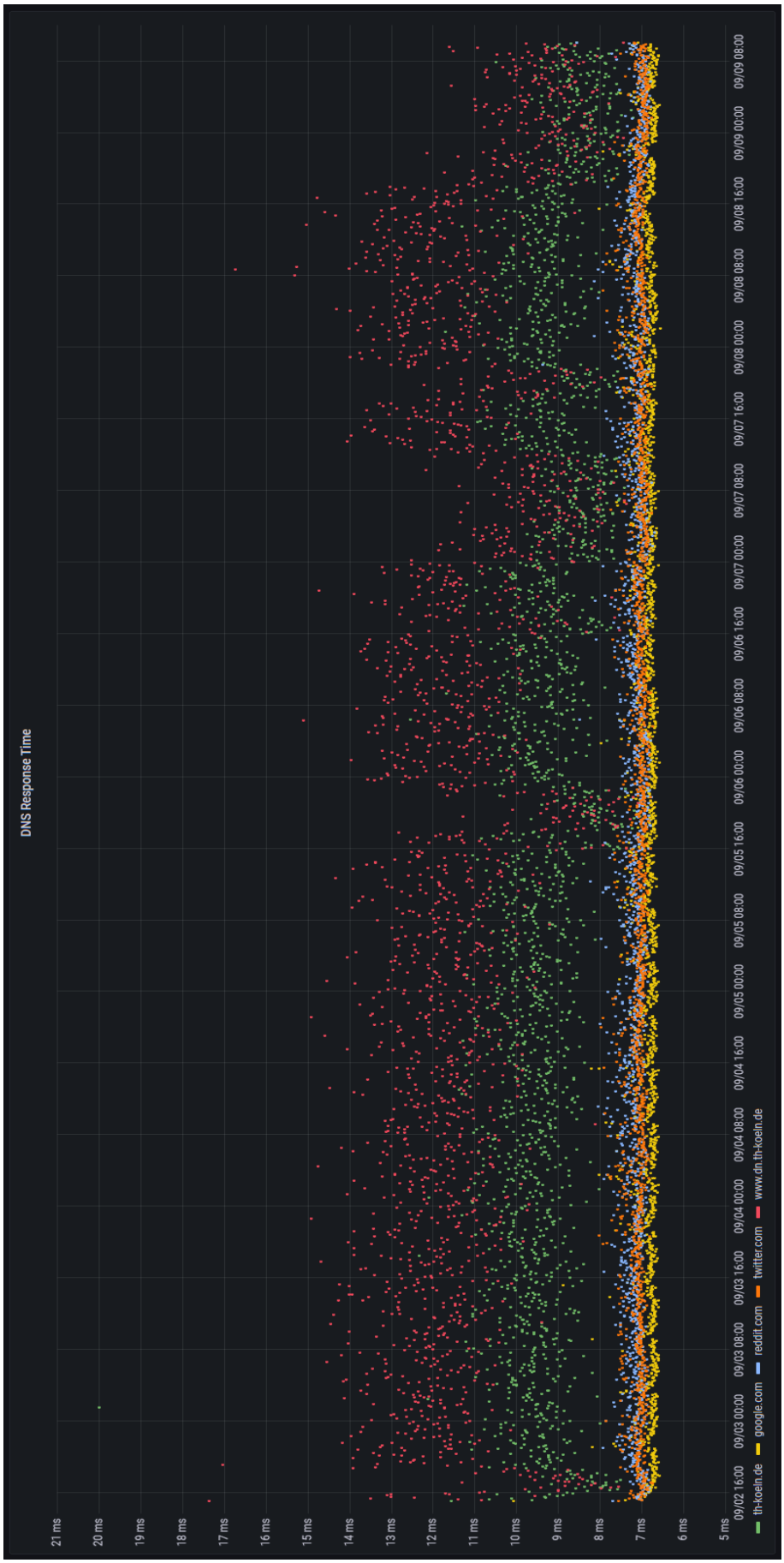


Figure C.3: PoC - DNS Latency

Measurement Location Infrastructure

Figure D.1: 5G Antenna at Gateway Gründungsservice (bottom view)

