HOCHSCHULE MITTWEIDA
University of Applied Sciences

BACHELOR THESIS

Mr.
Jesco Vogt

Design and Development of a User-Centric Distributed Ground Station Network Providing Bi-Directional Communications Access to Small Satellites

Mittweida, January 2023

Faculty of **Applied Computer Sciences and Biosciences**

BACHELOR THESIS

Design and Development of a User-Centric Distributed Ground Station Network Providing Bi-Directional Communications Access to Small Satellites

Author:
Jesco Vogt

Course of Study:
Mediainformatics and interactive Entertainment

Seminar Group:
MI19w3-B

First Examiner:
Dipl.-Ing. FH Alexander Marbach

Second Examiner:
Dipl.Inform. Alexander Kleinschrodt

Submission:
Mittweida, 08.01.2023

Defense/Evaluation:
Mittweida, 2023

Faculty of **Applied Computer Sciences and Biosciences**

BACHELOR THESIS

Design und Entwicklung eines benutzerzentrischen, verteilten Bodenstationsnetzwerks für den bidirektionalen Kommunikationszugang zu Kleinsatelliten

Author:
Jesco Vogt

Course of Study:
Medieninformatik und interaktives Entertainment

Seminar Group:
MI19w3-B

First Examiner:
Dipl.-Ing. FH Alexander Marbach

Second Examiner:
Dipl.Inform. Alexander Kleinschrodt

Submission:
Mittweida, 08.01.2023

Defense/Evaluation:
Mittweida, 2023

Bibliographic Description:

Vogt, Jesco:
Design and Development of a User-Centric Distributed Ground Station Network Providing Bi-Directional Communications Access to Small Satellites. – 2023. – 49 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Faculty of Applied Computer Sciences and Biosciences, Bachelor Thesis, 2023.

Referat:

In the field of satellites it is common practice to combine multiple ground stations into one network, to increase communication times with satellites. This work focuses on TIM, which is an international academic colaborative project. Important criteria for this project are elaborated and used to evaluate existing ground station networks. It concludes that there is no appropriate solution availiable for this specific use case and establish a proposed solution. The proposed ground station network software will be elaborated and evaluated.

Contents

List of Figures

List of Tables

Acronyms

ADC . . . . . Analog Digital Converter

API . . . . . Application Programming Interface

ASK . . . . . Amplitude Shift Keying

AWS . . . . . Amazon Web Service

COTS . . . . . comertial of the shelf

ESA . . . . . European Space Administration

FSK . . . . . Frequency Shift Keying

GENSO . . . . . Global Educational Network for Satellite Operations

GEO . . . . . Geosynchronous Earth Orbit

GSN . . . . . Ground Station Network

ISS . . . . . International Space Station

LEO . . . . . Low Earth Orbit

NASA . . . . . National Aeronautics and Space Administration

PSK . . . . . Phase Shift Keying

RF . . . . . Radio Frequency

RLS . . . . . Regional Leaders Summit

RMI . . . . . Remote Method Invocation

RPC . . . . . Remote Procedure Call

SDR . . . . . Software Defined Radio

SiDS . . . . . Simple Downlink Share Convention

SSC . . . . . Swedish Space Cooperation

TIM . . . . . Telematic International Mission

TLE . . . . . Two-Line Element

TOM . . . . . Telematics earth Observation Mission

UI . . . . . User Interface

USA . . . . . United States of America

USRP . . . . . Universal Software Radio Peripheral

VLEO . . . . . Very Low Earth Orbit

VM . . . . . Virtual Machine

ZfT . . . . . Zentrum für Telematik

1 Introduction

1.1 Motivation

Over the last two decades, the landscape of satellite engineering and space research underwent a big paradigm shift. The concept of small satellites emerged, gained a great amount of relevance and enabled a wide range of research projects [27, p. 3]. This can be traced back to the direct correlation between the cost to launch a spacecraft into orbit and its mass. The heavier the object, the more fuel is needed to lift it of the ground and up into space. Small satellites can also be deployed as secondary payload as part of the launch of a large primary spacecraft. This makes them very versatile and further helps reducing the costs. Compared to their large sized counterparts, the small satellites usually use comertial of the shelf (COTS) components where feasible which reduces development costs. Without the need to develop custom satellite sensors or other spacecraft elements researches can focus on more relevant aspects of their mission while keeping costs low. [52, p. 1]

Although the idea to build lighter and smaller spacecraft is as old as 1960, it took a lot of advances in technology and miniaturization to enable the widespread use that small satellites have today [16]. Secondary payload options or dedicated launches of multiple small satellites also did not exist when the concept was first developed [34]. With more launch options tailored around the special requirements of small satellites, the possibility to launch small spacecrafts at an affordable price made the field of satellites accessible to a wide variety of actors [15, p. 2]. At first this was limited to a small number of academic projects, but they also found growing commercial interest in the recent years as well [27, p. 6]. Today there are countless academic, commercial and private projects all around the globe that were made possible by lower cost of entry and better accessibility. [27, p. 5]

One of these projects is the Telematic International Mission (TIM). TIM can be described as a global initiative to create a constellation of small cooperative satellites for Earth Observation. The project comprises several members from the Regional Leaders Summit (RLS) that all participate in this effort with respective research projects. TIM is organized under the technical leadership of the Zentrum für Telematik (ZfT), a bavarian organization. As this work will be developed in conjunction with the ZfT, it is important to mention their contribution to the TIM project. They will develop a formation of three small satellites specifically aimed at observing ash clouds of volcanoes called Telematics earth Observation Mission (TOM). This work will be directly contributing to these projects and aims to support upcoming missions. Since partners only consist of academic institutes and universities, TIM is an academic project. [24, p. 2]

While small satellites have a long list of benefits, they also pose an interesting set of challenges in regard to communication. Small satellites are usually deployed in Low Earth Orbit (LEO), which is characterized by short time windows during which satellites are in reach of a ground station. This communication window is restricted to a brief period of time each day, which is incredibly deficient as it drastically limits the ability to contact the satellite [52, p.39]. During the time where communication is not possible the ground station furthermore sits idle, which is subject to optimization. To solve this issue, it is common practice to share the available resources of the ground station and make them accessible to other partners for communicating with their own satellites. This is done by a

Ground Station Network (GSN) which is responsible for routing packets between the respective operations center, the remote ground station and the satellite it is currently communicating with. This solves unused idle time and expands the communication window since the satellite is available through partnered ground stations. Furthermore, it increases situational awareness for operators and introduces the possibility of immediately contacting the satellite in case of emergency events, which require rapid input of the operator.

In the case of the TIM GSN there is a special set of requirements. The RLS partners providing their ground stations have their own existing sets of hardware and software, which is already able to track their own satellites and communicate with them. Their solution usually utilizes some form of custom code, which they would prefer to retain. Respective solutions for tracking and communication also vary in a few key aspects, which need to be respected when trying to incorporate them into the GSN. Another critical requirement is the need for bidirectional access to a satellite over the network. Some existing GSNs for example only enable the capture of received data, which is not sufficient for this use case. The most significant point which needs to be addressed are the legal issues. Due to regulations imposed by different overseeing actors and organizations, remote control of ground station hardware is out of the question due to liability. The control over the ground station as well as liability needs to remain with the operator of the ground station.

These criteria are further elaborated upon in the next chapter. Once established, existing GSN will be evaluated based on these criteria. This work will come to the conclusion, that there is no applicable network which could be utilized for TIM and establish a concept based on the lessons learned from existing GSNs. The proposed solution is outlined in the subsequent section.

1.2 Considered Approach

The system established and evaluated in this work uses the preexisting infrastructure of the partner ground station to the greatest extent feasible. This is accomplished by expanding the preexisting system with a decentralized planning and communication structure needed for a GSN. It is not productive to implement or integrate the necessary range of needed hardware interfaces into the software.

Instead, it is much more feasible to provide a standardized interface to access relevant data, which can then be used by the preexisting ground station software that is already functional. Furthermore, the already running software is tested and verified to be working during multiple independent missions, which is another great advantage provided by this approach.This way new partners can join without having to fundamentally change their underlying system. The risk of catastrophic failure is minimized drastically. Each partner also has the ability to decide how their station is to be used, since it is important that the station remains under the control of the respective institute, regarding legal liability. This includes the movement of the antenna as well as the transmission of signals.

A further vital aspect of our solution is the flexibility necessary so it can be applied to all TIM partners. It needs to be adaptable enough to cover the unique aspects that differ between partners. The last prerequisite of this system is bidirectional access. This means, not only received packages shall be recorded by the remote stations, but also uplinks over these stations should be possible in order to

request data, or to configure the satellite even when not in direct reach of its home-station. The GSN is a real time data transmission system that transmits packets as soon as they enter the system, provided that a connection to the satellite can be established.

2 Background

Since the field of space engineering presupposes a wide range of prior knowledge and background information, it is required to sketch out the necessary foundations to this field of research. This chapter summarizes a variety of required subjects to deepen the understanding of this topic and lay the necessary groundwork. Crucial constraints enforced by the use-case will be established and elaborated to form the basis of this work.

2.1 Small Satellites

Since the word *small* is not bound to a precise measurement, the term *Small Satellites* can be seen as a hyperonym that can be used for a wide variety of spacecrafts [52, p. 7]. There are numerous definitions for what is considered a small satellite, however, there are a few terms that are used consistently with similar definitions across multiple sources. Pico-Satellites refer to very small satellites, with a mass below 1 kg. Nano-Satellites categorize satellites with a mass in the range of 1 to 10 kg. Micro-Satellites are defined to be above 10 kg, however there are different definitions of the upper bound. [27, p. 2] [52, p. 8] [7, p. 3]

This work will categorize spacecrafts using the most detailed definition, as established by the Federal Aviation Administration of the United States of America (see Table 2.1).

Small Satellites in this context will refer to all satellites in the range of Femto to Mini, as established in [7] as well as [38, p. 1]. This range of satellites played a crucial role in enabling new and innovative concepts. Figure 2.2 shows the evolution of size and weight of launched spacecrafts over the last years. One can see that Pico and Nano-Satellites paved the way for the new mega constellations of e.g. Starlink and OneWeb, and are still launched in relevant numbers in the Pico, Nano and Micro size.

Small satellites have an interesting history regarding their upcoming and establishment in the scientific community. They were first explored in the 1960s, where numerous small satellites were launched. This was mostly due to the fact that there were limited capabilities to deploy heavier objects into LEO. With increasing payload capacity, the option to build heavier and much more capable

Class Name	Kilograms (kg)
Femto	0.01 - 0.09
Pico	0.1 - 1
Nano	1.1 - 10
Micro	11 - 200
Mini	201 - 600
Small	601 - 1,200
Medium	1,201 - 2,500
Intermediate	2,501 - 4,200
Large	4,201 - 5,400
Heavy	5,401 - 7,000
Extra Heavy	>7,000

Table 2.1: Spacecraft Mass Class [1, p. 94]

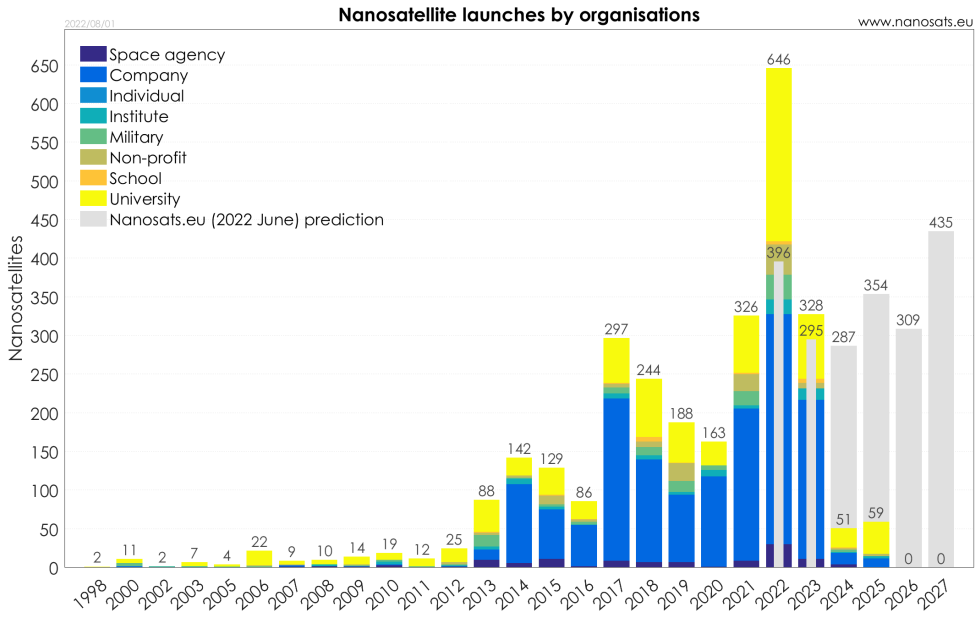


Figure 2.1: Nanosats by Organisation [27, p. 6]

and complex spacecraft became much more attractive and overshadowed small satellites. This area, where large and heavy spacecrafts dominated the space environment, is known as the small satellite doldrums. 1987 marked the end of this trend. Technology started to enable small equally capable spacecrafts and two small satellite conferences were held [16, p. 3]. This evolved into the trend that can be documented over the last two decades to minimize the size and weight of satellites. Figure 2.1 shows the amount of launched nano satellites per year, separated by the type of organization the satellite was created and launched for. The data indicates that Universities were the first major source of nano satellites. This is due to the fact that universities operate on a very limited budget compared to huge space programs like National Aeronautics and Space Administration (NASA) or European Space Administration (ESA). They were the first to take advantage of these advancements in technology that lowered the barrier of entry for this line of research. Another significant trend is the growing commercial interest in small satellites, which can be documented over the last decade. Due to their low costs, small satellites are attractive for achieving commercial goals such as earth observation, remote sensing, communications, and more[10, p. 7]. Figure 2.2 gives an insight into how big the impact of commercial spaceflight is. The two biggest profit-oriented projects are Starlink and OneWeb, both of which aim to provide internet via satellites, were highlighted due to their high importance for the spacecraft landscape [44] [59]. The impact of these two projects alone is so significant, that the number of deployed satellites by only these two projects distorts the graph significantly.

Despite their large impact, this work will refrain from discussing satellites from a commercial perspective. OneWeb or Starlink satellites require high performance with the ability to handle vast amounts of data with low rates of failure for their respective use case. To accomplish this they usually require specialized and heavy equipment, which is the reason why they fall into the Micro and Mini category. The focus of this study is the field of academic small satellites in the weight range of Pico and Nano, which is heavily dominated by CubeSats.

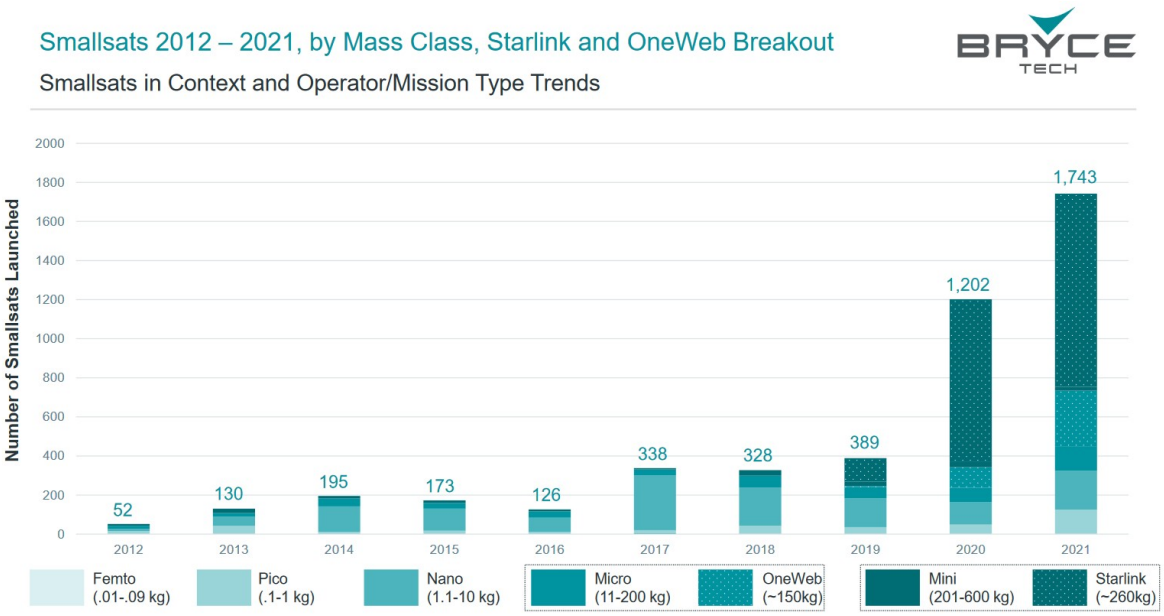


Figure 2.2: Smallsats 2012 – 2021, by Mass Class, Starlink and OneWeb Breakout [7, p. 8]

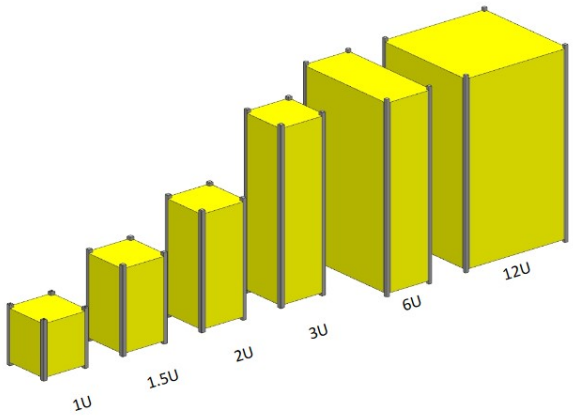


Figure 2.3: The Current CubeSat Family (1U – 12U) [55, p. 8]

2.1.1 CubeSat Specification

The success of small satellites can not be explored without mentioning the CubeSat Specification. The CubeSat Project was started in 1999, with the intent to improve accessibility to space, cut down on development costs and time, and enable frequent launches. It was a joint effort between Prof. Jordi Puid-Suari from the California Polytechinc State University and San Luis Obispo and Prof. Bob Twiggs from the Stanford University's Space Systems Development Laboratory [55, p. 7]. The joined effort resulted in a globally respected standard (see [55]), which is utilized by a wide variety of actors in this field.

