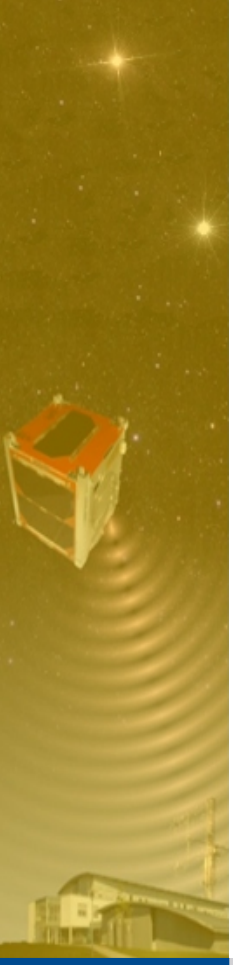


Institut für Informatik
Lehrstuhl für Robotik und Telematik
Prof. Dr. K. Schilling

Würzburger Forschungsberichte in Robotik und Telematik

Uni Wuerzburg Research Notes in Robotics and Telematics

Julius-Maximilians-
UNIVERSITÄT WÜRZBURG

Band 6

Marco Schmidt

Ground Station Networks for Efficient Operation of Distributed Small Satellite Systems

Lehrstuhl Informatik VII
Robotik und Telematik
Am Hubland
D-97074 Wuerzburg

Tel.: +49 (0) 931 - 31 - 86678
Fax: +49 (0) 931 - 31 - 86679

schi@informatik.uni-wuerzburg.de
http://www7.informatik.uni-wuerzburg.de

ISSN (Internet): 1868-7474
ISSN (Print): 1868-7466
eISBN: 978-3-923959-77-8

Die Schriftenreihe

wird vom Lehrstuhl für Informatik VII: Robotik und Telematik der Universität Würzburg herausgegeben und präsentiert innovative Forschung aus den Bereichen der Robotik und der Telematik.

Die Kombination fortgeschrittener Informationsverarbeitungsmethoden mit Verfahren der Regelungstechnik eröffnet hier interessante Forschungs- und Anwendungsperspektiven. Es werden dabei folgende interdisziplinäre Aufgabenschwerpunkte bearbeitet:

- Robotik und Mechatronik: Kombination von Informatik, Elektronik, Mechanik, Sensorik, Regelungs- und Steuerungstechnik, um Roboter adaptiv und flexibel ihrer Arbeitsumgebung anzupassen.

- Telematik: Integration von Telekommunikation, Infor-matik und Steuerungstechnik, um Dienstleistungen an entfernten Standorten zu erbringen.

Anwendungsschwerpunkte sind u.a. mobile Roboter, Tele-Robotik, Raumfahrtsysteme und Medizin-Robotik.

Zitation dieser Publikation

SCHMIDT, M. (2011). Ground station networks for efficient operation of distributed small satellite systems. Schriftenreihe Würzburger Forschungsberichte in Robotik und Telematik, Band 6. Würzburg: Universität Würzburg.

Ground Station Networks for Efficient Operation of Distributed Small Satellite Systems

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius-Maximilians-Universität Würzburg

vorgelegt von

Marco Schmidt
aus
Würzburg

Würzburg 2011

Eingereicht am: 13.5.2011
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr. Klaus Schilling
2. Gutachter: Prof. Dr. Shinichi Nakasuka
3. Gutachter: Prof. Dr. Hakan Kayal
Tag der mündlichen Prüfung: 29.7.2011

Contents

Chapter 1

Introduction

1.1 Motivation

In the last 10 years emerged a new approach in the field of satellite engineering. The idea of sending extremely small satellites into space originated from educational institutions and then adopted in diverse application fields. This paradigm shift involved building satellites from commercial of-the-shelf components, designed for restricted lifetime, but at affordable prices. Many of these pico and nano-satellites were already brought into orbit. The University of Würzburg contributed in that scope with the University of Wuerzburg Experimental satellites (UWE) and demonstrated successfully how extremely small satellites can be used to perform innovative space research.

A benefit of designing lightweight satellites is, that they can be launched 'piggyback' as secondary payloads, and hence at moderate costs. This makes it possible to install networks of satellites, carried from a single launch vehicle into space. Many researchers look at satellite networks (especially formations) as the next necessary step to transfer successful terrestrial distributed system technology to the context of space exploration, for example at utilizing virtual instruments or very long baseline interferometry. The concept of networked satellites promises progress in diverse application fields. Specifically from a computer scientists point of view, the network aspect is very interesting, although it is known that handling and controlling distributed systems is very challenging.