Measurement Results

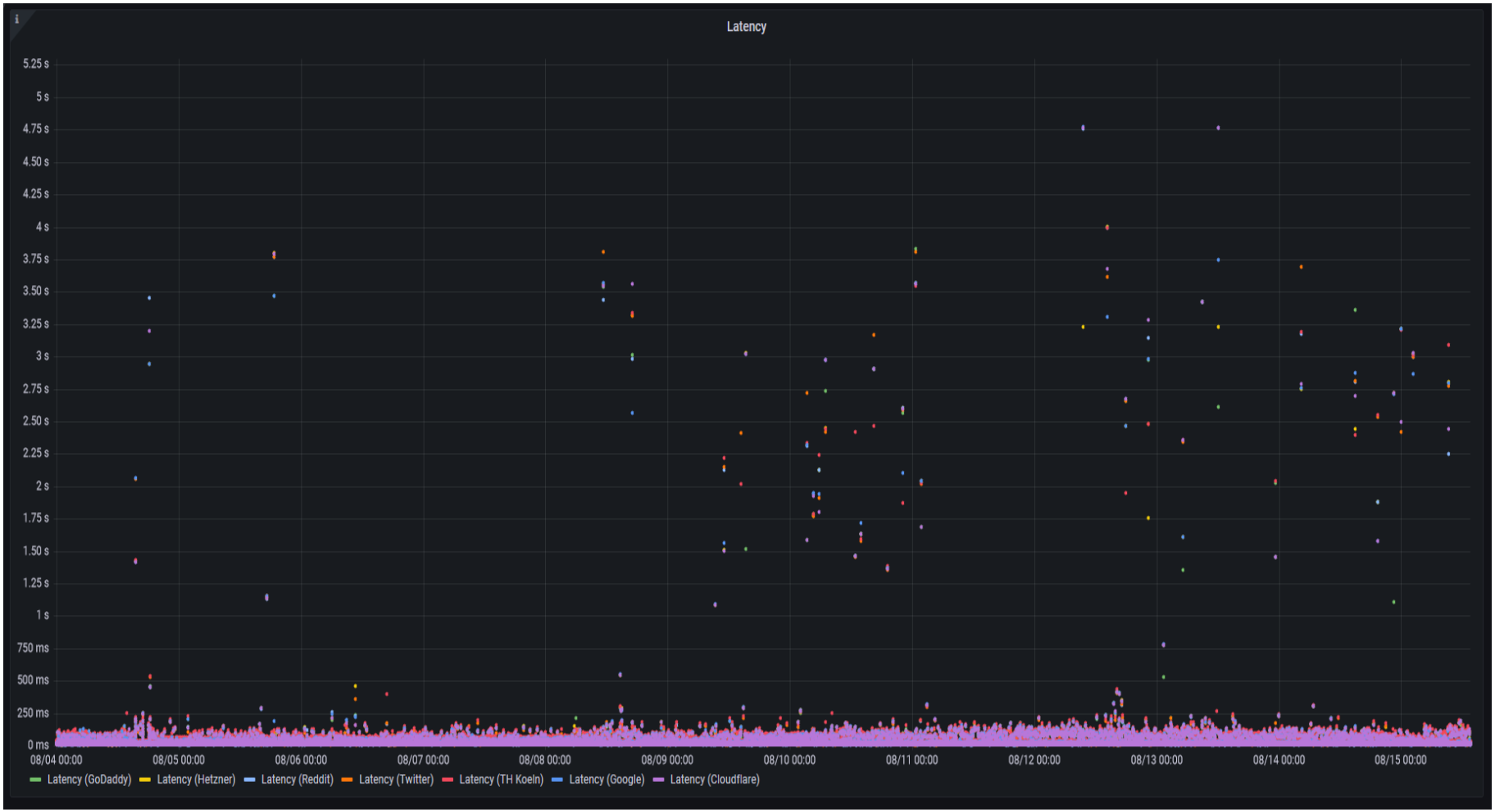


Figure E.1: Latency (ICMP) - Slice Jumpiot on TIGClient2

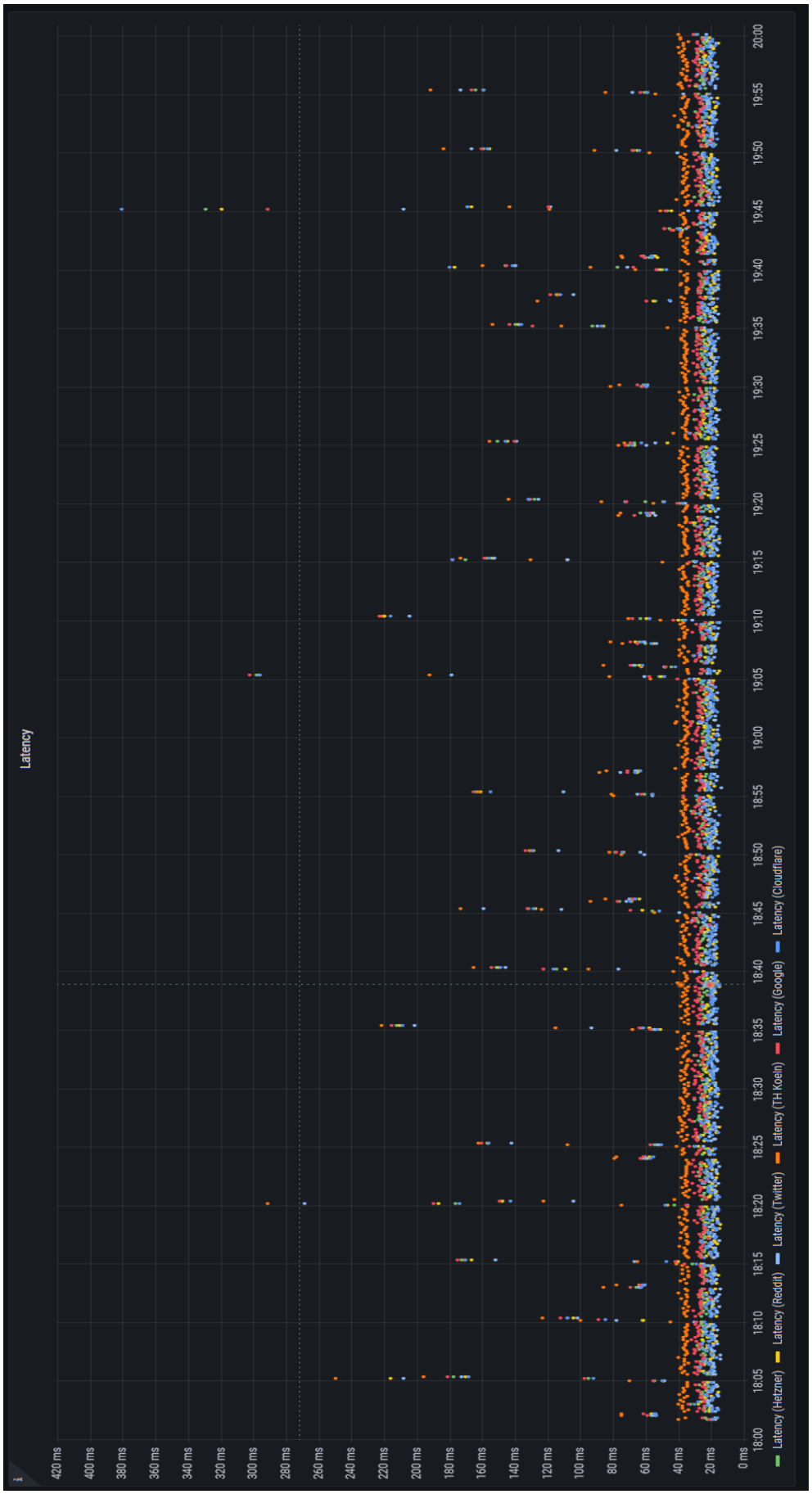


Figure E.2: Latency (ICMP) - Slice Jumpmediaupload on TIGClient1

List of Tables

List of Figures

Listings

Glossary

5G Fifth-generation mobile broadband standard. 4, 5, 11, 18–23, 26, 27, 29, 31, 34, 37, 47, 51, 54, 61, 65, 71, 74, 76, 78, 93

availability Network uptime over a specific time interval. 20

Campus Network Computer network limited to a geographical area. 4, 5, 18–23, 26, 27, 31, 34, 37, 47, 51, 54, 61, 65, 71, 76

Go Programming language developed by Google. 19

Grafana Data visualization platform. 20, 22, 43–45, 47

GUI Graphical User Interface. 15

InfluxDB Time-series optimized DBMS. 19, 20, 37, 43–45, 47

jitter Variance in latency. 20, 23, 38, 39, 68, 72

latency packet delay. 20, 23

Linux Open-source operating system derived from Unix. 17, 19, 23, 37

Raspberry Pi 4 Single-Board Computer. 18

Telegraf Data collection and aggregation agent. 16, 17, 19, 21, 23, 24, 37, 38, 41, 43–45, 47, 48, 58, 61, 65, 72

throughput Amount of data transferred from source to destination over a specific time interval. 22

TIG-stack Software stack consisting of Telegraf (data collection and aggregation), InfluxDB (DBMS) and Grafana (data visualization). 4, 5, 16, 17, 26, 37, 45, 47, 76, 77