The designed features recognizable rails on each side of the CubeSat, which are used by the launcher to safely deploy the spacecraft by sliding it along its internal rails [55, p. 11, 12] [45, p. 3]. The design is flexible and supports a variety of different sizes ranging from 1U or even 0.25U to 12U or sometimes even more, which are showcased in figure 2.3. 1U or one unit is defined to be a 10cm cube. This specification was due to the size of specific COTS components. For example there is a

Figure 2.4: Third Iteration of the P-POD [43, p. 2]

variety of solar cells in the range of 30 mm x 70 mm on the marked. With a size of 10 cm per unit, CubeSats can fit two solar cells on each face [43, p. 2, 3]. The Specification also introduces limits for the size and center of gravity for each respective CubeSat Design (1U – 12U). [55]

The CubeSat specification marked a major advancement in the field of small satellites. The standard made access to space easier than ever, due to the standardized launch options it provided [15, p. 2]. Figure 2.4 shows such a launching pod, which is able to launch a standardized CubeSat with a size of 1U up to 3U. Today there exists a diverse landscape of CubeSats in orbit. CubeSats make up the vast majority of launched Pico and Nano-Satellites, with 3U CubeSats being the most common. [27, p. 7]

2.1.2 Low Earth Orbit

LEO is the most important and popular orbit for satellites besides geosynchronous orbit (for an illustration of both orbits refer to figure 2.5). It is defined as a range of orbits from a height of 400 km minimum up to 1500km maximum. One of the most critical aspects for this range is the lifetime of a deployed satellite. [46, p. 44, 45]

Orbits lower than 400km are called Very Low Earth Orbit (VLEO) and experience a lot of atmospheric drag which causes them to decay very quickly. At this orbit height atmospheric drag becomes one of the primary issues for the spacecraft. While there are some satellites that are launched in VLEO, they are all defined by their serious consideration of atmospheric drag and how it will be accounted for. [61]

The upper bound of LEO is limited by the Van Allen Radiation Belt. This belt is an area around the earth where charged particles are trapped in the Earth's magnetic field. Theses particles cause major issues for spacecrafts, making this area the harshest radiation environment between VLEO and geosynchronous orbit [31] [46, p. 44]. Due to this issue there is only a small fraction of CubeSats that are launched beyond LEO, with most Nano-Satellites being deployed in LEO [27, p. 8] [27, p. 6]. The most popular use case of satellites in this zone is earth observation, similar to satellites created

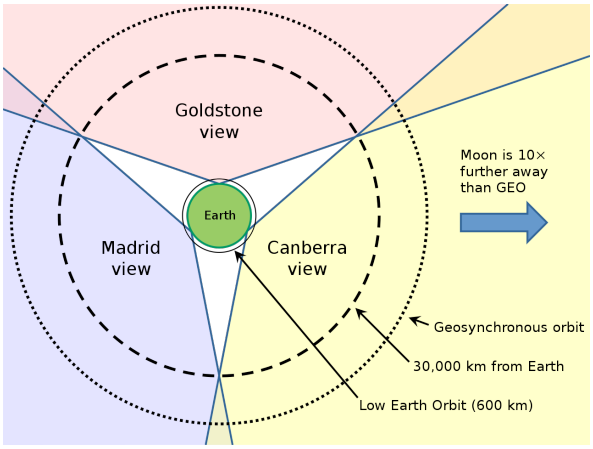


Figure 2.5: North pole view of NASA's Deep Space Network depicting the view angles of each ground station

for TOM which are also deployed in LEO [46, p. 44]. This is due to the fact that LEO offers greater resolution compared to higher orbits. However, it is sacrificed for coverage since the section one camera is able to cover is much smaller [46, p. 46]. This orbit height is also useful for communication. Less distance provides better signal strength and therefor less power needed by the satellite. It also removes the need for bulky communication equipment which makes it perfect for small satellites which operate on a limited budget for mass and power, due to their smaller surface area that can be used to collect solar energy.

2.1.3 Communication

An integral part of each spacecraft is the communication module, which handles up to three tasks. Important data, captured by the satellite needs to be transmitted to a ground station associated with the satellite (downstream). Commands that will be executed by the satellite need to be sent to the spacecraft from an respective ground station (upstream). And the last task is to communicate with other satellites (crosslink or inter-satellite link). Upstream and downstream require the spacecraft to be in range of the respective ground station, otherwise there is no communication possible. In range in this context means, that the ground station can point its antenna at the satellite and is able communicate via electromagnetic waves, encoded with data. This time window where effective communication can occur will be referred to as an overpass in this work as established in [9, p. 5]. [38]

Small Satellites in LEO have unique properties. Since they have a low orbit, they need to fly at higher speeds so the centripetal force keeps them in place. They fly at high speeds (approximately 7km/s relative to earth's surface) and orbit earth several times a day. This means that they are able to cover a larger distance compared to typical spacecrafts, which usually are deployed in Geosynchronous Earth Orbit (GEO) causing them to stand still over one specific region of the earth. Since they are closer to the surface of the earth it is significantly harder to contact them consistently. Figure 2.5 displays this dilemma. The Deep Space Network which is used as an example is able to cover every Satellite in geosynchronous orbit, but is unable to provide any significant coverage for LEO. This factor causes contact windows to be very limited, with around 6 to 8 and a duration of 5 to 15 minutes each on one day for a single ground station. [52, p.39]

The perfect scenario would be to be able to contact the satellite at any point in time. This would provide the operator of the satellite with more situational awareness of the satellite, regarding the state of the satellite, compared to only receiving insight about its status at certain limited overpasses. This would further benefit the utilization of the satellite. A satellite capturing images of the earth in addition is able to capture and downlink more images in the same amount of time, since it is not -as much- restricted by the downlink volume. The goal of this work is to find an optimal solution to this problem. This includes the creation of a GSN concept, allowing to overcome these problems by achieving more communication time to the satellites as described in the next section. The design is required to incooperates use-case specific restrictions, which are elaborated and summarized in this chapter.

2.2 Ground Station

A ground station can be summarized as a ground based setup enabling the exchange of data with a satellite. To operate a satellite it is necessary to send commands (upstream) and receive collected data (downstream). A ground station provides the necessary hardware and software to enable this functionality. On the hardware side this requires an antenna that can be pointed at the satellite when its in range. On the software side this requires tracking of overflights of targeted satellites, as well as calculating which way the antenna needs to be pointed when the satellite is in range for the entire arc of the overpass. Data exchange is only possible if the antenna is continually tracking the satellite. When sending or receiving data at a certain frequency, it is also necessary to calculate and account for the doppler shift that occurs since the spacecraft is in relative motion compared to the ground station.

The process of predicting the flight trajectory of a spacecraft is called propagation. In this work the term propagation is also used to refer to the calculating of the overpass window. For the case that multiple overpasses overlap it is important to determine which one should be prioritized. This process of determining a schedule which dictates the exact satellite which will be tracked at what point in time is called scheduling. In the initial stages of spaceflight scheduling was usually done by hand, however today it is a deeply studied topic with a variety of algorithms to solve this issue. [63] [21]

The utilization of multiple ground stations in conjunction can serve to augment their collective efficiency. Such a system of ground stations is called a Ground Station Network (GSN). It opens up the possibility of contacting satellites remotely, utilizing another ground station. This significantly expands the time intervals in which effective communication can occur and greatly benefits all participants of the network. However, this also imposes great difficulty for the scheduling process if the goal is to find a perfect solution for all ground stations.

The field of scheduling can be separated into Single and Multi Resource Range Scheduling. Multi Resource Range Scheduling handles scheduling multiple satellites with multiple ground stations, while Single Resource Range Scheduling only targets one ground station. It is important to note that Multi Resource Range Scheduling is classified as NP hard and there is no known algorithm which is able to find the optimal solution in an affordable amount of time. [21, p. 1-2]

One of the most important questions for GSNs is the overall design of the system. Central factors include ground station hardware, communication channels between stations as well as the overall software design. Finding an optimized solution depends heavily on the use case of the developed network, a singular, universally superior solution does not exist. This is the reason why there is a wide variety of GSNs which have been established for vastly different use cases. The study at hand aims to find an appropriate solution for its own specialized use case, by classifying and evaluating existing GSNs. The full analysis can be found in chapter 3.

2.3 Software Defined Radios

Radio Systems offered the opportunity to directly transfer data over a long distance. The ability to send an encoded signal over the air enabled the creation of the first mobile devices like phones. This process can be separated into two stages: analogue and digital. The signal provided by the antenna is analogue, however a computer is only able to handle a digital format. This conversion is done by an Analog Digital Converter (ADC), which samples the analogue signal and converts it into bytes. This needs to happen vice verca for the The most important attribute of an ADC is its sampling rate. With a higher sampling rate it is possible to process signals with a higher frequency. When radio systems were first developed, sampling rates were too low to support the high frequencies required by antennas. To deal with this issue analogue components where used to process the signal, so the data could be handled by the ADC later in the chain. As sampling frequencies increased converters moved closer to the antenna and more processing was done by the computer itself. [60, p. 9-12]

This offered a wide variety of opportunities. For example the process of encoding bytes onto a carrier wave, which was usually done by analogue components, can now be done by the computer digitally. This process is called modulation and is further elaborated in subsection 2.3.1. A Software Defined Radio (SDR) is defined as a communication system where components that have been traditionally implemented in analog hardware are implemented in software instead. Handling this process digitally offers two main advantages: flexibility and ease of adaptation. SDRs are able to easily change the way how the data is modulated. They also enable to easily switch channels or switch between different modes of operation. New behavior can easily be developed and distributed via software. If there is a need to reconfigure the radio system, this can be easily done as well, taking significantly less time than in a traditional radio systems. [49] [22]

All of these factors led to SDRs becoming the state of the art when it comes to Radio Frequency (RF) signal processing. Their adaptability makes them perfect for reconfigurable communication links in GSNs as subject of this work. GSNs require a wide range of frequency and modulation types, since different satellites require different communication settings. Since the targeted satellite and therefor the communication parameters are dictated by the software, it is nearly necessary to use SDRs, to achieve this kind of behavior. Partners of TIM also conform to this state of the art and expect to work with SDRs, which makes them an integral part bound to this use case.

2.3.1 Modulation

Modulation describes the process of encoding a baseband signal onto a carrier wave. The wave is able to transport and convey this signal onto a receiving target. The wave can then be demodulated and the signal is restored. The topic of modulation is a widely studied subject in field of communi-

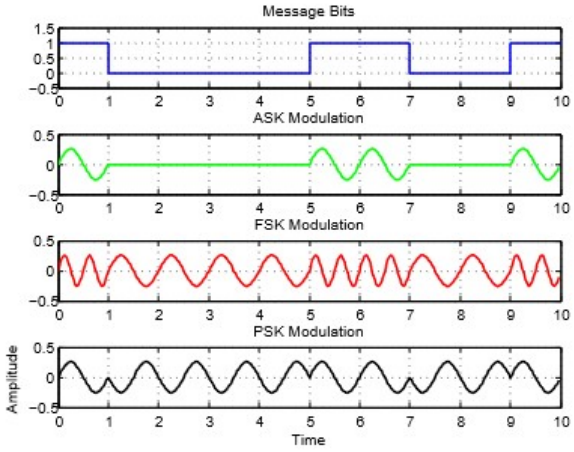


Figure 2.6: Basic types of modulation [22, p. 3]

cation. A wave has three basic parameters that can be modified, which in turn result in three basic modulation approaches. The parameters are the amplitude, frequency and the phase. The resulting approaches are named Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK). Their behavior is visualized in this graphic: 2.6. More detailed information is available in [22].

2.3.2 GNU Radio

GNU Radio is an open source tool that can be used to create SDRs. It organizes signal processing primitives into blocks that provide a single functionality. Blocks can have in- and outputs for different data types and edges that connect them to other blocks. Edges represents the data flow that defines the behavior of the created SDR. GNU Radio graphs are created using a visual editor and are then compiled to either c++ or python, which can then be executed by the computer to run the created graph. Due to its open source approach there are a variety of libraries which extend the behavior of GNU Radio. New blocks that provide new features can be easily implemented using the Application Programming Interface (API) and can be shared with the community. This open source approach makes GNU Radio one of the most powerful tools for creating SDRs. [51, p. 2]

GNU Radio is used in this work to create a SDR that handles the incoming and outgoing data. Open source extensions for modulation blocks already exist for the most common types, which enables us to support different types simultaneous and dynamically switch between them. GNU Radio also offers the opportunity to easily insert and extract data out of the running system, using for example the out of the box options of audio streams or sockets (see 2.4.2).

2.4 Software Interfaces and Concepts

This section introduces key concepts on the technical side which are of importance for the implementation for this work. Use cases for applying these concepts in the TIM GSN are established and elaborated.

2.4.1 Protocols

There are two protocol groups with high significance to this work that are covered in this subsection. Since both protocol groups are used for packet switched networks like the internet it is important to establish this concept first. Communication over networks is usually defined by small bursts of data traffic. To optimize efficiency, early telecommunication networks utilized circuit switching. For a communication to occur the network would establish a dedicated electrical circuit, which would be used to transfer data directly until it is closed and utilized differently later. This concept was expanded upon to establish packed switched networks, which work by separating data into packets which contain information about the destination where they should be sent. Utilizing this information, the network subsequently directs the packet towards its intended destination.