To better illustrate such issues, let us assume a satellite cluster of 10 small satellites is placed in similar low Earth orbits, acting together to achieve a common mission

goal. On the receiving side, 30 ground stations are available, geographically distributed over Europe. The starting point for this work was a simple question: How can data from vehicles in space be transferred efficiently to an operator on Earth? One could think of this as a simple deterministic problem, taking into account the orbit geometry and the transmission rate of each satellite. However, a closer look reveals many sophisticated theoretical and technical problems: For example, which specific ground station should track a satellite when several are visible at the same time? How can the restricted link capacity of a single satellite be compensated, when a superior number of ground stations are available? In which way can ground stations be efficiently synchronized with each other?
Starting from this point, a detailed investigation of current ground station networks was performed with focus on established infrastructure at academic institutions. These satellite receiving stations are built from low-cost devices and components, the architecture being similar all over the world as they are primarily used for operation of small satellites. The actual transmission rate of these satellites are limited to utilization of commercial of-the-shelf components and low frequency bands for communication. Nevertheless, the lack of performance can be compensated with intelligent utilization of network resources and redundant communication links. The central topic of this contribution are different strategies to improve the operation of small satellites with ground station networks. In the following chapters are different concepts that optimize the use of ground station resources in the scope of small satellite missions proposed. Their performance is investigated in detail and results from conducted experiments are presented.

1.2 Contributions

This contribution achieves progress for improved performance in ground station networks by combining results from three major fields:

Scheduling for low cost ground station networks

An efficient utilization of ground station resources requires proper scheduling of satellite contact windows. In this context, the particularities of low cost ground station networks were analyzed and specific scheduling requirements were derived. The limited applicability of state of the art satellite range scheduling algorithms to cur-

rent situation in small satellite scenarios leads to identification of a novel scheduling problem, incorporating the issue of redundant contact window scheduling. This new problem formulation reflects the high degree of flexibility and redundancy of such missions. In this scope, a scheduling approach was developed to guarantee equal distribution of redundant ground station resources between individual scheduling requests. This innovative approach was implemented and tested extensively, with the test results demonstrating the applicability and efficiency of the proposed system.

Data synchronization of satellite downlinks

The advantage of highly distributed ground station networks is the inherently high degree of redundancy. This enables the reception of a satellite from multiple terrestrial stations, resulting in parallel received data streams. In this work a novel concept is introduced, addressing the combination of redundant data streams to enhance the limited capabilities of state of the art small satellite communication systems. Crucial is the synchronization of parallel downlinks, which is a necessary prerequisite to enable the utilization of redundant information. For this, a time synchronization procedure for low cost ground station networks was developed, which is independent of external time sources and is accurate enough for the purpose of parallel data stream synchronization. Thus, taking into account the heterogeneous nature of current ground networks. Using that developed concept of synchronization has two major advantages: It significantly simplifies, on the one hand, the operation of small satellites due to automated data reception from parallel data streams and on the other hand, opens new possibilities for data recovery from redundant communication links.

Data management in ground station networks - Transmission error identification and recovery in distributed space missions

The limited power budget of current small satellite platforms causes more error prone communication links than in classic satellite engineering. To overcome this, the available capacity of highly distributed ground station networks needs to be coordinated in an efficient way: A novel concept was developed to use parallel downlink channels from a satellite to identify and correct transmission errors. The proposed

system identifies the occurrence of transmission errors from redundant information contained in parallel received data frames. This detection process is based on the data synchronization algorithm mentioned before. When transmission errors are detected, the system resolves the data inconsistency automatically. In this work three algorithms were developed to decrease the bit error rate, each using a different source of redundancy. This innovative data management system improves the efficiency of small satellite communication and can be applied to any low cost ground station network. The proposed algorithm was implemented and tested extensively. Software simulations were used to identify the performance limits of the presented approach, the benefits for low cost ground station networks are discussed in detail. Furthermore, hardware in the loop experiments were conducted to validate the system for the application in real world scenarios.