Acronyms

5G NR 5G New Radio. 51

API Application Programming Interface. 14, 19, 20, 37

APN Access Point Name. 34

AR Augmented Reality. 78

AT ATtention. 17, 32

CLI command-line interface. 31, 39, 40, 45, 47, 76

DBMS Database Management System. 14, 19, 37, 41, 43, 97

DHCP Dynamic Host Configuration Protocol. 23, 34

DNS Domain Name System. 17, 20, 23–25, 30, 38, 48, 54, 65, 73, 76, 77

GNU GNU's Not Unix!. 17, 19, 23, 37

HAT Hardware attached on top. 18

HDD hard disk drive. 21

HF High frequency. 27

HTTP Hypertext Transfer Protocol. 20, 22, 37, 39, 41, 43, 47, 48

HTTPS Hypertext Transfer Protocol Secure. 20, 22

ICMP Internet Control Message Protocol. 23, 30, 37, 44, 61, 72

IMAP Internet Message Access Protocol. 23

InfluxQL Influx Query Language. 22, 44

IP Internet Protocol. 23, 24, 30, 38

ISP Internet Service Provider. 22

JSON JavaScript Object Notation. 16

KPI Key Performance Indicator. 4, 5, 24, 54, 71–73, 77

L4S Low Latency Low Loss Scalable Throughput. 78

MBIM Mobile Broadband Interface Model. 17

MTR My Traceroute. 23, 38, 48

NIC network interface controller. 21

NPM Network Performance Monitoring. 37

OS Operating System. 16, 17, 19, 21, 23, 26, 30, 37–39

OSI Open Systems Interconnection. 12, 23, 24, 61, 72

PDU Protocol Data Unit. 10, 23

PIN Personal identification number. 32

PoC Proof of Concept. 5, 37, 77

QMI Qualcomm MSM Interface. 17, 32

QoS Quality of service. 4, 5, 16–18, 20–24, 37, 54, 71–73, 76–78, 94

RAID redundant array of independent disks. 21

RAM Random-Access Memory. 26, 30

RAN Radio Access Network. 21

RDP Remote Desktop Protocol. 22

REST Representational state transfer. 19, 20, 37, 41, 47

RP-SMA Reverse-polarity-SMA. 27

RRU Remote Radio Unit. 51

RTT round-trip time. 25, 58

SA Standalone architecture. 27

SCTP Stream Control Transmission Protocol. 30

SIM Subscriber Identity Module. 27, 31, 32

SMA SubMiniature version A. 27, 99

SMS Short Message Service. 32

SMTP Simple Mail Transfer Protocol. 22

SSH Secure Shell. 22

TCP Transmission Control Protocol. 20, 22, 23, 25, 30, 39, 40, 48, 54, 61, 72, 73

TLS Transport Layer Security. 20

TTFB time to first byte. 20

UDP User Datagram Protocol. 20, 22, 23, 30, 39, 40, 68

UE User Equipment. 22

URL Uniform Resource Locator. 43

USB Universal Serial Bus. 17, 31

vCPU Virtual Central Processing Unit. 30

VM Virtual Machine. 15

VoIP Voice over IP. 23, 25, 48

XML Extensible Markup Language. 16

YAML Yet Another Markup Language. 16

Bibliography

[1] 3GPP. About 3gpp. https://www.3gpp.org/about-3gpp, 2022. (visited on: 10/08/2022).

[2] 3GPP. Portal - releases. https://portal.3gpp.org/#/55934-releases, 2022. (visited on: 10/08/2022).

[3] 3GPP. Releases. https://www.3gpp.org/specifications/releases, 2022. (visited on: 11/08/2022).

[4] Mohamed Abbas. Quality of service (qos) in 5g networks. https://5ghub.us/quality-of-service-qos-in-5g-networks/, Feb 2021. (visited on: 13/08/2022).

[5] Deutsche Telekom AG. Easy and simple - network slicing. https://www.telekom.com/en/company/details/network-slicing-485776, Feb 2017. (visited on: 11/08/2022).

[6] Mercateo Deutschland AG. Delock 63166 schnittstellenkarte/adapter m.2 eingebaut. https://www.mercateo.com/p/CIDE7-85839976/DeLOCK_63166_Schnittstellenkarte_Adapter_M_2_Eingebaut.html, 2022. (visited on: 16/09/2022).

[7] Tom Arbuthnot. What are thresholds for good and poor network packet loss, jitter and round trip time for unified communications? https://tomtalks.blog/what-are-thresholds-for-good-and-poor-network-packet-loss-jitter-and-round- May 2018. (visited on: 28/09/2022).

[8] Internet Assigned Numbers Authority. Service name and transport protocol port number registry. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, Sep 2022. (visited on: 15/09/2022).

[9] The ModemManager Authors. https://modemmanager.org/, 2022. (visited on: 21/09/2022).

[10] Baeldung. Propagation delay vs transmission delay. https://www.baeldung.com/cs/propagation-vs-transmission-delay, Oct 2021. (visited on: 15/09/2022).

[11] BitWizard. Traviscross/mtr: Official repository for mtr, a network diagnostic tool. https://github.com/traviscross/mtr, 2022. (visited on: 15/09/2022).

[12] BMWi. https://www.bmwi.de/Redaktion/EN/Publikationen/Digitale-Welt/guidelines-for-5g-campus-networks-orientation-for-small-and-medium pdf?__blob=publicationFile&v=2, Apr 2020. (visited on: 12/08/2022).