The first protocol developed for this use case is called X.25. It works similar to old circuits by first sending a request packet with a target address, which establishes a connection throughout the network. A connection is identified by its address as well as its logical channel which makes multiple connections from the same address possible. Once this connection is established and validated using a control packet, data can be sent though directly providing only the logical channel. The network takes care of routing the packet directly to the established target using the addresses provided by the request packet. X.25 is therefor an end to end protocol. [5] [13]

However, this approach is not particularly efficient. For a connection to be established a router in the network is required to remember every active connection including its source and target addresses. This works well for smaller networks but quickly takes up enormous amounts of RAM when scaling the network. X.25 is therefor not used in many places and is considered a legacy protocol. [20, p. 2]

One variant of extreme importance derived from X.25 is the Amateur X.25 or AX.25 framing protocol. It is the legacy standard which emerged from the packet radio operation of amateur radio operators and is widely used within the amateur radio community. The protocol was optimized to send data using error prone communication channels like for example VHF or UHF. AX.25 introduces a check to detect if the transmitted data contains errors introduced during its transmission. AX.25 was proven to be a reliable protocol for transmitting data between ground stations and satellites which is the reason why it is widely used to this day. It carries significant importance for this works, since it is not only used by many existing academic GSNs, but is also used by the projects which this work aims to support. [33] [4]

While AX.25 was rather successful X.25 was replaced by a more efficient protocol stack, which is the backbone of the modern internet. The TCP/IP protocol group solved many issues which X.25 was not equipped to handle. IP removes the need for dedicated communication channels to be established. Every sent packet contains a header which consists of all information required for the packet to be routed through the network to its destination. The most important information are the Source Address as well as the Destination Address. Addresses as defined by the IP protocol are called IP addresses and follow a specific pattern which is beneficial for routing the package through the network. [20]

This alone makes it possible for data to be dynamically sent through the network, however modern networks still require dedicated end to end communication to communicate larger amounts of data. This is handled by the TCP protocol which “provides a virtual circuit (connection-oriented) across a

network" [20, p. 19]. TCP makes it possible to map packets to a session using the IP address and the port of the source. This means that packets can take a dynamic route over the network without blocking a specific path like X.25, while still being part of a virtual circuit. It also means that routers only have to handle sent packets, they never have to establish sessions or remember them. This is entirely handled by the two end points. TCP describes how these virtual circuits are established and closed using IP packets. It also provides the ability for error correction, to ensure proper data flow. [20]

2.4.2 Sockets

Sockets are an universal concept in network programming. They have a corresponding implementation in nearly every language and connect software applications of all types. A socket can be defined as an endpoint for inter-process communication flow across a computer network. [18]

Sockets were developed on top of TCP and IP protocols as an API Layer to make using these protocols easier. Sockets work by separating between server and client. The server is a program that provides some sort of service, while a client socket refers to a user that is using this service. Multiple client sockets can connect to a server socket, which is the backbone of this network architecture. A socket is defined by an IP-Address as well as a port number, which is used by another socket to connect. Once the connection is established data can be sent from one socket to the other in both directions. The data transmission utilizes TCP and IP as mentioned earlier. This process is hidden behind the socket and handled by the socket, which makes this API very easy to use for programmers. [18]

The TIM GSN utilizes sockets to connect the running GNU Radio graph to the rest of the application and to easily send data between the running python script and the rest of the system which is implemented in Java. The wide spread usage of sockets makes them one of the easiest options to achieve this kind of behavior. There are socket implementations for GNU Radio and the underlying Python script as well as Java. Once the connection is established the incoming bytes can be easily handled and processed in Java.

2.4.3 SIDS

Simple Downlink Share Convention (SiDS) describes a public interface that enables radio amateurs all over the world to forward received packages to the ZfT. It is the main interface that is used by the TIM GSN to submit received data to the respective endpoint. SiDS describes an interface in the form of a web server that handles incoming requests. Data is submitted by sending a defined web request to this server, which has to include parameters regarding the received packages. The most important parameters are an identifier of the spacecraft as well as the receiver, a timestamp, the received data which is an AX.25 Packet encoded in hexadecimal, as well as the location where the package was received. The SiDS standard does not enforce an implementation of the web server, but provides an example in PHP. This enables partners to easily setup their own SiDS end point, but also provides the flexibility to implement custom package handling. [11]

2.4.4 Java Remote Method Invocation

Remote Method Invocation (RMI) describes a technology which is part of the Java ecosystem. With this system it is possible to invoke methods from other Java Virtual Machines. This is done using Remote Procedure Call (RPC) a concept that was first presented by Birrel and Nielson in 1984. RPC proposes a system where the complexity of network programming is hidden behind what looks like a local method call. For this to work the client calls a proxy method which forwards the call to the server in form of a request and returns the result provided by the server. This proxy method looks exactly like the normal method which makes the usage for the programmer really comfortable. RMI for Java takes this concept a little bit further. Since Java is a heavily object oriented language it uses objects instead of methods. Java RMI provides the ability to share an object over the network. The client who wants to use this object uses a proxy object which handles the networking behavior as described for RPC. This makes it uncomplicated to host and share objects over the network. [6] [18]

As explained later in 4.2, the architecture of the proposed system is separated into multiple micro services. RMI provides the ability to host and reference another micro services and their exposed behavior. Even though it is possible to host the objects over the network, we only host them locally for other micro services. This enables a modular system design with multiple components that work independent from each other. This design choice makes the existing system adaptable as well as extendable and also provides a clean and separated architecture for the whole system, where different tasks are split into respective micro services.

2.5 Ground Station Network Use Case

The TIM GSN is developed with an exact project and use case in mind. The established network incorporates the preexisting partners of the TIM project and support the existing satellites which are already in orbit, as well as upcoming missions. Since participants are mainly in the frame of the RLS-Sciences Small Satellites group, the network is of academic nature.

2.5.1 TIM Project

“The Telematic International Mission is a multinational effort to combine multiple Nano-Satellite missions in a large formation aiming at different remote sensing applications. In TIM, project partners contribute with their own satellite formations as well as ground infrastructure. Members from the Regional Leaders Summit (RLS), which is a multilateral forum of seven partner states from five continents cooperate under the technical leadership of ZfT.” [24] Figure 2.7 shows the actors and their respective ground station locations, which are part of the TIM project. The terms TIM partner, project partners and actors is used synonymous in this work to refer to the institutions, which participate in the TIM project, as displayed in this figure. If all actors were to utilize the TIM GSN they would greatly increase their communication rate, due to the relatively even global distribution of participants, resulting in great coverage in a variety of regions. This characteristic makes the project highly appealing to participating institutions.

Figure 2.7: Overview of the TIM GSN
[23, p. 2]

Due to the fact that all TIM partners already operate their own ground station, each partner has different preexisting hardware and software as well as an overall different architecture of the whole ground station system. The GSN software needs to be flexible enough to support all architecture variations of TIM partners and incorporate them into the network.

The most important constrain is the legal liability of the network. Due to constraints enforced by the International Telecommunication Union, the International Amateur Radio Union as well as the German Bundesnetzagentur, it is generally not possible to control hardware remotely over the network. Additionally this would make the ZfT liable for potential damages which the antenna could cause. It is therefor an important constrain that the control over ground station equipment remains in the hands of the local operator. In this case, the operator holds ultimate responsibility and has the final decision-making authority in regards to any forms of communication. This is a prerequisite for any system used by TIM participants.

2.5.2 TOM Project

TOM is the German contribution to TIM which is developed by the ZfT. The goal of this project is the Stereo-Photogrammetric observation of ash clouds from volcanoes. With this data it is possible to assess the height of an ash cloud and make better predictions about the spread of volcanic material. Figure 2.8 shows an example of Stereo-Photogrammetry which was conducted using images captured by the International Space Station (ISS). This serves as a proof of concept, however the accuracy of TOM will be even higher. Since the ISS is the only source of images the images are captured in a timed interval. During this time the ISS moves along its orbit and captures images from different angles. Since time needs to pass for the angle to change each picture captures a different stage of the ash cloud. Due to the fact that they move with speeds of up to 100 km/h or more, this can negatively impact the result. Another issue is the linearity of image positions. Since the ISS travels along an arc the images are all aligned one axis. This causes the result to be “one dimensional”, along the axis the images were taken, which means that some details might not be visible. The ideal scenario would be to have multiple images taken from different angles at the same time. TOM tries to solve this issue by deploying a formation of three CubeSats deployed in a triple pendulum formation (see figure 2.9). Three CubeSats equipped with cameras will cooperate to take multiple images from

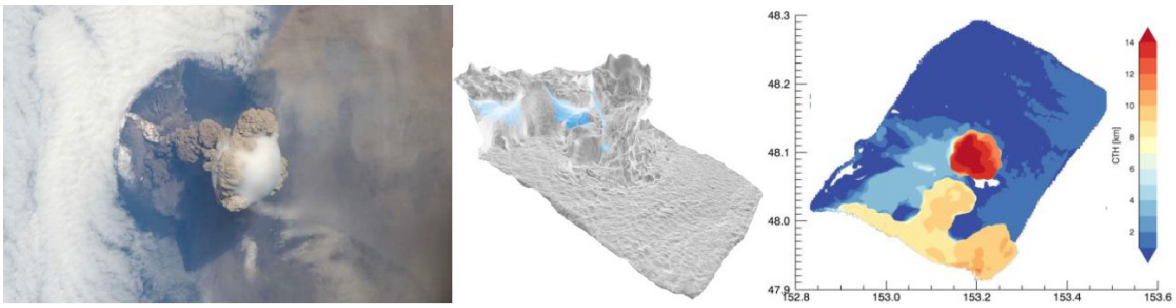


Figure 2.8: Example for Stereo-Photogrammetry as planned for TOM [50, p. 5]

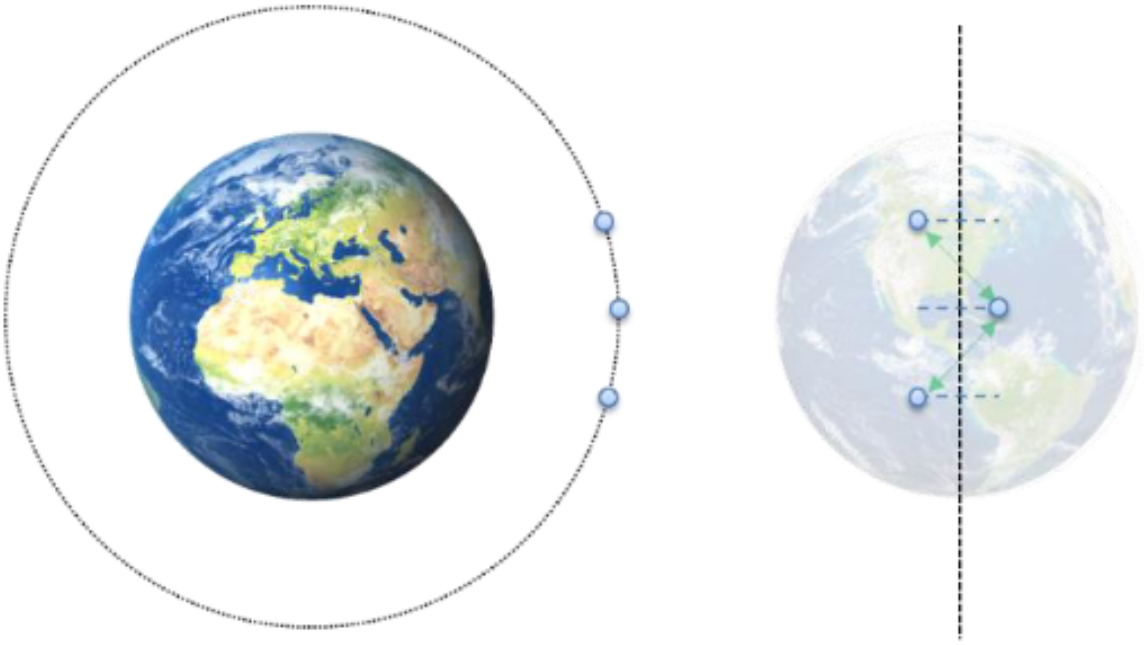


Figure 2.9: Triple Pendulum Formation for TOM [50, p. 6]

different angles of the same cloud at the same time. This requires a great amount of communication and coordination. All satellites have to select and aim their camera at the same target and take a synchronized image. This task is subject to multiple research questions and papers at the ZfT. [50] [64]

TOM as well as most other mission requires the operator to upload new versions of software or trigger specific functionality on the satellite. It is necessary for the GSN to possess this upstream capability in order for the system to be effectively used. This feature is crucial and must be taken into consideration as a constraint during the evaluation process of preexisting networks.

2.6 Research Goal

As established, multiple ground stations shall be joined together into a worldwide network, in order to increase communication time with small satellites in LEO. This poses multiple challenges which can be overcome in a variety of ways. The study at hand ignores the issue of scheduling and data

transmission. Scheduling is magnitudes less important than the overall design of the network. Data transmission needs to use the architecture of the internet, since TIM as an academic project is not able to afford custom dedicated data channels.

This work elaborates existing solutions in the context of TIM to derive an optimal GSN for this use case. The system must abide by several constraints as established in this chapter. Each of these constrain can be considered a kill criteria, if a GSN is not able to abide to one of the criteria it is not fit to support this use case.

1. **Accessible:** TIM partners need to be able to join the system.
2. **Upstream:** The system needs to support upstream in order to successfully support TIM missions. It is not sufficient for the system to only support data collection.
3. **Hardware Flexibility:** The system needs to support the varying ground station architectures. This presupposes a high flexibility and adaptability of the system as well as highly modular code.
4. **Legal Liability:** Control remains in the hands of the operator.

The next chapter introduces relevant GSN and elaborate how the respective network was managed and different hardware was supported. It features an analysis which demonstrates if and to what extend existing ground station networks are able to support the established criteria.

3 State of the Art

The history of GSNs is rich and defined by a wide variety of networks covering different use cases and approaches. With the first satellite Sputnik launched in 1957 by the Russians, the race for space started [32]. Following this first launch were numerous satellites and research projects by the United States of America (USA) and Russia alike. This dispute saw the start of one of the first GSNs, which is still relevant to this day, the Deep Space Network created by NASA. What started as a single antenna in the desert, will later evolve into of the most relevant networks discussed in this chapter [35, p. 3]. While the deep space network was always a relevant representative of a successful ground station network, it underwent a wide range of changes, adaptations and improvements with advances in satellite technology. This technology driven change affected the entire landscape of GSNs and introduced new actors and use cases for satellites and ground station networks alike. The current state of spaceflight and GSNs is defined by a large count of independent networks which aim at different applications using varying approaches. The following part of this chapter presents all GSNs relevant to this work. They will be separated and categorized using two distinct attributes: accessibility and system design. Each attribute is covered in its respective following section.

3.1 Accessibility

The attribute accessibility describes how limited the audience is which is able to use the GSN. There are three distinct categories which all correlate to a specific use case. Private describes networks which access is entirely limited, academic describes networks which are focused on collaboration and open to partner groups while service driven is the most open network, accessible for everyone.

The following subsections will outline the most prevalent representations, along with their function and typical use case.

3.1.1 Private

Private GSN are defined by only granting limited access to a small user group, typically from a company or an organization level. They are maintained and managed by this one institution and display a tendency to focus on a specific mission or goal, like for example high stake research or military advantages. This type of network marks the origin of GSNs, since in the early days of spaceflight satellite technology and research was highly expensive. Only state institutions had sufficient funding to create entire networks of ground stations. Private networks are defined by being highly controlled. Since they lead high stake operations ground station hardware is usually controlled directly, and highly expensive custom antennas are used to provide the best communication possible. They strive for providing the best quality available.

There are three relevant GSNs which will be examined in this subsection, the official large scale networks by the USA, Europe and France respectively.

Figure 3.1: The NASA Ground Station Network [37]

NASA Network

NASAs ground network in fact consists of multiple different networks. It includes an Earth Relay Element also refereed to as the Space Network, a Deep Space Element, referred to as Deep Space Network, and the Near Earth Element, referred to as Near Earth Network. Figure 3.1 provides an overview of the entire network and displays the locations of each ground station. It is important to note that Near Space Network comprises both the Near Earth Network as well as the Space Network. This analysis will mostly focus on the Deep Space Network since it is the most prominent one of the three.

As already mentioned the Deep Space Network by NASA was one of the first GSNs ever created. Since its creation in 1957 it remained of uttermost relevance in the field of spaceflight and is today considered “a world leader in the development of deep space communications and navigation” [40]. The Deep Space Network enabled multiple space missions with historical significance, for example the first moon landing [35, p. 56-58], Voyager 1 and 2 [35, p. 83] “one of the most ambitious planetary missions ever undertaken” [35, p. 107] or one of the most recent scientific breakthroughs, the James Webb Space Telescope. As the name suggest the network is focused on providing communications access over incredibly large distances far outside our solar system. For example Voyager 1 is currently (28.02.2022) 23,84 billion km far away from earth with active communication still happening [39], compared to LEO with a height between 400 and 1500km. These kind of missions require the highest quality of hardware, as well as high levels of control. They need to be incredibly reliable due to the high stakes these operations provide. The huge distances require specialized high performance antennas as well as receivers and transmitters. Such devices are extremely costly not only to acquire but also to operate and maintain. This stands in contrast to the majority of academic and service-oriented networks, in which the requirement for high performance devices is generally not present, allowing for the implementation of simpler designs. As established in figure 3.1, the network consists

Figure 3.2: The ESTRACK Ground Station Network [12]

of three strategically placed large ground stations. The stations are controlled using a principle called Follow-the-Sun Operations. Each station is responsible for remote controlling the entire network during their respective 8 hour local day time shift. 1 extra hour is used to pass on control. [28]

ESA ESTRACK

The ESTRACK system was developed by the ESA. The network as depicted in Figure 3.2 is comprised of three distinct parts. The Core Network features ground stations owned and operated by ESA. These ground stations run the ESTRACK stack which will be elaborated in this section. The network is extended by the collaborative network consisting of external entities which provide regular support as well as the augmented network which consists of commercial stations operated on behalf of ESA. The core network is designed so it can be both controlled remotely and locally, however remote control was much more prominent. All core ground stations are controlled by the Operations Control Center which acts as centralized control point for the network. The system features a unique solution to dealing with differences in ground station architecture. The operation subsystem provides the option to monitor and control the ground station. This behavior is customized by the tailoring subsystem which allows unique tuning to fit the specific hardware of the ground station. The system directly controls the specific hardware of the ground station remotely. [29]

CNES Ground Network

The CNES Ground Network is a french GSN that supports french national satellite. Ground stations are directly controlled using a generalized interface that includes ground station information and functions. To achieve this a redundant hardware architecture is used. Even though there are local differences it is possible to run all ground stations using the same software, and to access all ground stations using the same interface. [56, p. 6]

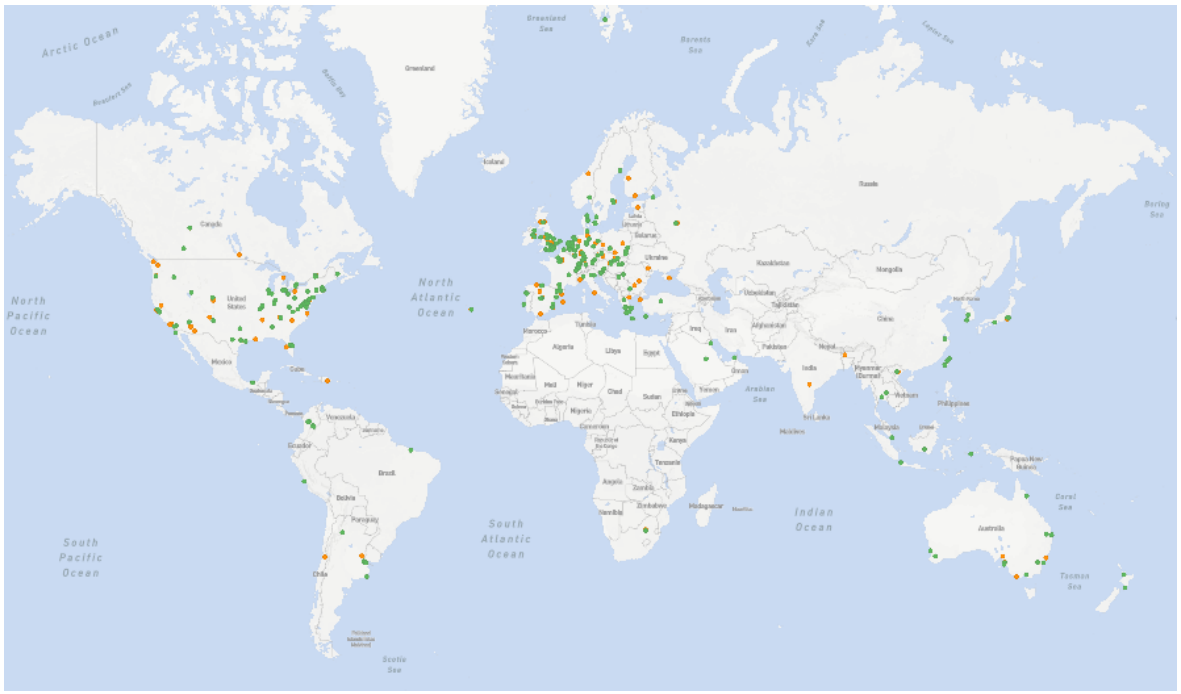


Figure 3.3: The SATNOGS Ground Station Network [12]

3.1.2 Academic

As outlined in section 2.1, after the small satellite doldrums there was a major up-rise in academic spaceflight. Academia is characterized by the overly high usage of COTS components for spacecrafts and ground stations alike [52, p. 1, 19]. It is based on the fundamental principles of cooperation and scientific exchange, which is reflected in academic GSNs. They are used by a variety of actors which act as equals, with the focus on collaboration to support partners by sharing resources. They are accessible to the public or project partners free of charge, that means anyone (or selected partners) can join the network easily and the option to expand the network to more and more stations is considered in the design of the system. Academic ground stations also need to accounts for huge variations in ground station hardware due to the fact that there is no overlying hardware standard compared to private or service oriented networks.

SATNOGS

SATNOGS is by far the most important academic network of its time. It is an open source ground station network that provides a crowdsourced approach to connecting users and satellites. It is possible for every user to establish an existing ground station. Figure 3.3 shows all ground stations which are currently part of this continually growing network. The ground station uses the software provided by SATNOGS, which controls the rotation and frequency of the antenna using standardized protocols. The design uses GNU Radio for its signal processing and features adjustable tuning, translation, demodulation, and decoding for the incoming data. It is important to note that SATNOGS only handles down stream. It functions more as a collection and logging system for received data. It does not feature the ability to upstream commands or data to satellites which are currently over passing. Besides creating a ground station it it is also possible to schedule the tracking of specific satellites with customization options for the used modulation and carried data. Each ground station

is managed by the individual who build and maintains it. Received satellite data is logged and the owner of the satellite is responsible for downloading and dealing with it. The network is based around the autonomy of ground stations and collaborative effort. [8] [41]

GENSO

Global Educational Network for Satellite Operations (GENSO) was an academia focused ground station network largely funded by ESA and developed by students of multiple universities. The project was started in 2006 and featured an extensive road map. Planned features include remote up and downstream, integration of an internet relay chat client as well as opening the project up to the public as open source, with anyone being able to join. The software controls the hardware directly using standardized interfaces as well as an abstraction layer to provide the ability to implement own drivers to interact with unsupported hardware. The software provides an interface that is used to view satellites and their respective overpasses. The software can then be used to contact the corresponding ground station to book the pass. The system was designed to enable cooperation between self managed ground stations, however it was discontinued without any apparent explanation and was never publicly released as open source. [25] [30]

UNISEC GSN

The UNISEC GSN was a GSN developed by Japanese students. Due to the fact that it was only used in japan there are only limited non Japanese resources available regarding its functionality or current state. It can be said that it is focused on web based remote control of other ground stations. The hardware of the remotely accessed ground station is directly controlled by a locally running software provided by the network stack. Unlike traditional private ground station networks, UNISEC GSN was one of the first networks to use the internet for communication between applications instead of dedicated private channels. This approach has allowed UNISEC to leverage existing internet based technology stacks for all of its service points, which are web-based applications. Differences in ground station hardware are offloaded to the station owner by providing a protocol which can be implemented by any software solution to incorporate ground stations into the system. It also is open to all organizations which want to join the system. It focuses on an easy-to-understand framework which can be used by anyone.[19] [53, p. 3] [36]

3.1.3 Service Driven

With the latest trend of commercial spaceflight being more prominent than ever as outlined in section 2.1, there is a growing need for cost effective ground station networks available to businesses. Service driven GSN are defined by providing access to everyone with the sufficient funds. They share similarities with private networks as they are highly controlled and managed by a single entity which provides the service, how ever they are focused on expanding and scaling their network to as much customers and ground stations as possible.

Cloud based networks

While there are multiple different solutions that exist on the market today, they all follow a similar formula. The considered service driven networks are AWS Ground Station by Amazon Web Service (AWS), SSC Satellite Ground Station Services by the Swedish Space Cooperation (SSC), KSAT a norwegian company providing a variety of services and LeafLine by LeafSpace an Italian company. While there is not much scientific information available regarding specific implementation details, they all share the same systematic approach to their system. The architecture features multiple ground stations which are maintained by the company and most of the important features are handled using a cloud. This means that the end user always interacts with the cloud and the system manages everything for him in the background. Scheduling is usually done in the cloud as well. These type of networks will be refereed to as cloud based networks in this work. [58] [3] [26] [57]

3.2 System design

There are two available extremes to this problem statement. One possible solution would be to use one central server or point as described in the last chapter which is responsible for every action of the system. Another approach would be to spread computation and responsibilities out to individual ground stations or services. In this case there would be a small to no central system, while everything happens peer to peer between the elements of the system. These extremes can be seen as two end points of an axis describing how centralized or distributed the system is. There exist a multitude of systems which can be situated along the continuum of this axis. Both have their own unique advantages an disadvantages covered in the following sections. It is important to note that these advantages or disadvantages are highly dependent on their use-case, which was covered in the last chapter. This section will also assess the established kill criteria to determine if the network can be utilized for TIM.

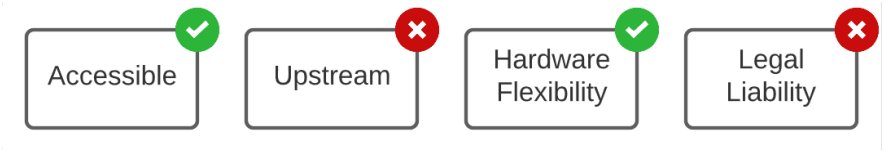
3.2.1 Centralized

A centralized system features a strong central unit which is responsible for handling close to all required tasks of the system. This might include propagating and scheduling overpasses, archiving data for future use, interpreting downloaded data, making it available in an organized fashion, etc [25, p. 4]. It needs to have sufficient computation power to manage this workload for it to be able to provide the entirety of this functionality. In the most extreme case this central unit not only computes the schedule for all ground stations, but also translates this schedule into precise commands regarding hardware which are remotely executed at the ground station. However, this type of direct hardware control is not particularly prevalent.

SATNOGS

SATNOGS is a prominent example of a centralized network. Even though it is an open source crown funded network it relies on one central system for its core functionality. The SATNOGS Network manages overpasses for all ground stations. It computes possible time windows and provides the option for users to schedule observations. Ground stations then only need to poll the observation job

queue which is then handled by a ground station scheduler. SATNOGS is able to support different rotators and signal receptors using standardized protocols which are driven by the active observation job. Collected data is then sent back to the SATNOGS Network and can be accessed by the user using an id. All of this functionality is accessed by the user using a web interface, but there is an API which provides the ability to fetch the data using web requests as well. The central server is hosted by the Libre Space Foundation, the non profit organization behind SATNOGS. [8] [41]

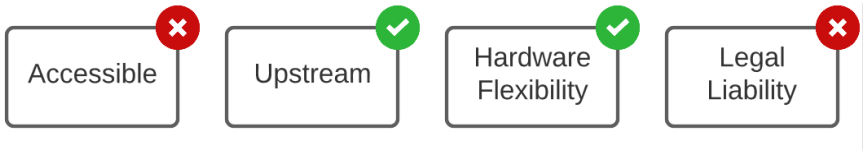


Evaluation SATNOGS

It would be technically possible for TIM partners to join the SATNOGS network, utilizing the standardized protocols. However these protocols aim to control hardware directly, which violates the legal constraint. It would furthermore not be feasible, since SATNOGS also does not support Upstream.

ESTRACK

As outlined before the ESTRACK system features a central control unit which is used to control all ground stations directly, the European Space Operations Centre. This center is a physical location with different sections responsible for managing different functions of a GSN. The scheduling was done using a manual planning and scheduling tool, however it was enhanced by the ESTRACK Management System, which is used for 90 % of missions today and automates this process. The schedule is then executed using the remote access feature of ground stations. The automated pipeline consists of the scheduling triggering, which executes scripts which interact with the ground station hardware. The schedule contains overpasses as well as to be up streamed data. [14] [29]

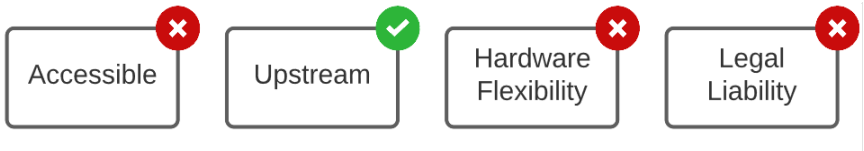


Evaluation ESTRACK

Even though ESTRACK supports upstream and features a powerful tailoring subsystem to support a variety of hardware configurations, there are the dominant issues which make ESTRACK as a system unable to be used. Firstly is the central control unit, which is not fitting for an academic ground station network. The direct control of ground station equipment by this central unit also directly violates the legal constraint. Lastly the network is closed and not open for organizations to join. ESTRACK is not a viable solution for participants of TIM. [29]

CNES Ground Network

The CNES Ground Network architecture features a system called Icare. It is a central system operated by the CNES Network Operations Center. Icare provides the ability for control centers to directly monitor and control ground station equipment. This is done by establishing a secure connection to the ground station. The controlling entity is then able to interface with ground stations which are abstracted to have the same information and functions, while still allowing for flexibility in hardware architecture. Icare is responsible for secure network routing as well as providing access to all ground stations. Scheduling is managed by a second central entity, the Orbit Computation Center which is responsible for creating schedules and providing them to ground stations to be executed. With these two centralized units, the CNES GSN is a highly controlled and centralized network.[56] [54]



Evaluation CNES GSN

Even though the CNES Ground Network supports upstream it is highly unsuitable for TIM participants. It is based on redundant hardware architecture, which causes it to fall short in terms of needed flexibility [56, p. 6]. Furthermore it is closed off and features a centralized management unit similar to ESTRACK. Lastly it also fails to abide to legal constraints since it controls ground station equipment directly. [56] [54]

Cloud based networks

Cloud based networks as previously outlined also fall into this category. The cloud acts as a single unit which handles all tasks. It provides access to the system for users and handles important tasks like propagation and scheduling for all ground stations of the network. This approach works greatly for cloud based networks since it allows them to easily scale to a large number of customers with only scaling the performance or capability of this one system.

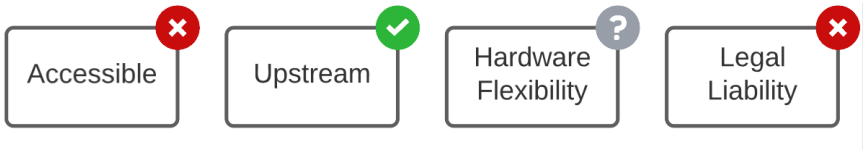
Evaluation cloud based networks GSN

Since cloud based networks can only be joined by payment, they are not an option for TIM. The accessibility would furthermore only regard utilizing the up and downstream options, but not incorporating the ground station into the network. While there are some cloud based networks which include partner stations into their network, there are several factors which make this approach fail.

Since there is only limited information available it can only be speculated if the respective software stacks are flexible enough while also not violating the legal constraint. Cloud based networks are also service oriented and thereby commercially oriented, which is fundamentally misaligned with the values of TIM, a non profit collaborative project. Furthermore the systems feature a centralized approach for managing the system in form of the cloud, which is also incompatible with TIM partners.

NASA Network

Event though the Deep Space Network is controlled by the three respective ground stations individually, it can not be considered a distributed system. The Deep Space Network is always considered one system with three stations. Important features are centralized, for example all collected data is forwarded to the Jet Propulsion Laboratory, the Laboratory responsible for running the Deep Space Network. Scheduling is further more done by including all stations and antennas not individually by each station. The Deep Space Network is therefor a centralized system with minor distributed elements.



Evaluation NASA networks GSN

While NASAs network does support Upstream, like most other systems as well, it is not accessible to the public. The Deep Space Network as well as the Near Space Network also both rely heavily on direct and precise remote control of other ground stations, violating the Legal constraint. The system of passing on control is further more not applicable to TIM, since partners should remain in full control of their respective system at all time. However the Deep Space Network offers interesting approaches in regards to scheduling. It uses a system called collaborative peer-to-peer scheduling. Users of the system are able to book time slots for their respective missions multiple months in advance. Conflicts are assumed to be resolved between participants them self, however there are backup mechanisms. The schedule is locked a few weeks before execution. While it is still possible to modify time slots, it is not possible to create conflicts. Despite being a highly relevant subject for research, the Deep Space Network is not relevant to TIM. [17, p. 8]

3.2.2 Hybrid

GENSO specifically aims to be a middle solution between the two extremes. The GENSO ecosystem consists of three elements. The most important element is a central server called AUS or Authentication Server. It is used for its network authentication as well as encryption. This server is responsible for distributing satellite lists as well as monitoring ground stations, the second element. These stations are organized distributed and are responsible for fetching the satellite list from the server that contains encryption keys for data forwarding. The software establishes a connection to over passing satellites by controlling the ground station hardware directly. The last element is the Mission Control Client. It is an application which will exist once per tracked satellite. The operator of the satellite is able to overview future overpasses and book them by contacting the respective ground

station. Received data is collected by the station independently and forwarded to the client using the encryption key. The last task of the central server which involves the Mission Control Client is status and quality tracking as well es compiling network statistics. The system interestingly combines aspects of both distributed and centralized networks. [25]

Evaluation ESA GENSO

GENSO's architecture a few innovative and interesting concepts. Firstly the separation between the central server, the ground station server and the Mission Control Client works perfectly for TIM and its cooperative approach. Not only does GENSO use the same standardized protocol as SATNOGS, it also provides an additional abstraction layer above, so ground stations can implement alternatives. Sadly GENSO was never properly finished and is not operational or available in any form today. This unfinished state also means that Upstream was only planned, but never properly implemented. GENSO provides an interesting concept which might not violate the legal constraint. By sending requests for overpasses one might argue that the liability remains within the hands of the operator. There is no remote control and data is collected independently by the ground station and later forwarded. Since the upstream feature was never implemented it can only be speculated if this feature would have violated the legal constraint.