[13] Linux Mobile Broadband. Linux mobile broadband. https://github.com/linux-mobile-broadband, 2022. (visited on: 06/09/2022).

[14] Davide Brunello, Ingemar Johansson S, Mustafa Ozger, and Cicek Cavdar. Low latency low loss scalable throughput in 5g networks. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–7, 2021. (visited on: 12/08/2022).

[15] Lorenzo Casaccia. Understanding 3gpp – starting with the basics: Qualcomm. https://www.qualcomm.com/news/onq/2017/08/understanding-3gpp-starting-basics, Aug 2017. (visited on: 10/08/2022).

[16] CenturyLink. How to improve your gaming latency. https://www.centurylink.com/home/help/internet/how-to-improve-gaming-latency.html, 2022. (visited on: 28/09/2022).

[17] Gerald Combs and Wireshark core developers. Wireshark foundation / wireshark · gitlab. https://gitlab.com/wireshark/wireshark, 2022. (visited on: 15/09/2022).

[18] Arch Linux community. Systemd. https://wiki.archlinux.org/title/systemd, Sep 2022. (visited on: 18/09/2022).

[19] IBM Corporation. At commands. https://www.ibm.com/docs/en/aix/7.2?topic=troubleshooting-commands, Aug 2022. (visited on: 06/09/2022).

[20] IBM Corporation. How tls provides identification, authentication, confidentiality, and integrity. https://www.ibm.com/docs/en/ibm-mq/9.1?topic=tls-how-provides-identification-authentication-confidentiality-integrity, Aug 2022. (visited on: 12/09/2022).

[21] Datapath.io. What is acceptable jitter? https://medium.com/@datapath_io/what-is-acceptable-jitter-7e93c1e68f9b, Jun 2016. (visited on: 15/09/2022).

[22] Rhonda D'Vine. List of sections in "jammy". https://packages.ubuntu.com/jammy/, 2022. (visited on: 18/09/2022).

[23] IBM Cloud Education. What is docker? https://www.ibm.com/cloud/learn/docker. (visited on: 25/08/2022).

[24] Roy T. Fielding and Julian Reschke. Hypertext Transfer Protocol (HTTP/1.1): Authentication. https://datatracker.ietf.org/doc/html/rfc7235, June 2014. (visited on: 09/09/2022).

[25] freedesktop.org. https://www.freedesktop.org/software/libqmi/man/latest/qmicli.1.html, 2021. (visited on: 21/09/2022).

[26] freedesktop.org. What is libqmi? https://www.freedesktop.org/wiki/Software/libqmi/, May 2021. (visited on: 21/09/2022).

[27] freedesktop.org. What is modemmanager? https://www.freedesktop.org/wiki/Software/ModemManager/, May 2021. (visited on: 21/09/2022).

[28] freedesktop.org. Mobile broadband connectivity / libqmi · gitlab. https://gitlab.freedesktop.org/mobile-broadband/libqmi, 2022. (visited on: 21/09/2022).

[29] freedesktop.org and contributors. What is libmbim? https://www.freedesktop.org/wiki/Software/libmbim/, May 2021. (visited on: 06/09/2022).

[30] freekdesktop.org. Mobile broadband connectivity / modemmanager · gitlab. https://gitlab.freedesktop.org/mobile-broadband/ModemManager, 2022. (visited on: 21/09/2022).

[31] Ashwitha B G. *TIG (Telegraf, Influx and Grafana) stack*. Thoughtworks Inc., Feb 2021. (visited on: 07/09/2022).

[32] Admin Globaldots. 13 key cloud computing benefits for your business. https://www.globaldots.com/resources/blog/cloud-computing-benefits-7-key-advantages-for-your-business/, Jul 2018. (visited on: 12/09/2022).

[33] T-Systems International GmbH. 5g campus networks lte and 5g-technology for local company networks. https://www.t-systems.com/resource/blob/12462/8a982ffd985a0f3bc65c918b573633b7/

DL-Factsheet-5G-Campus-Netze-t-systems-en-11-2019.pdf, 2019. (visited on: 13/08/2022).

[34] T-Systems International GmbH. 5g campus-networks. https://www.t-systems.com/de/en/connectivity/5g-campus-networks, 2022. (visited on: 13/08/2022).

[35] Michaela Goss. An overview of 3gpp 5g releases and what each one means. https://www.techtarget.com/searchnetworking/feature/An-overview-of-3GPP-5G-releases-and-what-each-one-means, Feb 2021. (visited on: 10/08/2022).