3.2.3 Distributed

Entirely distributed networks feature an architecture with a very simple or entirely without a central server. This is usually done by establishing multiple servers which each take on different tasks of the system. This might include mission based scheduling or separated data handling for different satellites. It is also important to point out what is not considered distributed for this use case. Handling Scheduling using one system but spreading out computation across multiple devices might be considered distributed in another context but for this work distribution only refers to systems with sufficient ground station autonomy.

UNISEC GSN

The UNISEC GSN serves as a valuable case in point of such a distributed approach. Figure 3.4 displays the three components of the system as well as their interactions using web requests (A, B and C). The network features a Central Server which is used to verify users and validate operation requests (A in figure 3.4). Users send their validated requests to a targeted Operation Server them self (C in figure 3.4). The Operation Server is then able to check if this request was properly validated using the Central Server (B in figure 3.4). Even though the system relies on the Central Server for its validation it works entirely peer to peer. Users send and receive data from the Operation Server directly and Operation Servers are responsible for tracking satellites autonomously. They are responsible for controlling the hardware using the implementation of the open protocol. An Operation

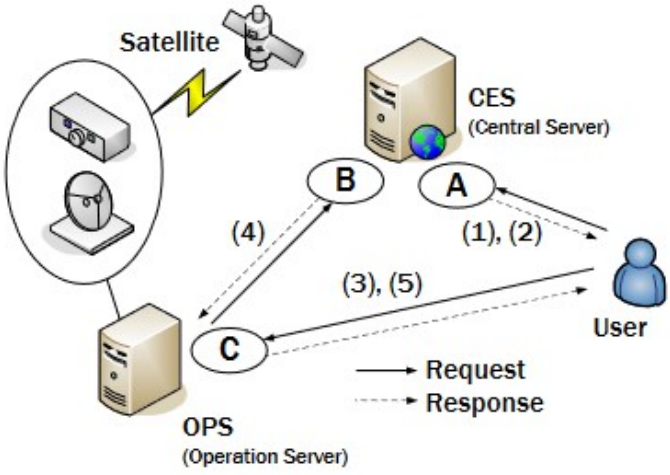
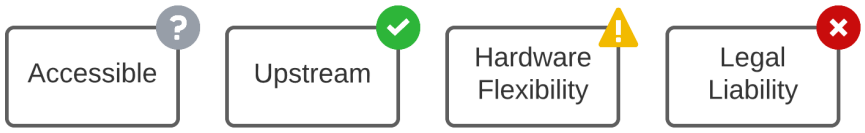


Figure 3.4: The UNISEC System [36]

Server is thereby self managed, creating a distributed peer to peer network. How ever since the system is not able to function without the Central Server it is important to note that the system can not be considered as an extreme example of a distributed network.



Evaluation UNISEC GSN

UNISEC is one of the most compatible GSN which has been evaluated in this chapter. It is technically open to all organizations which want to join, actors have great independence inside the system and it provides upstream. However the state of the project can not be properly accessed. Publications are spare with the latest published in 2012 [36]. Furthermore the project website is also not available anymore [62]. All of these factors make the availability of the system questionable at best. Another issue is that the software controls hardware directly using a custom GMS serverpackage. Not only does this violate the legal constraint, the package is also outdated and does not provide the necessary hardware support required. This makes setting up the software hard if not impossible.

3.3 Lessons learned

Figure 3.5 shows all analyzed GSN sorted based on their accessibility into the three established categories. The y axis displays how distributed or centralized a network is. Colors were used to indicate the respective accessibility category as well as system design. This overview establishes a few interesting trends. Private networks as well as service driven networks all are heavily centralized. This is due to the fact that centralized networks are easier to manage with a single point of failure. Ground stations can be controlled directly without having to worry about legal liability since there is only one organization liable. This also facilitates higher quality scheduling taking into account every ground station as for example seen in the Deep Space Network. Academic networks have a very different set of constraints. Central options are usually not applicable since it requires a central server

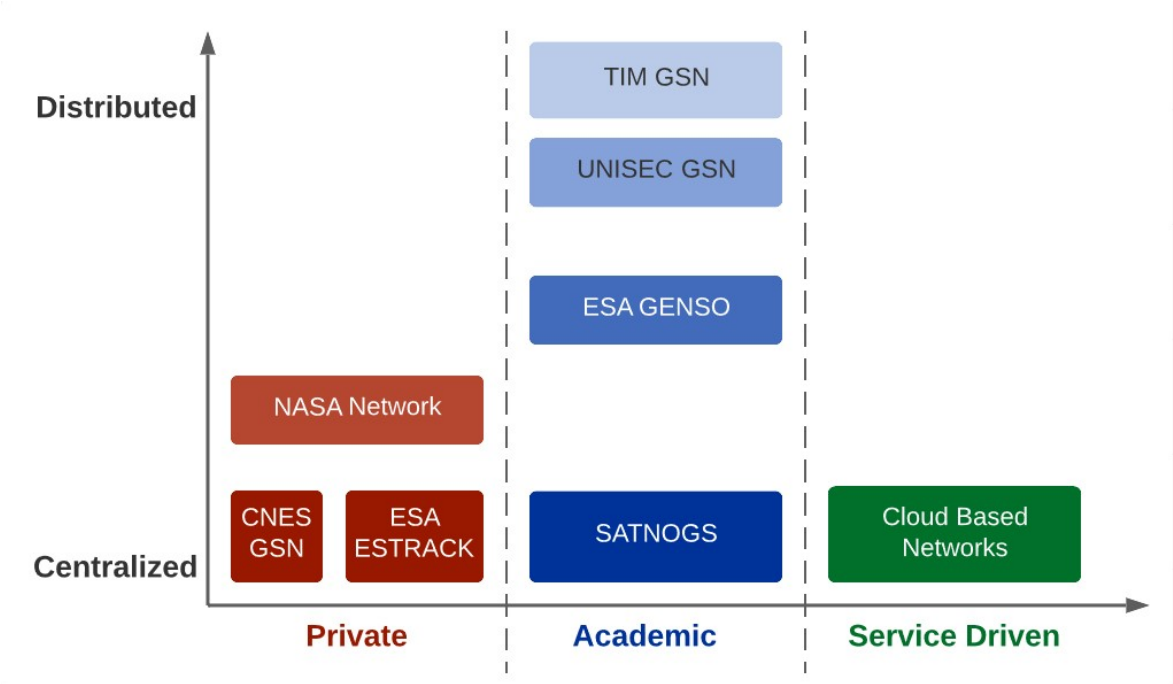


Figure 3.5: Categorization Matrix of all Ground Station Networks

	CNES GSN	ESA ESTRACK	NASA Network	SATNOGS	ESA GENSO	UNISEC GSN	Cloud Based Networks
Accessible	✗	✗	✗	✓	✗	?	✗
Upstream	✓	✓	✓	✗	⚠	✓	✓
Hardware Flexibility	✗	✓	?	✓	✓	⚠	?
Legal Liability	✗	✗	✗	✗	✓	✗	?

Figure 3.6: Constraint Overview for all Ground Station Networks

or system which is hosted by an organization trusted by all participants. This server needs to be strictly maintained since the system is not operational without it. In most scenarios it is unpractical or unlikely that this will be the case. This however does not mean that it is not possible. GENSO featured a central server which would have been hosted by ESA, while SATNOGS proved to be a working academic centralized system. The server is hosted by the non profit organization behind SATNOGS. UNISEC GSN opted for a different approach. While it featured a central server, it was especially planned to have redundant servers for this exact issue.

Figure 3.6 summarizes the four defined kill criteria and how well each network abides by them. One emerging pattern is all private networks being inaccessible as well as not worrying about legal liability. This was to be expected, since it is unnecessary for their respective use case. Another significant trend is the hardware flexibility shown by academic networks, which makes sense since they focus on allowing a wide variety of actors to join the network. The key conclusion is that none of the networks

in question are capable of meeting the specified kill criteria. This concludes that in order to full fill the goal of this work it is required to develop a new GSN which is specialized to this use case and able to abide to the constraints. This work proposes a system which fulfills this role in the next chapter.

There are several takeaways we can derive from the systems which came close to being compatible with TIM. The three systems with the least amount of violated kill criteria are ESA GENSO, UNISEC GSN as well as cloud based networks. Cloud based networks can be ignored since their application would not make much sense for TIM since they are a service and not a system aimed to be deployed. Both other systems introduce very interesting concepts, which will be used to shape the TIM GSN. One major commonality is that both systems are academic networks and distributed. The proposed system will feature a weak central server and aims to be as distributed as possible as shown in figure 3.5. This ensures maximum ground station independence similar to the UNISEC GSN. Furthermore it lifts the responsibility for the ZfT to host this server as well as ensure proper up time. The system is designed with this limitation in mind and is able to run without the central server for limited periods of time. Another significant element which will be borrowed will be the handshake established by GENSO, which takes care of the legal liability.

Lastly instead of providing the excellent generalized and abstracted protocol layers GENSO uses, TIM GSN will go one step further, by fully utilizing its use case. Since participants already have their own dedicated ground stations the software can ignore all hardware and instead focus on connecting the preexisting system into the developed network. This also takes care of legal liabilities, since there is no more direct control of hardware, not even locally. All of this is handled by the preexisting system at each ground station, TIM GSN will only provide data.

4 Proposed Solution

As established in the last chapter the network needs to abide to four kill criteria. This chapter will first derived more specialized requirements which the system should fulfill. This is done by establishing similarities between participants, to establish more concrete limitations or technological solutions which exist. The kill criteria will be integrated with the characteristics that all recognized GSN must possess in order to be considered as such. Afterwards the proposed system will be presented and its exact architecture and mechanisms will be explained.

4.1 Requirements Analysis

The developed ground station software should be able to easily be integrated into an existing ground station. The software needs to incorporate the partner station into the network, without large implementations required from the ground station operator. This necessitates the need for easy to use interfaces. Since all TIM partners have similar preexisting academic ground stations one can derive restrictions for ground stations the system should be able to support. A suitable ground station must be able to track satellites given a respective Two-Line Element (TLE), which is handled by the preexisting system. Another requirement is the need for the ground station to connect to the provided software. Two evident approaches for ensuring easy and extendable connection to the software, are using either an SDR or base band audio via a sound modem. Both approaches have been established to be utmost flexible for handling data streams between application and ground station hardware. As already established in chapter 2.6 the network will use the internet for transferring data between ground stations.

4.1.1 Definition of Scope

The TIM GSN is a software package that extends the network functionality of an existing ground station such that it is incorporated into the GSN.

4.1.2 Functional Requirements

Functional requirements cover specific features which are integral to the product. This means that they must be implemented for the product to function as intended. [2]

1. The software shall provide the ability to receive satellite data and distribute it using the internet.
2. The software shall provide the ability to send data to a satellite over the internet.
3. The software shall be able to read modulated data from an SDR or a sound modem (downstream).
4. The software shall be able to output modulated data from an SDR or a sound modem (upstream).
5. The software shall handle the propagation and scheduling of satellites and their overpasses over the ground station.

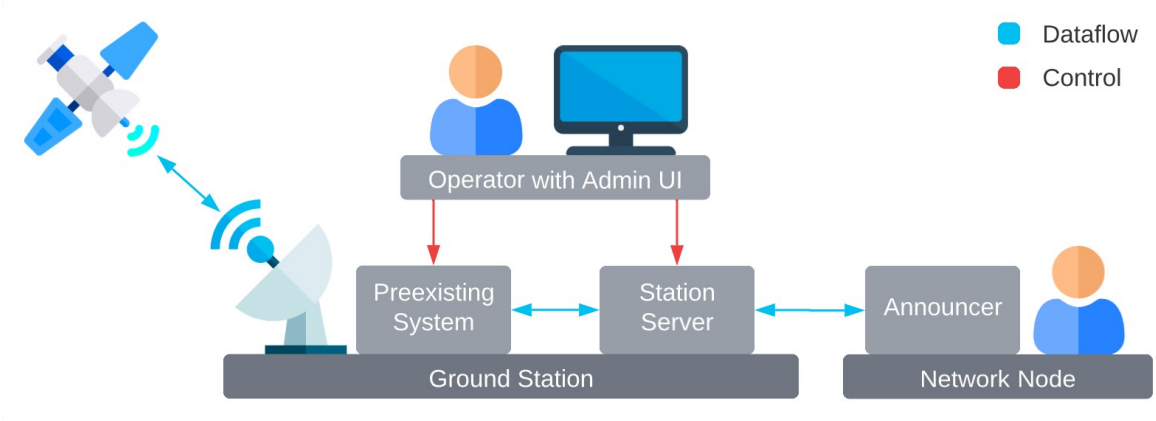


Figure 4.1: Dataflow of the TIM GSN

4.1.3 Non Functional Requirements

Non functional requirements do not refer to specific functionality. They define how the system should behave and highlight certain design principles. [2]

1. All technical knowledge and control shall remain in the hands of the operator.
2. The software shall run in real time.
3. The software should be easily portable. Logic can not be hardware depended. When porting the software it should be flexible enough so most of it can be reused without convoluted problems.
4. The software should be easy to use for the ground station operator. This includes an easy setup as well as a pleasant user experience when running.
5. The software shall be fully transparent.

4.2 Software Architecture Concept

The proposed system focuses on gathering, processing and finally forwarding the relevant data. The choice not to control specific hardware like the antenna or an Universal Software Radio Peripheral (USRP) directly, provides two significant advantages, compared to traditional systems. Since users of the TIM GSN already have their own operational systems, this use case enables the reusage of these setups. Reusing preexisting ground station soft- and hardware, makes the software flexible and easy to use for the target group, since they do not have to setup an entirely different system. Instead they only have to implement the required link between data end points of the GSN and their preexisting systems. The second advantage lies on the implementation side. Leaving the specific hardware interaction up to the user, frees the software of having to implement a wide variety of solutions for different used hardware. Different antennas might have different interfaces to interact with them, which the software would otherwise have to account for. It would also be necessary to consider different signal processing chains employed by users of the GSN, which would increase the complexity of the system as well as development time. The software can rather focus on the important aspects of a GSN, like controlling and coordinating participating stations of the network. This also takes care of legal liability and ensures that the control of station equipment always remains in the hands of the organization maintaining the station.

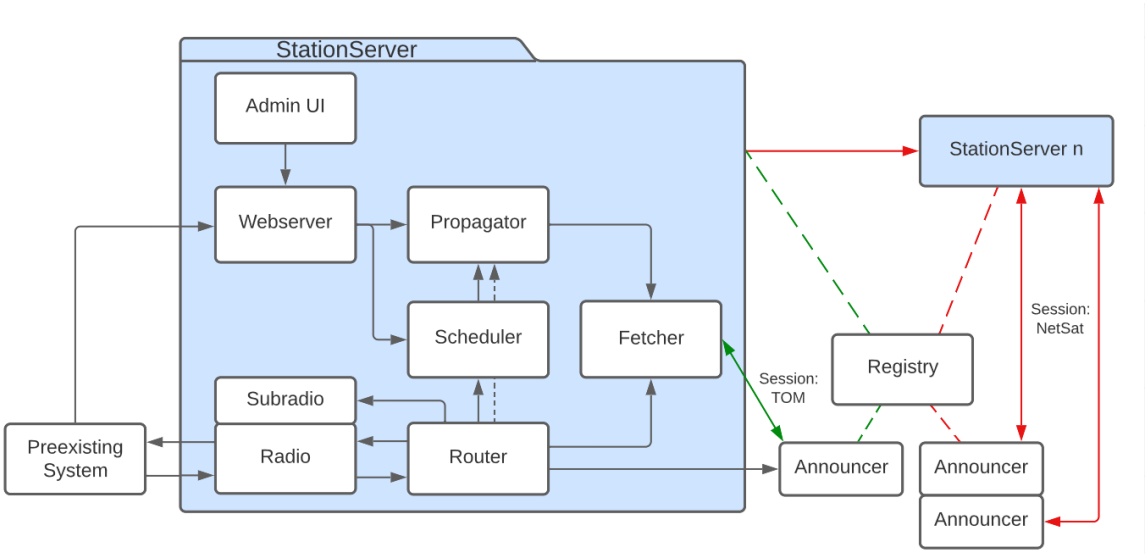


Figure 4.2: System Concept of the TIM GSN

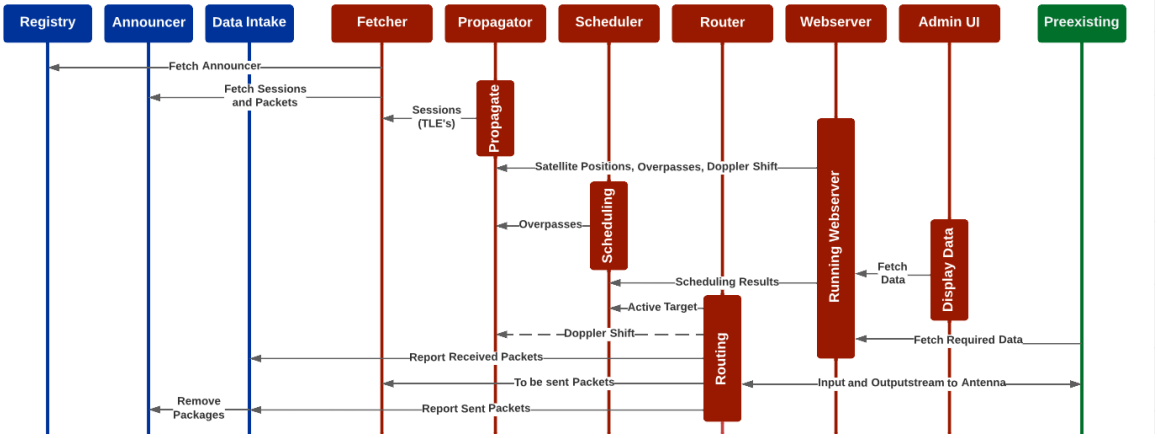


Figure 4.3: Sequence Diagram of the TIM GSN

The TIM GSN consists of two parts: the Station Server and its Network Environment it exists in. The GSN consists of multiple Station Servers which will each be located at their respective ground station as well as multiple distributed Network Nodes which make up the Network Environment. Figure 4.1 presents a general overview of the system architecture, illustrating the rough data flow of the system. 4.2 displays the inner workings of a Station Server and its interaction with the Network Environment. A more detailed sequence chart is provided in 4.3, which further visualizes the processes and dependencies inside the TIM GSN. Systems of the Station Server are colored red, elements of the Network Environment in blue, while external systems provided by the partner are marked in green.

The network provides crucial services to the Station Server and is separated into the the registry and multiple announcers. The registry is a centralized server which provides access to all announcers of the network, which are responsible for providing relevant data regarding tracked satellites. A crucial design choice is the distributed architecture of the existing announcers, which is further elaborated in section 4.2.1

The Station Server uses the Network Environment and handles the core GSN functionality. Information provided by the network is gathered and distributed in the Station Server using the fetcher. The propagator calculates positions of each satellite as well as overpasses, when the satellites will be in range of the ground station. The scheduler takes the processed data and solves the question of which satellite, out of all which are currently in range, is the most important. The final step in the pipeline, before the data reaches the preexisting system is the router. This subsystem takes all calculated data to manage out going as well as incoming data. Incoming data is sent directly to a data intake of the announcer, while outgoing data is gathered by the fetcher, before it reaches the router. All relevant data of the system is gathered by the web server and can be requested by any system using a web request. The preexisting systems uses this web server to request all required data. Another system using this web server is the admin ui. The admin ui is a web front end which displays all available information regarding the Station Server, to the operator. It also is used by the operator to approve of transmissions, which covers the essential legal liability requirement.

The following sections further establish the Network Environment and the Station Server. Any subsection of either part describes a specific sub element and is separated into two parts each. At first the concept of this certain element is elaborated while the end of the subsection covers technical implementation details which are relevant in this context.

4.2.1 Network Environment

The Network Environment comprises multiple distinct communication end points (Network Nodes), which make up the environment the software stack exists in. The network enables users of the TIM GSN to interact with it and use the functionality of the entire ground station network. This is done by providing a set of interfaces and systems which handle the incoming or outgoing data. It is also the back bone of the system and provides core functionality to the software running on each ground station. The Network Environment is a distributed system, comprised of multiple Network Nodes, which each are independent servers managed by their respective TIM partner. Since these are hosted and maintained by another entities they are open to customization, so participants can easily connect them to their own systems. They are tracked by one central server that provides access to all nodes of the network. This Server is called the registry and is only required when starting the

ground station software. Its behavior and dependency for the GSN is further elaborated in 4.2.1.1. Since the central server is not mandatory during runtime, the distributed node architecture is very prevailing. This design choice makes the system easily extendable and provides flexibility regarding downtime. If more functionality would be moved from nodes to the central server, the system would heavily depend on the availability of this central server. The main server would have to posses a high amount of computation power, to handle web requests by all actors, which would further limit scalability. A Network Node is defined as a combination of one announcer and one or many data intakes. Nodes are managed by their respective participating institution, which is the reason why the announcer and the data intake are only defined by the type of data they handle and provide or receive. The exact implementation is left to the partner who is managing this node, so they have full control over their data.

4.2.1.1 Registry

The Registry is the central server of the TIM GSN. It has the single task of keeping track of existing announcers and their URLs. The registry provides a unique identifier for each announcer, which can be used to retrieve the corresponding URL. It is organized into multiple missions which each can have their own set of announcers, handling different parts of the mission. The goal behind this system design is to minimize the amount of unnecessary tasks the registry handles. Core and especially runtime functionality should always be handled distributed by the nodes making up the Network Environment, to ensure proper availability of service. The central server still has some significant advantages. Since it has a static URL and is a constant of the system, it performs exceptionally well for tracking the nodes of the network. Other systems which require access to nodes of the system, can always refer to the registry. Handling this entry point centralized provides a great and simple solution to this otherwise complex problem. Considering that the provided access is needed to start up any Station Server, this integral part is hosted and provided by the ZfT.

4.2.1.2 Announcer

The announcer is part of a distributed node and is responsible for managing a set of sessions. A session comprises all relevant information regarding one satellite. This includes a TLE of the spacecraft, to calculate the position at any time as well as all relevant data to establish a link and communicate with the satellite. While it would be sufficient to only provide the communication frequency, the used modulation, as well as the maximum transmission power. A session also stores multiple Data Intakes (see 4.2.1.3) which has received packets forwarded to them. Parties which are interested in the satellite and its received data can add their own Data Intake to this list.

Each session can have a transmission order, which consists of one or many packets which the operator of the satellite wants to sent to the satellite of the respective session. A packet consists of payload data, as well es meta information regarding the packet. The proposed system provides special options to customize the process of data intake and output, which are further elaborated in section 4.2.2.4. An Announcer provides the functionality to add packets into the packet queue, which causes them to be sent out to the satellite if there is a connection available. The access to this feature depends on the specific implementation the node owner chooses to follow. For example node owners might connect the packet queue to internal services and add packages using them. This would also allow to only make upstream available to certified operators of the organization, however

these are restrictions which will be handled by each TIM member internally. This feature is vital, since it enables the uplink of data to the satellite and therefor the bidirectionality of the GSN. There are limitations the entire system must account for in its design when handling packets.

Packets should not be kept in the queue forever, since the focus of this upstream functionality is real time based. Packets that enter the queue should either be send instantly or get removed after a short period of time, if no connection is available. This is done by giving each packet an expiration timestamp after which it will be discarded.

If there are multiple connections available, the issue arises of two ground stations sending the same package at the same time. Packets should never be sent twice, since this violates the full transparency approach. Duplicate packets are no issue for ZfT satellites, since the transport layer protocol is able to recognize and handle them. The system should however account for different types of satellites from TIM partners, which is the reason why the TIM GSN can not expect this behavior. Packets can only be sent once, which is done by removing the packet from the queue as soon as it was sent. How exactly this works is described in the next section.

4.2.1.3 Data Intake

A session also contains a reference to one or many data intakes. A data intake is defined as a web server, which implements the SiDS specification. A ground station sends all received packets to the data intakes for the corresponding session, using the SiDS convention. If a party is interested in a certain satellites data, they are able to receive all communication, if their own data intake is added to the session. The possibility and way to do this depends on the implementation of the announcer which the respective partner chooses to follow. This system enables highly customizable data handling. For example it is possible to link multiple sessions to the same intake, since a session just stores the URL of a data intake. This would for example be useful if there is a formation separated into multiple sessions. The system is able to support single data intakes for each session as well as one which handles all at the same time. There are numerous more useful configurations which are possible in this system. The actual data processing by the intake can also be customized, because SiDS only specifies the way the data is sent and received by the intake, but not how the data is handled internally. Participants are encouraged to implement the SiDS interface them self and for example hook up this end point to their internal data bases, or other data processing applications.

4.2.2 Station Server

The Station Server is a software stack, that is responsible for connecting to the Network Environment, handling and converting the session data as well as dealing with incoming and outgoing packets. It is coded in Java and organized into a set of micro services, with each service managing its own tasks and responsibilities in this process. The term *stack member* refers to a single micro service in this context. A Station Server is organized as a Virtual Machine (VM) and can be treated as a black box. The ground station only needs to start the VM and it will take care of loading up all micro services. The owner of the ground station does not need to technical subtleties which might complicate a normal setup.

Separating and breaking up bigger tasks into smaller services causes the system to be tidily separated into logical units, which can be tested and maintained separately. Another meaningful advantage is that the systems can be loosely coupled, which means that a micro service is independent, or dependent on an abstraction, never an implementation. This enables individual testing, since it is possible to implement a testing implementation of a dependency, which just provides testing data.

Micro services are connected via RMI. The architecture features a class that provides functions to host and retrieve the primary object of the service, which provides access to the functionality. While the current scope only supports running all services inside the same VM, RMI is flexible enough to support more sophisticated setups. It is for example possible to run a costly micro service on a computation server without modifying the system architecture. This is due to the fact that RMI uses internet protocols to share Java objects between applications. Since the service with the dependency needs access to the code to retrieve and use the object, any service is exported as a jar, which is then included as a library in dependent micro services.

4.2.2.1 Fetcher

The process of fetching in the context of this work refers to retrieving relevant data from a source and making them available somewhere else. In the context of the Fetcher this means collecting all existing data from the Network Environment, converting them into a form that can be used in Java and making them available to other stack members. Existing data includes all found announcers, every session of every announcer and every package of every session. The fetcher can be described as the root of the Station Server, since it is the only stack member which has no dependencies to other stack members via RMI. The only dependency is the Network Environment which is not part of the Station Server. The fetcher functions as an interface to easily access the Network Environment from Java, which makes it a crucial part of the Station Server. The fetcher works by retrieving all announcers from the registry when booting up (*Fetch Announcer* in Figure 4.3). A partner is able to customize which announcers are relevant, which causes the fetcher to ignore all other ones. The fetcher continually re-fetches the selected announcers, if the registry is active and running. It is crucial to implement error handling for the case that the registry or any announcer stops being available while the fetcher is running. In this case the fetcher should try to reconnect until the respective server is running again, without stopping the fetching process for the other elements. This enables the full usage of the advantages of a distributed system, since some elements can fail or have downtime for a short duration, without it causing issues. The gathered announcers from the registry are fetched individually, to collect all available data regarding every relevant session (*Fetch Session and Packets* in figure 4.3). Fetched JSON strings are converted to a respective Java instance, which can be processed by the other stack members. The instances are shared with them via RMI in two possible methods. The data can be accessed either directly or event based. Directly means it is for example possible to collect all sessions of an announcer or all packets of a session. This approach of sharing the data sadly is flawed, since stack members would need to fetch or check the shared data again, to recognize and react to changes. This issue is the reason why the fetcher provides an event API, which is the primary way to access the data and handle changes. Other stack members can subscribe and react to for example when new sessions or packets are available or when data for a session changes.

4.2.2.2 Propagator

The Propagator is one of the first elements in the chain of processing the data. The fetched data for each session (*Sessions (TLE's)* in Figure 4.3) is used by this stack member to calculate all satellite positions over a period of time in relation to the specific stations geo coordinates. Its responsibilities include determining when a satellite is visible above a certain ground station. This is required to determine, at which point in time it is possible to establish a connection to the satellite. This time range is referred to as an overpass as established in [9, p. 5] and briefly covered in chapter 2.1.3. An Overpass for this system consist of entry and exit points, which relate to the point in time at which the signal of the satellite can first be detected (arrival) and when the signal is lost.

Since the computation of this can be rather costly sometimes, the propagator features an expandable architecture to best cover different use cases. For example the next step in the processing chain needs the latest overpasses that are either happening right now, or will happen in a few minutes. If the system should have for example a calendar, which visualizes all overpasses in a given time frame, it would be necessary to calculate all existing overpasses in this window. These two use cases alone necessitate a modular system with different kinds of propagation modes. Access to these modules is shared via RMI.

4.2.2.3 Scheduler

Since the antenna is only able to track one satellite at a time, it is necessary to decide which satellite should be prioritized for communication. The scheduler is responsible of making this decision. While the field of scheduling is target of a wide range of research, optimizing this process will not be a primary concern in this system. TIM GSN features a simple yet effective solution and is designed so the system can be extended, if there is ever the need for a more sophisticated algorithm. [21, p. 1-2]

The TIM GSN handles scheduling in a single resource range context. It is inspired by the proposed model established in [9]. Propagated overpasses (*Overpasses* in Figure 4.3) are sorted into slots. A slot is a time window, in which the same satellites are passing over the ground station [9, p. 5]. For example, two overpasses which slightly overlap in the middle would be sorted into three distinct slots. In the case of such an overlap the scheduler has to answer one of the most relevant questions: Which satellite should be prioritized? This issue has been the content of numerous research papers and has been optimized using a variety of approaches. The TIM GSN makes it possible to implement different scheduling approaches and easily use them in production, by abstracting the scheduling process. The base implementation provided by the software uses the index of the corresponding session in the context of the announcer. If two sessions have same index, the index of the announcer is used instead. This enables operators of announcers to provide ground stations with an order of importance. More important satellites can be sorted with a lower index, so they get scheduled more often. This offers interesting approaches that can be implemented on the announcer side. One noteworthy approach is sorting sessions based on how many packets were received for each session. Satellites with low cover rates automatically get more communication time if available. Offloading the priority determination onto announcers enables a wide variety of these kind of approaches. Once a set of slots is solved the solution is further evaluated, by checking which of the solved time windows is currently active. The results of the scheduling process are shared via RMI. The scheduler also provides an event interface that notifies listeners when a new solution is found, or a new solved time window becomes active due to time passing.

4.2.2.4 Router

A router is responsible for handling active communication with the satellite through preexisting soft- and hardware of the ground station. The data processing is handled through a dedicated GNU Radio Graph, which connects to the router using a socket. This graph varies between different ground stations. One difference is the input and output stream that connects to the station specific system (*Input and Outputstream to Antenna* in Figure 4.3). Possible options include using an audio signal for the in and output stream, utilizing a socket the data is pushed through or directly transferring the data to a SDR as demonstrated in chapter 5. Depending on the existing signal processing chain of the preexisting system the signal processing happening before transferring the data needs to be adjusted as well. This can for example include modulating the data as well as adjusting the frequency of the signal based on the current doppler shift of the satellite. For this to work, it is required that the router provide the real time doppler shifted frequency to GNU Radio graph by fetching the current shift from the the propagator (*Doppler Shift* in Figure 4.3). This heavily depends on the preexisting system of the station, some stations might choose to calculate the shift them self and handle this at another point in the data processing chain. Modulation is a very relevant topic for the router and more precisely the GNU Radio graph handling the signal. Different satellites modulate their signal using different approaches. The system must be flexible enough to support a large array of modulation methods. This is done by instantiating a secondary GNU Radio graph which takes care of modulation. This graph, which is essentially a python script, is stored as a string as part of a session. When a new connection becomes active this graph is instantiated and linked up to the main graph using sockets. This enables users to freely customize modulation. The ZfT provides standardized graphs which deal with the most basic modulation approaches. Other participants can either use them or use them as orientation to build their own graphs. Besides managing the GNU Radio connection, the router is responsible for tracking the currently active solution of the scheduler (*Active Target* in Figure 4.3). This mostly includes the detection of data. If the GNU Radio graph detects successfully demodulated packets they need to be reported to the data intake (*Report received Packets* in Figure 4.3). One of the most relevant events which might happen while tracking is the retrieval of new packets from the active session (*To be sent Packets* in Figure 4.3). If this happens the packet content needs to be fed into the connected socket, which causes the data to be processed by the GNU Radio graph, so it can be sent out. After the package was sent, the system also needs to report that the package was successfully sent out, so that it can be removed from the transmission queue. This is done by sending the package to the data intake, which is responsible for handling this case (*Report Sent Packets* and *Remove Packets* in Figure 4.3). The router is also responsible for setting up the modulation graph for every active connection respectively as well es starting up the main GNU Radio graph for signal processing when loading up. These tasks can be summarized as preparing and setting up communication access to the currently active satellite connection.