[36] The Wireless Haven. Quectel rm500q-ae 5g nr sub-6 ghz modem module with adapter enclosure. https://thewirelesshaven.com/shop/modems-hotspots/quectel-rm500q-ae-enclosure/, 2022. (visited on: 16/08/2022).

[37] Docker Inc. Docker overview. https://docs.docker.com/get-started/overview/, 2022. (visited on: 25/08/2022).

[38] Docker Inc. Overview of docker compose. https://docs.docker.com/compose/, 2022. (visited on: 25/08/2022).

[39] Fibocom Wireless Inc. 5g module fm150-ae. https://techship.com/download/fibocom-fm150-ae-datasheet/, 2022. (visited on: 18/09/2022).

[40] Fibocom Wireless Inc. Fm150-ae. https://www.fibocom.com/en/products/5G-FM150-AE.html, 2022. (visited on: 18/09/2022).

[41] Google Inc. Bandwidth, data usage, and stream quality. https://support.google.com/stadia/answer/9607891?hl=en#zippy=, 2022. (visited on: 28/09/2022).

[42] InfluxData Inc. Authentication and authorization in influxdb. https://docs.influxdata.com/influxdb/v1.8/administration/authentication_and_authorization/, 2022. (visited on: 25/09/2022).

[43] InfluxData Inc. Configure influxdb oss. https://docs.influxdata.com/influxdb/v1.8/administration/config/#http-endpoints-settings, 2022. (visited on: 25/09/2022).

[44] InfluxData Inc. Influxdata/influxdb: Scalable datastore for metrics, events, and real-time analytics. https://github.com/influxdata/influxdb, 2022. (visited on: 18/08/2022).

[45] InfluxData Inc. Influxdata/telegraf: The plugin-driven server agent for collecting & reporting metrics. https://github.com/influxdata/telegraf, 2022. (visited on: 16/08/2022).

[46] InfluxData Inc. Plugin directory. https://docs.influxdata.com/telegraf/v1.23/plugins/, 2022. (visited on: 16/08/2022).

[47] InfluxData Inc. Telegraf open source server agent: Influxdb. https://www.influxdata.com/time-series-platform/telegraf/, Jul 2022. (visited on: 16/08/2022).

[48] InfluxData Inc. Telegraf/plugins/inputs/ping at master · influxdata/telegraf. https://github.com/influxdata/telegraf/tree/master/plugins/inputs/ping, 2022. (visited on: 23/09/2022).

[49] InfluxData Inc. Telegraf/readme.md at master · influxdata/telegraf. https://github.com/influxdata/telegraf/tree/master/plugins/inputs/ping, 2022. (visited on: 23/09/2022).

[50] IONOS Inc. Influxdb – explanation, advantages, and first steps. https://www.ionos.com/digitalguide/hosting/technical-matters/what-is-influxdb/, Sep 2020. (visited on: 18/08/2022).

[51] Linux Kernel Organization Inc. cdc_mbim - driver for cdc mbim mobile broadband modems. https://www.kernel.org/doc/Documentation/networking/cdc_mbim.txt, 2022. (visited on: 06/09/2022).

[52] Netflix Inc. Internet connection speed recommendations. https://help.netflix.com/en/node/306, 2022. (visited on: 28/09/2022).

[53] Sierra Wireless Inc. Em919x/em7690 product technical specification. https://source.sierrawireless.com/-/media/support_downloads/airprime/hardware_specs_user_guides/41113174-em919x-em7690-product-technical-specification-r5.ashx, Mar 2022. (visited on: 18/09/2022).

[54] USB Implementers Forum Inc. Mobile broadband interface model v1.0 errata -1 and adopters agreement. https://www.usb.org/document-library/mobile-broadband-interface-model-v10-errata-1-and-adopters-agreement, Dec 2012. (visited on: 06/09/2022).

[55] GSMA Intelligence. *Bandwidth and latency requirements for different applications*. Intelligence, GSMA, 2014. (visited on: 28/09/2022).

[56] Iot-Lnu. Iot-lnu/tig-stack: Tig-stack. https://github.com/iot-lnu/tig-stack, Jul 2021. (visited on: 26/09/2022).

[57] Sacha Kavanagh. What is enhanced mobile broadband (embb). https://5g.co.uk/guides/what-is-enhanced-mobile-broadband-embb/, Nov 2021. (visited on: 12/08/2022).

[58] TH Köln. Makerspace und mini-inkubatoren. https://www.th-koeln.de/forschung/makerspace-und-mini-inkubatoren_77114.php, 2022. (visited on: 16/09/2022).

[59] ESnet / Lawrence Berkeley National Laboratory. Esnet/iperf: Iperf3: A tcp, udp, and sctp network bandwidth measurement tool. https://github.com/esnet/iperf, 2022. (visited on: 15/09/2022).

[60] Grafana Labs. Grafana - dashboard anything. observe everything. https://grafana.com/grafana/, 2022. (visited on: 25/08/2022).

[61] Grafana Labs. Grafana plugins - data sources. https://grafana.com/grafana/plugins/?type=datasource, 2022. (visited on: 12/09/2022).

[62] Grafana Labs. Grafana/grafana: The open and composable observability and data visualization platform. https://github.com/grafana/grafana, 2022. (visited on: 20/08/2022).

[63] Grafana Labs. Intro to time series. https://grafana.com/docs/grafana/latest/basics/timeseries/, 2022. (visited on: 20/08/2022).

[64] Grafana Labs. Visualization panels. https://grafana.com/docs/grafana/latest/visualizations/, 2022. (visited on: 12/09/2022).

[65] Fujitsu Limited. Datenblatt fujitsu esprimo q520. http://support.harlander.com/upload-artikelsupport/computer/fujitsu-siemens/esprimo-q520/datenblatt-q520.pdf, Feb 2017. (visited on: 16/09/2022).

[66] Ookla LLC. Speedtest by ookla. https://play.google.com/store/apps/details?id=org.zwanoo.android.speedtest, Jul 2022. (visited on: 14/09/2022).

[67] Canonical Ltd. Ubuntu 22.04.1 lts (jammy jellyfish). https://releases.ubuntu.com/22.04/, Aug 2022. (visited on: 18/09/2022).

[68] Panorama Antennas Ltd. 2g/3g/4g/5g paddle antenna pwb-bc3g-38-rsmap. https://cdn-reichelt.de/documents/datenblatt/F100/PWB-BC3G-38-RSMAP_DB-EN.pdf, Mar 2019. (visited on: 16/09/2022).

[69] Quectel Wireless Solutions Co. Ltd. Quectel rm500q-gl - iot/embb-optimized 5g sub-6 ghz m.2 modul. https://www.quectel.com/wp-content/

uploads/2021/03/Quectel_RM500Q-GL_5G_Specification_V1.3.pdf, Mar 2021. (visited on: 18/09/2022).

[70] Quectel Wireless Solutions Co. Ltd. 5g rm50xq series. https://www.quectel.com/product/5g-rm50xq-series, 2022. (visited on: 15/08/2022).

[71] Quectel Wireless Solutions Co. Ltd. Quectel rm50xq serie. https://www.quectel.com/wp-content/uploads/2021/09/Quectel_RM50xQ_Series_5G_Specification_V1.1.pdf, 2022. (visited on: 15/08/2022).

[72] Matt Martz. Sivel/speedtest-cli: Command line interface for testing internet bandwidth using speedtest.net. https://github.com/sivel/speedtest-cli, Apr 2021. (visited on: 15/09/2022).

[73] Anshu Mishra. Delays in computer network. https://www.geeksforgeeks.org/delays-in-computer-network/, Sep 2021. (visited on: 15/09/2022).

[74] Eiman Mohyeldin. Minimum technical performance requirements for imt-2020 radio interface(s). https://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2020/Documents/S01-1_Requirements%20for%20IMT-2020_Rev.pdf, 2016. (visited on: 12/08/2022).

[75] Mike Muuss. Iputils/ping.c at master · iputils/iputils. https://github.com/iputils/iputils/blob/master/ping/ping.c, Aug 2022. (visited on: 15/09/2022).

[76] Palo Alto Networks. What is quality of service? https://www.paloaltonetworks.com/cyberpedia/what-is-quality-of-service-qos. (visited on: 13/08/2022).

[77] Federal Network Agency of Germany. Frequenzauktion 2019 - ergebnis der runde 78. https://www.bundesnetzagentur.de/_tools/FrequenzXml/Auktion2019_XML/078.html, 2022. (visited on: 15/08/2022).

[78] Joe O'Halloran. Ericsson launches time-critical communication to guarantee real-time 5g experiences. https://www.computerweekly.com/news/252508696/Ericsson-launches-Time-Critical-Communication-to-guarantee-real-time-5G-exp Oct 2021. (visited on: 12/08/2022).

[79] Carin Overturf. How does speedtest measure my network speeds? https://help.speedtest.net/hc/en-us/articles/360038679354-How-does-Speedtest-measure-my-network-speeds-, Nov 2019. (visited on: 23/09/2022).

[80] postmarketOS.org contributors. Qmi. https://wiki.postmarketos.org/wiki/QMI, May 2020. (visited on: 06/09/2022).

[81] The Debian Project. Software packages in "bullseye". https://packages.debian.org/stable/allpackages, 2022. (visited on: 01/09/2022).

[82] K Ramakrishnan, Sally Floyd, and D Black. Rfc3168: The addition of explicit congestion notification (ecn) to ip. https://www.rfc-editor.org/rfc/rfc3168.html, 2001. (visited on: 12/08/2022).