4.2.2.5 Webserver

The webserver functions as an interface and universal access point for all calculated data by the ground station. Preexisting systems can easily formulate web request to required relevant data. The data is gathered from corresponding stack members via RMI. Besides the preexisting systems this interface is also used by the admin ui, since it is built on a web framework and unable to access the RMI instances.

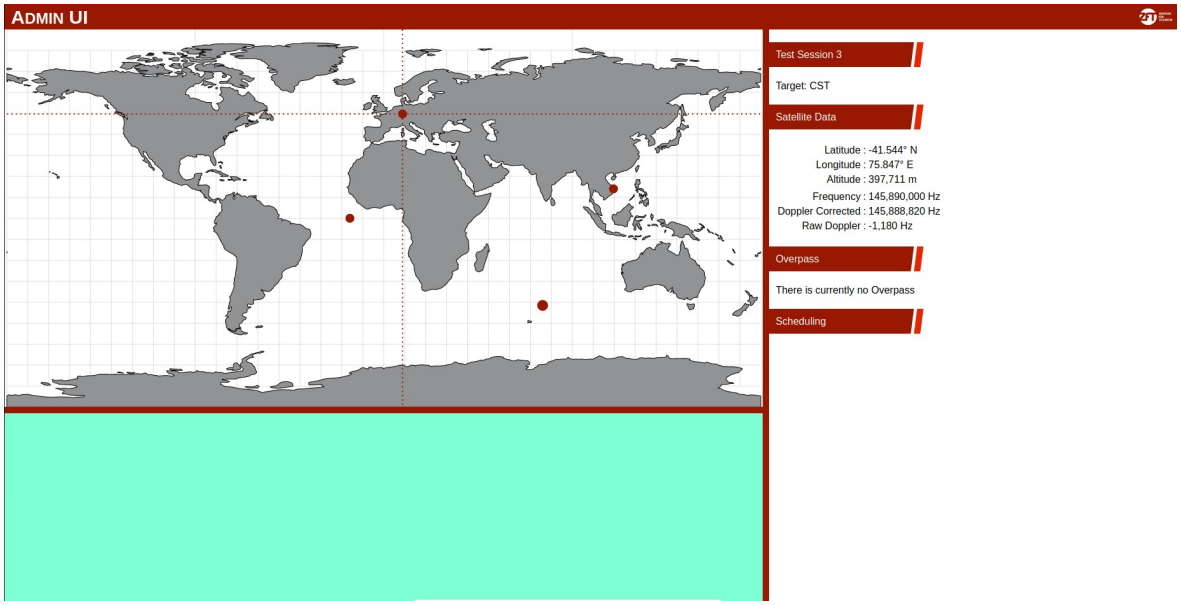


Figure 4.4: The current state of the Admin UI

Figure 4.3 displays all connections of the system. This includes retrieving satellite positions, overpasses and the doppler shift from the propagator as well as active scheduling results from the scheduler. The system might be expanded to allow specific web requests which utilize both the propagator and the scheduler, like for example requesting all overpasses scheduled in a certain time window. All the available data and functionality is available via web request and is used by both the admin ui, as well as the preexisting system.

4.2.2.6 Admin UI

The Admin User Interface (UI) provides visual access to all relevant data for the operator of the ground station. This includes visualizing currently tracked satellites as well as scheduling results. The Admin UI also fulfills a crucial legal requirement. When an antenna tracks a satellite, there is a need for a person to take responsibility for issues which might arise. The Admin UI ensures that the local operator has to confirm liability for incoming transitions, to avoid potential legal problems. The operator also is able to adjust and manage connection consent for different satellites individually. The system is designed, so the local operator has the last say about anything that happens. It was built using the Svelte Framework and relies on Node packages for visualization features.

Figure 4.4 shows the current state of the Admin UI. Satellites and satellite data is properly visualized, however there is no proper schedule display. This is due to the fact that TIM GSN is a large system and prioritized the finalization of core systems during its development. This includes primarily fetcher and router, since they are crucial parts of the direct processing chain of upstream packages. The AdminUI will be later extended for missing functionality.

5 Evaluation

The primary goal of this work is to develop a system that allows for remote bidirectional access to satellites. While the task of receiving and logging data is relatively simple, this evaluation focuses on the upstream aspect of communication. The full range of functionality for a production ready GSN requires the entire station server and each micro service, however the most crucial micro services are the in and out points of the station server. This comprises the fetcher, which handles the connection to the network environment and the router, which manages the signal streams for sending and receiving. While individual unit tests have been conducted to verify the core functionality of each microservice, it is crucial to also test their interactions and cooperation within the context of the entire system. To evaluate the performance of the GSN, a comprehensive test was conducted using a dedicated setup in collaboration with the ZfT, simulating realistic scenarios to ensure proper data flow.

5.1 Scope

The evaluation focuses on the following essential points: The data transmission should be tested from the announcer to the hardware level. This transmission should happen as fast as possible to prove the real time approach. The last goal is, to test the transmission in a production scenario, with the focus on being as realistic as possible. The scenario to conduct this test is a mission operator who wants to send data to a satellite by putting packages into the queue of an announcer. The evaluation should assess how good the TIM GSN is at fulfilling this task. Furthermore it will determine if the system abides to the constraints introduced in chapter 4.

5.2 Concept

The concept for the test at is shown in figure 5.1, while the final setup which was built at the ZfT is depicted in figure 5.2. The purpose of this test is to evaluate the upstream functionality of the network, which is done by simulating a sending ground station and a receiving satellite using two laptops (marked as 1 and 4 in figure 5.2).

The to be sent data is provided in the form of packages by an announcer. This announcer is hosted by the ZfT and can be accessed through a static URL. It has only one session which is the single source of packets for this test setup. The first laptop runs an instance of the station server. The fetcher hosts a debug registry, a custom registry designed exactly for this test setup which just provides access to the hosted announcer. While it would be possible to access the announcer directly, the goal is to simulate a realistic production scenario, which is the reason for using a registry. The station server processes the fetched data and lastly output the packet in the queue.

The test will evaluate the data processing ability utilizing the most critical elements of the chain. For this test the station server only uses the fetcher and the router. Since there is only one announcer and one session the router is able to fetch packets of the session directly without having to consider

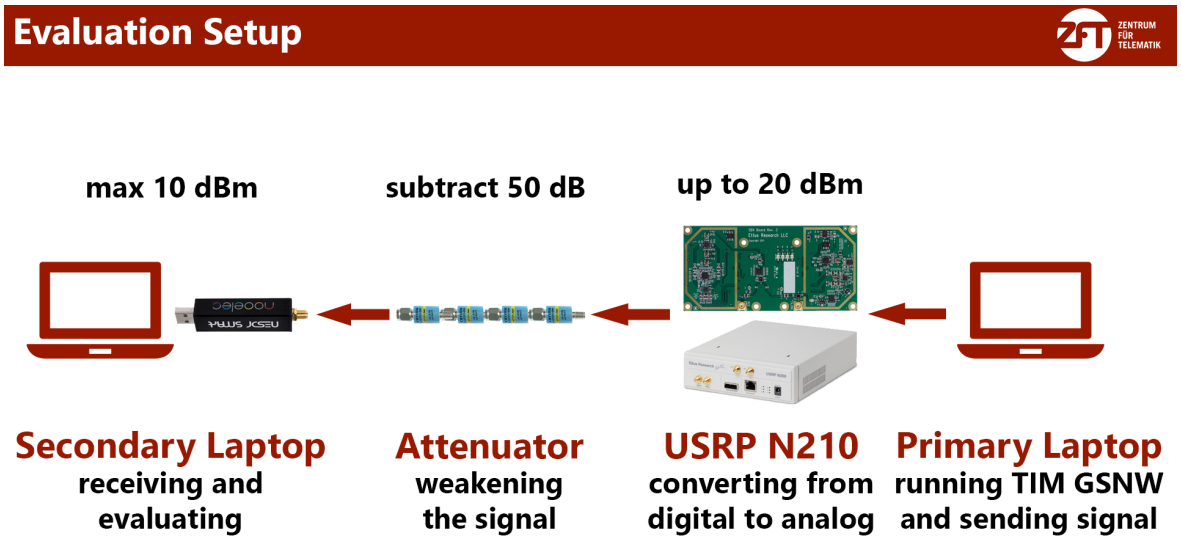


Figure 5.1: Evaluation Concept

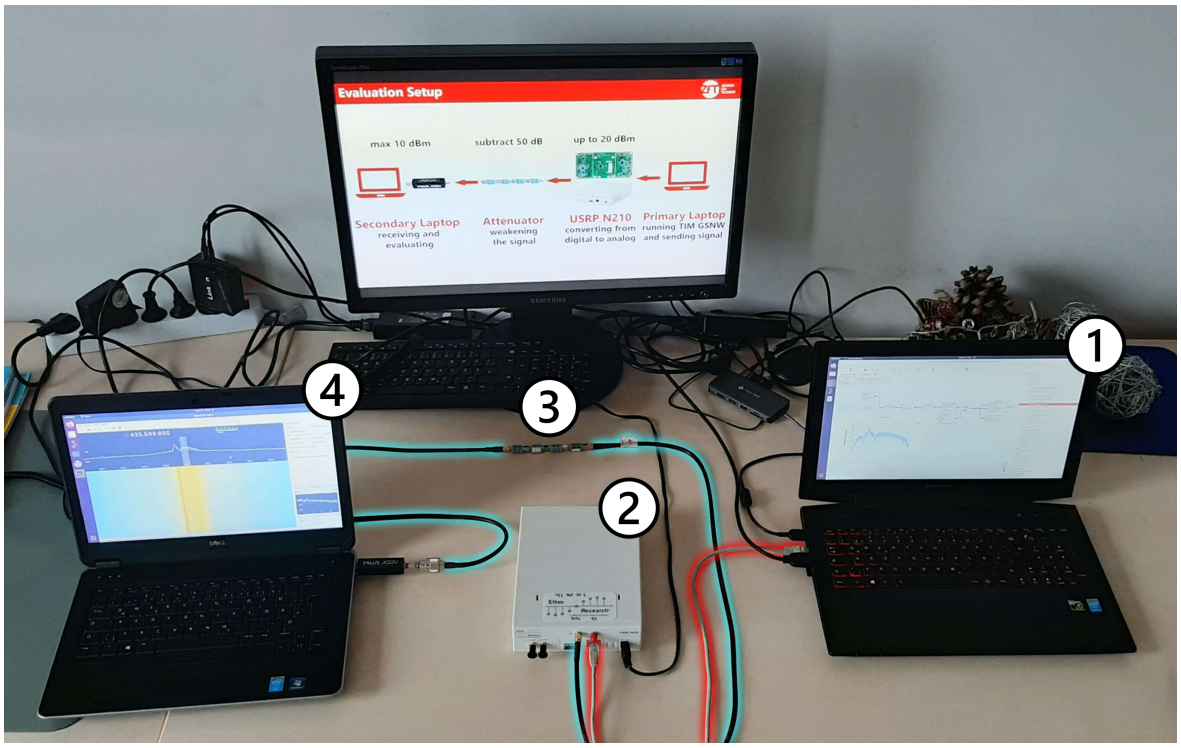


Figure 5.2: Evaluation Setup

the propagation or scheduling results. Despite its minimal configuration, this setup maintains a significant proportion of features while simultaneously simplifying the process of debugging and resolving issues.

The router sends the packages to a GNU Radio graph provided by the ZfT, which modulates the data into a digital signal that is forwarded to a connected USRP N200 (marked as 2 in figure 5.2). A network cable connects the laptop to the gigabit Ethernet interface of the device (marked red in figure 5.2). It can be easily accessed using its IP address while data is transferred using the UDP protocol. Hardware drivers which take care of connecting and sending data trough the network are provided by the company creating the device. [48]

USRPs need to be configured with a daughter board which is responsible of converting the digital signal into an analogue one, which is output to an rf cable. Different daughter boards are able to cover different frequency ranges depending on the use case. For this evaluation a SBX is used, which covers frequencies from 400 up to 4400 MHz. It outputs up to 20 dBm which is critical for connecting it to the second laptop. [47]

The signal is transported by an rf cable (marked blue in figure 5.2). The second laptop (marked as 4 in figure 5.2) uses a simple SDR receiver to convert the signal into data which can be processed. The receiver used is a Nooelec NESDR SMArt v5 SDR which has a maximum input capacity of 10 dBm. To transform the signal strength into a range the receiver is able to handle, the cable features multiple attenuators lowering the signal by 50 dB to -30 dBm (marked as 3 in figure 5.2). This level is not only realistic when receiving real signals, it also ensures that the SDR does not get a signal with a higher strength than it is able to handle, since this would cause the signal to overdrive. The second laptop runs simple software to demodulate, process and log the incoming data. [42]

5.3 Test Evaluation

The test concluded the following results. Data transmission is possible and working as intended. The packets were properly picked up by the fetcher when placed in the announcer queue and forwarded to the router. Furthermore the data corresponding to the packets was registered on the second laptop. This concludes that the router, GNU Radio, USRP chain works mostly as intended, however during the course of this test two notable issues were identified.

The first problem was RMI, which was already a point of concern regarding the event based architecture. In previous tests RMI worked without any notable issues however using it to share a tree like data structure the size of the registry created unforeseen problems. To continue with the tests, the system was successfully restructured to run all modules inside one Java application, to avoid the need for RMI. This caused minor thread safety issues which were easily eliminated. Future plans to return to RMI will be elaborated in the next chapter (see 6).

An additional problem, identified during the evaluation campaign concerned the VM the software ran in. Establishing a connection to the USRP was slightly more challenging in regards to transmission speed as the connection suffered from rate limitation. This happened due to the VM being too slow to handle higher data rates such as 2 million Hz.

This test proved the proper functionality of the fetcher as well as the router. Both systems work as intended. The real time approach was also able to be proven. With times under one fourth of a second between adding the package to the queue and the package being sent, it is save to say that the system runs fast enough to process data in real time. These values were even better than expected.

The evaluation also proved that it is possible to integrate the system into a working ground station. The system is able to communicate via SDR peripherals and the planned connection links work as intended.

5.4 Approach Evaluation

This section evaluates if the developed system fulfills the defined requirements. They can be found in section 4.1.

5.4.1 Functional Requirements

1. The software provides the ability to receive satellite data and distribute it using the internet. Incoming data is automatically handled by the router and forwarded to the respective data intakes configured for the satellite. The forwarding is done using web requests, which means data can be received worldwide. This behavior was established and evaluated very early in development and verified to be working during multiple subsequential tests.
2. The software provides the ability to send data to a satellite over the internet using the data processing chain of announcer, fetcher and router. This chain provides the ability to easily send data to a satellite over the network. Data can be added to a transmission queue for a certain satellite and the network will automatically fetch and upstream the data if the satellite is in range of a ground station. The system is supports bidirectional and was verified to be working in the described test.
3. The software is able to read modulated data from an SDR or a sound modem (downstream), which is realized using the respective GNU Radio graph. Demodulating data was proven to work during tests earlier in development. The test consisted of streaming in modulated audio and outputting the data via a socket to the router.
4. The software is able to output modulated data from an SDR or a sound modem (upstream), which is realized using the respective GNU Radio graph. Modulation of to be sent data was proven to work during the described evaluation of this work. The data was modulated and properly sent out to a USRP.
5. The software handles the propagation and scheduling of satellites and their overpasses over the ground station using the respective micro services dedicated to solely solving one task. Their behavior was tested and verified individually using unit tests of either system.

5.4.2 Non Functional Requirements

1. All technical knowledge and control will remain in the hands of the operator. Even though the Admin UI in its current stage is not applicable for providing the desired amount of control, the system is designed around this core principle and will be expanded to account for this current shortcoming. The Admin UI will provide the operator with all necessary tools to observe and control the station.
2. The software runs in real time as established in the evaluation.
3. The software should be easily portable. There is no logic which is hardware depended. When porting the software it should be flexible enough so most of it can be reused without convoluted problems. Even though the system was designed with flexibility as a primary concern, specific tests on porting capabilities were not yet carried out. The only indicator for the flexibility of the system is the fast adaptation when facing problems with RMI during the evaluation.
4. The software is easy to use for the ground station operator. The VM can be installed and executed on nearly every system with minimal setup. The user just has to deal with linking up the necessary in and out points of the system. The VM functions as a stand alone black box which just needs to be started. While running the software can be easily controlled using the admin UI
5. The software is overly transparent. The system is access and location transparent, the origin of the packet and the target ground station are entirely exchangeable. If a connection is possible packets are treated the same in all cases. Access and location transparency are the most critical transparency types. The system is furthermore concurrency transparent, all ground stations run in parallel while using the shared announcers. The system is also failure transparent, registry or announcer downtime is expected and silently handled in the background. The last relevant transparency this software fulfills is scaling transparency, new ground stations and announcers can be easily added without modifying the system. The system is lastly parallelism transparent, ground stations run in parallel and handle incoming data independently, which does not concern the end user. It is important to note that there are other types of transparency this system does not abide by, which are replication transparency, mobility transparency and performance transparency. These transparency types are not precisely relevant to this system.

6 Summary

The goal of this work was to find a system which is able to connect all TIM ground stations into one GSN. It was shown that there is not a single existing solutions which is able to abide to the established kill criteria. A new GSN concept was derived, established, implemented and evaluated. TIM GSN was proven to offer high flexibility in supporting possible ground station setup, support down and upstream while not offering a legal risk to participants. In its current stage it is ready to be deployed for beta testing.

6.1 Future Software Development

Since the current iteration of the system can only be considered a proof of concept, there are concrete plans regarding future development. The utter most effort will be directed towards ensuring the proper functionality of all systems as the system is step by step deployed. This includes further improving and extending unit tests to uncover possible existing errors as well as to avoid errors which might be introduced when expanding the code base.

Despite the evaluation uncovering deficiencies in the usage of RMI, it is intended to revert to its utilization due to the flexibility it offers. As described in chapter 4.2.2 it offers the ability to run micro services on entirely different machines, which makes the system extremely adaptive. Current approaches include abstracting Orekit away from shared classes to reduce the size and complexity of the shared tree.

Another notable step is the currently missing control of the Admin UI. It is lacking a schedule visualization as well as features to control incoming overpasses for the operator. These features need to be added and tested.

6.2 Outlook

The end of this work will be the start to a dedicated beta test program in cooperation with partners, especially in the USA and South Africa, which already agreed to utilizing the software. Due to regular consultation with South Africa an earlier iteration was already tested and verified to be working. Once the beta test program is started these kind of tests will be expanded to utilize the new ground station stack with all active subsystems. The TIM GSN will be utilized by partners and will stand out as a platform of cooperation and mutual be

Bibliography

[1] Federal Aviation Administration. “The Annual Compendium of Commercial Space Transportation: 2018”. In: (2018). URL: https://www.faa.gov/about/office_org/headquarters_offices/ast/media/2018_ast_compendium.pdf.

[2] Altexsoft. *Functional and Nonfunctional Requirements: Specification and Types*. July 2021. URL: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/.

[3] Amazon. *AWS Ground Station*. 2022. URL: https://aws.amazon.com/de/ground-station/.

[4] Nariman A Salam Bauomy and Eslam Ahmed Elbeh. “Design of SDR Simulation for Wireless Communication between Ground Station and CubeSat Implemented by LabVIEW”. In: (2020), pp. 60–63.

[5] Eugene Blanchard. *X.25 - TelecomWorld 101*. July 2013. URL: https://www.telecomworld101.com/X25.html.

[6] Stephan Bögel. *RMI - Remote Method Invocation*. 2005. URL: http://www.sbgl.de/rmi/.

[7] BryceTech. “Smallsats by the Numbers 2022”. In: (2022). URL: https://brycetech.com/reports/report-documents/Bryce_Smallsats_2022.pdf.

[8] Kevin Croissant et al. “An Updated Overview of the Satellite Networked Open Ground Stations (SatNOGS) Project”. In: (2022). URL: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=5331&context=smallsat.

[9] Rogério De Carvalho. “Optimizing the Communication Capacity of a Ground Station Network”. In: *Journal of Aerospace Technology and Management* 11 (May 2019). DOI: 10.5028/jatm.v11.1026. URL: https://www.researchgate.net/publication/333006277_Optimizing_the_Communication_Capacity_of_a_Ground_Station_Network.

[10] Stephanie DelPozzo and Caleb Williams. “Nano/Microsatellitemarket Forecast, 10th Edition”. In: (2020). URL: https://www.spaceworks.aero/wp-content/uploads/Nano-Microsatellite-Market-Forecast-10th-Edition-2020.pdf.

[11] Slavi Dombrovski. “Simple Downlink Share Convention v0.9 (SiDS)”. In: (2015).

[12] ESA. *Network map*. Dec. 2015. URL: https://www.esa.int/Enabling_Support/Operations/ESA_Ground_Stations/Estrack_ground_stations.

[13] Rhys Haden. *X.25 and X.121 address, packet switched network, Link Access Protocol Balanced*. 2022. URL: https://www.rhyshaden.com/x25.htm.

[14] Wolfgang Heinen and Martin Unal. “Scheduling tool for ESTRACK ground station management”. In: (2010), p. 2263. URL: https://arc.aiaa.org/doi/pdf/10.2514/6.2010-2263.

[15] Ryan Hevner et al. “An Advanced Standard for CubeSats”. In: *25th Annual AIAA/USU Conference on Small Satellites* (2011). URL: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1111&context=smallsat.

[16] Siegfried W. Janson. “25 Years of Small Satellites”. In: *25th Annual AIAA/USU* (2011). URL: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1111&context=smallsat.

[17] Mark D Johnston et al. “Automated scheduling for NASA’s deep space network”. In: *AI Magazine* 35.4 (2014), pp. 7–25. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2552.

[18] Limi Kalita. “Socket programming”. In: *International Journal of Computer Science and Information Technologies* 5.3 (2014), pp. 4802–4807. URL: https://doc.presentica.com/11477009/5ebad98adffc8.pdf.

[19] Rei Kawashima, Shinichi Nakasuka, and Tetsuo Yasaka. “A Perspective on International Collaborative Programs in UNISEC-Challenges of Japanese University Students”. In: (2012), E1–2. DOI: https://doi.org/10.2514/6.IAC-06-E1.2.07.

[20] Gary C Kessler. “An overview of TCP/IP protocols and the internet”. In: *InterNIC Document, Dec* 29 (2004), p. 42. URL: http://www.sci.brooklyn.cuny.edu/~parsons/courses/3120-fall-2012/notes/tcp-ip-notes.pdf.

[21] Alexander Kleinschrodt, Nikolai Reed, and Klaus Schilling. “A comparison of scheduling algorithms for low cost ground station networks”. In: (2016), pp. 1–15. URL: https://www.researchgate.net/profile/Alexander_Kleinschrodt/publication/308899251_A_COMPARISON_OF_SCHEDULING_ALGORITHMS_FOR_LOW_COST_GROUND_STATION_NETWORKS/links/580677c708ae0075d82c751e/A-COMPARISON-OF-SCHEDULING-ALGORITHMS-FOR-LOW-COST-GROUND-STATION-NETWORKS.pdf.

[22] Alexander Kleinschrodt et al. “Advances in Modulation and Communication Protocols for Small Satellite Ground Stations”. In: (Sept. 2017). URL: https://www.researchgate.net/publication/320165014_Advances_in_Modulation_and_Communication_Protocols_for_Small_Satellite_Ground_Stations.

[23] Alexander Kleinschrodt et al. “Ground Station Network for the TIM Nanosatellite Earth Observation Constellation”. In: *71st International Astronautical Congress* (2020).

[24] Alexander Kleinschrodt et al. “Mission Planning for the TIM Nanosatellite Remote Sensing Constellation”. In: (Oct. 2018). URL: https://www.researchgate.net/publication/328334288_Mission_Planning_for_the_TIM_Nanosatellite_Remote_Sensing_Constellation.

[25] Bryan Klofas. “GENSO: A Global Ground Station Network”. In: (2007). URL: https://www.klofas.com/papers/AMSAT_2007.pdf.

[26] KSAT. *KSAT - Kongsberg Satellite Services*. 2022. URL: https://www.ksat.no/.

[27] Erik Kulu. “Nanosatellite Launch Forecasts - Track Record and Latest Prediction”. In: *36th Annual Small Satellite Conference* (Aug. 2022). URL: https://digitalcommons.usu.edu/smallsat/2022/all2022/7/.

[28] Jigna Lad et al. “Complexity-Based Link Assignment for NASA’s Deep Space Network for Follow-the-Sun Operations”. In: (2018), p. 2579. URL: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2579.

[29] Catherine Lannes. “A new Generation of Monitoring and Control System for ESTRACK”. In: (2011), p. 1286386. URL: https://arc.aiaa.org/doi/pdf/10.2514/6.2012-1286386.

[30] Kyle Leveque, Jordi Puig-Suari, and Clark Turner. “Global educational network for satellite operations (GENSO)”. In: (2007). URL: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1506&context=smallsat.

[31] W Li and MK Hudson. “Earth’s Van Allen radiation belts: From discovery to the Van Allen Probes era”. In: *Journal of Geophysical Research: Space Physics* 124.11 (2019), pp. 8319–8351. DOI: 10.1029/2018JA025940.

[32] Andrei Margarint and Alexandru Barbovschi. “Automation of satellite tracking for worldwide ground stations”. In: (2015).

[33] Fergal Marshall et al. “Development of the Ground Segment Communication System for the EIRSAT-1 CubeSat”. In: (2021). URL: https://researchrepository.ucd.ie/bitstream/10197/12233/1/SpaceOps-2021,4,x1364.pdf.

[34] Douglas Messier. *Tiny ’Cubesats’ Gaining Bigger Role in Space*. May 2015. URL: https://www.space.com/29464-cubesats-space-science-missions.html.

[35] Douglas Mudgway. *Uplink-downlink: a history of the NASA Deep Space Network, 1957-1997*. Vol. -1. NASA, Jan. 2000. URL: https://history.nasa.gov/SP-4227/Uplink-Downlink.pdf.

[36] Yuya Nakamura and Shinichi Nakasuka. “Ground station network to improve operation efficiency of small satellites and its operation scheduling method”. In: (2006), pp. C1–6. URL: https://arc.aiaa.org/doi/abs/10.2514/6.IAC-06-C1.6.10.

[37] NASA. *Networks*. 2022. URL: https://www.nasa.gov/directorates/heo/scan/services/networks/index.html.

[38] NASA. “Small Spacecraft Technology State of the Art Report”. In: (Oct. 2021). URL: https://www.nasa.gov/sites/default/files/atoms/files/soa_2021_1.pdf.

[39] NASA. *Voyager - Mission Status*. Dec. 2022. URL: https://voyager.jpl.nasa.gov/mission/status/.

[40] NASA. *What is the Deep Space Network?* Mar. 2020. URL: https://www.nasa.gov/directorates/heo/scan/services/networks/deep_space_network/about.

[41] Julien Nicolas. “SatNOGS: Towards a Modern, Crowd Sourced and Open Network of Ground Stations”. In: 2.1 (2021). URL: https://pubs.gnuradio.org/index.php/grcon/article/view/100/60.

[42] Nooelec. *Nooelec NESDR SMArt v5 SDR*. 2022. URL: https://www.nooelec.com/store/sdr/sdr-receivers/nesdr-smart-sdr.html.

[43] Ryan Nugent et al. “CubeSat: The Pico-Satellite Standard for Research and Education”. In: (Sept. 2008). DOI: 10.2514/6.2008-7734. URL: https://www.researchgate.net/publication/242172999_CubeSat_The_Pico-Satellite_Standard_for_Research_and_Education.

[44] OneWeb. *One Web*. 2022. URL: https://oneweb.net/.

[45] Prof Puig-suari, Clark Turner, and William Ahlgren. “Development of the standard CubeSat deployer and a CubeSat class PicoSatellite”. In: 1 (Feb. 2001), 1/347–1/353 vol.1. DOI: 10.1109/AERO.2001.931726. URL: https://www.researchgate.net/publication/3901492_Development_of_the_standard_CubeSat_deployer_and_a_CubeSat_class_PicoSatellite.

[46] Tyler Reid. “Orbital Diversity for Global Navigation Satellite Systems”. PhD thesis. June 2017. URL: https://www.researchgate.net/publication/323245224_Orbital_Diversity_for_Global_Navigation_Satellite_Systems/figures?lo=1.

[47] Ettus Research. *SBX 400-4400 MHz Rx/Tx (40 MHz)*. 2022. URL: https://www.ettus.com/all-products/sbx/.

[48] Ettus Research. *USRP N200*. 2022. URL: https://www.ettus.com/all-products/un200-kit/.

[49] Mathew NO Sadiku and Cajetan M Akujuobi. “Software-defined radio: a brief overview”. In: *Ieee Potentials* 23.4 (2004), pp. 14–15.

[50] Klaus Schilling et al. “TOM: A Formation for Photogrammetric Earthobservation by three CubeSats”. In: *4th IAA Conference* (2017).

[51] Thomas Schmid. “Gnu radio 802.15. 4 en-and decoding”. In: *unpublished document and source* (2006).

[52] Marco Schmidt. “Ground station networks for efficient operation of distributed small satellite systems”. PhD thesis. Universität Würzburg, 2011. URL: https://opus.bibliothek.uni-wuerzburg.de/opus4-wuerzburg/frontdoor/deliver/index/docId/4984/file/DissertationSchmidt.pdf.

[53] Marco Schmidt and Klaus Schilling. “Internet-based ground stations networks for pico satellites”. In: (2008), p. 3205. URL: https://arc.aiaa.org/doi/pdf/10.2514/6.2008-3205.

[54] Damien Servant. “The new CNES station scheduling system: both an operational and simulation tool”. In: (2012), p. 1278273. URL: https://arc.aiaa.org/doi/pdf/10.2514/6.2012-1278273.

[55] Cal Poly SLO. “CubeSat Design Specification”. In: (2022). URL: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/62193b7fc9e72e0053f00910/1645820809779/CDS+REV14_1+2022-02-09.pdf.

[56] Jean-Marc Soula, Olivier de Beaumont, and Marc Palin. “The CNES ground networks: lessons learned and future plans”. In: (2008), p. 3414. DOI: 10.2514/6.2008-3414.

[57] Leaf Space. “Leaf Space – Ground segment as service”. In: (2022). URL: https://leaf.space/.

[58] SSC. “SSC CONNECT”. In: (2021). URL: https://sscspace.com/wp-content/uploads/2022/08/SSCConnectFolder_web.pdf.

[59] Starlink. *Order Starlink*. 2022. URL: https://www.starlink.com/.

[60] Robert W. Stewart et al. “Software Defined Radiousing MATLAB®& Simulink®and the RTL-SDR”. In: *Strathclyde Academic Media* (2015).

[61] TIA. *VLEO for Telecommunications - Very Low Earth Orbits (VLEO) for Satellite Communications ARTES FPE*. Mar. 2021. URL: https://artes.esa.int/projects/vleo-telecommunications.

[62] UNISEC. *Ground Station Network*. 2022. URL: http://unisec.jp/activitiesen/groundstationen.

[63] Jian Wu et al. “Research on Task Priority Model and Algorithm for Satellite Scheduling Problem”. In: *IEEE Access* PP (July 2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2928992.

[64] Klemen Zakšek et al. “Using picosatellites for 4-D imaging of volcanic clouds: Proof of concept using ISS photography of the 2009 Sarychev Peak eruption”. In: *Remote Sensing of Environment* 210 (2018), pp. 519–530. URL: https://www.sciencedirect.com/science/article/abs/pii/S0034425718300737.

Eidesstattliche Erklärung

Hiermit versichere ich – Jesco Vogt – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Mittweida, 08. January 2023	
Ort, Datum	Jesco Vogt