[83] reichelt elektronik GmbH & Co. KG. Delock 90398 - hf antennenkabel, sma buchse zum einbau zu mhf® 4 stecker, 35 cm. https://www.reichelt.de/ch/de/hf-antennenkabel-sma-buchse-zum-einbau-zu-mhf-4-stecker-35-cm-delock-90398-p2663 html?trstct=pos_2&nbc=1&&r=1, 2022. (visited on: 16/09/2022).

[84] Aiswarya Lakshmi Renganathan. Video streaming traffic study in india. https://www3.cs.stonybrook.edu/~arunab/course/2019-3.pdf, Mar 2019. (visited on: 15/09/2022).

[85] Salvatore Sanfilippo (Antirez). Antirez/hping: Hping network tool. https://github.com/antirez/hping, 2014. (visited on: 15/09/2022).

[86] Wilfried Seifert. Ubuntu lte verbindung via kommandozeile. https://www.thomas-krenn.com/de/wiki/Ubuntu_LTE_Verbindung_via_Kommandozeile, Jul 2021. (visited on: 21/09/2022).

[87] Styopa Semenukha. How we use virtual clouds to achieve virtually uninterrupted uptime. https://developmentgateway.org/blog/how-we-use-virtual-clouds-to-achieve-virtually-uninterrupted-uptime/, Jan 2013. (visited on: 12/09/2022).

[88] M Series. Imt vision–framework and overall objectives of the future development of imt for 2020 and beyond. https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf, 2015. (visited on: 11/08/2022).

[89] Dirk Seul. Co:creationlab at th-köln, Jul 2021.

[90] Dirk Seul. slice_config_udm_v091, 2021.

[91] Dirk Seul. Schematics co:creationlab cologne, Aug 2022.

[92] Jordan Shamir. 5 benefits of virtualization. https://www.ibm.com/cloud/blog/5-benefits-of-virtualization, Apr 2021. (visited on: 12/09/2022).

[93] Prakhar Srivastav. A docker tutorial for beginners. https://docker-curriculum.com/, 2022. (visited on: 25/08/2022).

[94] History Computer Staff. The complete history of the modem. https://history-computer.com/modem-complete-history-of-the-modem/, Oct 2021. (visited on: 06/09/2022).

[95] Techship. How to step by step set up a data connection over qmi interface using qmicli and in-kernel driver qmi_wwan in linux? https://techship.com/faq/how-to-step-by-step-set-up-a-data-connection-over-qmi-interface-using-qmicl Dec 2021. (visited on: 22/09/2022).

[96] Techship. Using networkmanager and modemmanager in linux to automatically establish a connection and configure ip details. https://techship.com/faq/using-network-manager-and-modem-manager-in-linux-to-automatically-establish 2022. (visited on: 21/09/2022).

[97] Torvalds. qmi_wwan.c - torvalds/linux. https://github.com/torvalds/linux/blob/master/drivers/net/usb/qmi_wwan.c, Aug 2022. (visited on: 06/09/2022).

[98] Jack Wallen. How to install a tig stack on ubuntu 18.04. https://www.techrepublic.com/article/how-to-install-a-tig-stack-on-ubuntu-18-04/, Jan 2020. (visited on: 06/09/2022).

[99] Dirk Wallerstorfer. Detecting network errors and their impact on services. https://www.dynatrace.com/news/blog/detecting-network-errors-impact-on-services/, Dec 2015. (visited on: 28/09/2022).

[100] Gavin Wright. 1000base-t (gigabit ethernet). https://www.techtarget.com/searchnetworking/definition/1000BASE-T, Aug 2021. (visited on: 26/09/2022).

[101] Kent Yunk. Voip bandwidth requirements: A small business guide. https://www.8x8.com/blog/voip-phone-bandwidth-requirements, May 2019. (visited on: 28/09/2022).

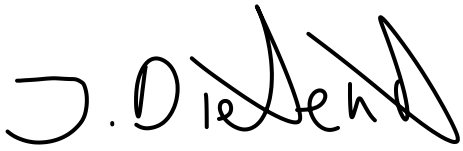
[102] zafaco GmbH. Broadband measurement. https://play.google.com/store/apps/details?id=com.zafaco.breitbandmessung, Feb 2022. (visited on: 14/09/2022).

[103] PerfOps Sp z.o.o. Dns performance analytics and comparison. https://www.dnsperf.com/, 2022. (visited on: 28/09/2022).

Statutory Declaration

Ich erkläre an Eides statt, dass ich die vorgelegte Abschlussarbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Köln, den 29. September 2022



Ort, Datum

Unterschrift
(Dieterich, Arn Jonas)