1.3 Outline

The remainder of this monograph is structured as follows: Chapter 2 provides an overview and introduction to small satellites and distributed space systems. Small satellites differ from traditional satellite engineering in many ways, e.g. in utilized technology or application fields. This background chapter 2 intends to bring the reader closer to the concept of small satellites, its advantages and drawbacks, opportunities and limitations. In the scope of small satellite projects, many low cost ground stations were established, this work explains their particularities and special demands. Furthermore, the emerging field of multi satellite systems is discussed in detail, describing a paradigm change from single large satellites to distributed systems in space.
Chapter 3 is dedicated to scheduling in ground station networks. After the presentation of state of the art scheduling approaches and their problems, a new problem formulation from the special demands of small satellites is derived. A tailored scheduling approach solving the derived problem formulation is presented in section 3.2. The characteristics of the new approach are investigated in detail and its proven capability of redundant scheduling is described in section 3.3. The implemented scheduling system was used to evaluate the developed approach and to determine its performance for different mission scenarios and problem sizes.

Data synchronization as well as transmission error identification and recovery are handled within a single chapter, as both approaches are aligned with each other. The beginning of chapter 4 covers the general issue of data management in ground station networks and introduces similar concepts. A time and data synchronization algorithm for low cost ground station networks is presented in section 4.2. Due to the inherent high degree of redundancy and the available network topology of those networks, a satisfying synchronization procedure can be realized. The downlinked data is then used to identify and correct transmission errors, which is extensively explained and analyzed in section 4.3. The chapter ends with results from the conducted hardware experiments.
Chapter 5 concludes this work with a short summary and some remarks about future developments in the field of ground station networks.

Chapter 2

Background

In this chapter a general overview on small satellites and ground station networks is presented to sketch the necessary background for this work, which is related to efficient operation in ground station networks. As the ground segment is the complementary part to the space segment, it is useless to concentrate solely on stations on Earth, rather the complete system consisting of ground stations and satellites needs to be viewed in a holistic way. Therefore, this chapter introduces the reader to the different aspects of small satellite missions. The chapter starts with a general survey on the small satellite concept, its origin and development in the last years. It is followed by a section dedicated to ground station networks, its special role in small satellite projects as well as new architectures and concepts. The third section discusses opportunities of the emerging field of highly distributed space missions. The paradigm shift from large conventional satellites to multi satellite systems will be addressed in detail and examples of new application fields will be given. The chapter concludes with challenges and technologies for distributed space missions.

2.1 Small satellites in education and research

2.1.1 The small satellite concept

The phrase *small satellites* was promoted in the last years as an important catchword and expresses the ongoing miniaturization efforts in satellite engineering. Nevertheless, the term *small satellite* is not strictly defined, depending on the context it

is used for a wide range of space vehicles. In the last years the common understanding of a small satellite evolved to a spacecraft with less than 100 kilograms. Especially in academia the terms *pico*, *nano* and *micro-satellites* were established to describe different types of small satellites. Often is the following classification used [Haeusler and Wiedemann, 2008]: The mass of a pico-satellite is less than 1 kilogram, nano-satellites are space vehicles between 1 and 10 kg, micro satellites have a maximum mass of 100 kg. This classification is meanwhile used quite consistently, however, other definitions exist [Schilling and Brieß, 2008].
A large number of small satellites were launched successfully, first pico-satellites were operated in orbit in 2000, nano-satellites were launched long before (e.g. OSCAR-1 in 1961), but in the last 10 years the number increased steadily due to advances in miniaturization. In this work the term *small satellites* refers in general to *pico* and *nano*-satellites, i.e. small space vehicle with limited mass and dimension, which can be launched jointly to form swarms or formations.
One of the milestones in small satellite development is for sure the creation of the *CubeSat* standard (respectively *CubeSat* specification), which initiated a huge number of pico-satellite projects worldwide. The concept of a *CubeSat*, a pico-satellite with cubic shape, side length of 10 cm and a maximum mass of 1 kg (see figure 2.1), was first introduced by Professor Jordi Puig-Suari from California Polytechnic State University (CalPoly) [Puig-Suari et al., 2001a] [Nugent et al., 2008] and Robert Twiggs from Stanford University [Puig-Suari et al., 2001b] [Twiggs, 2002], the corresponding CubeSat Design Specification was already published in 1999. A first batch of 5 *CubeSats* was launched with a Russian Rokot rocket in 2003, in the last 7 years more than 30 *CubeSats* were brought into orbit, neglecting many more projects currently under development or waiting for launch (mainly at university level) [Schilling and Brieß, 2008].
One of the key issues for the success of the *CubeSat* specification is the standardized structure, which is now a broadly adopted interface used in small satellites development. Consequently, launch providers offer launch slots for *CubeSats*, several launch adapter devices for piggyback launches exist, launch opportunities are granted from the space agencies to universities.

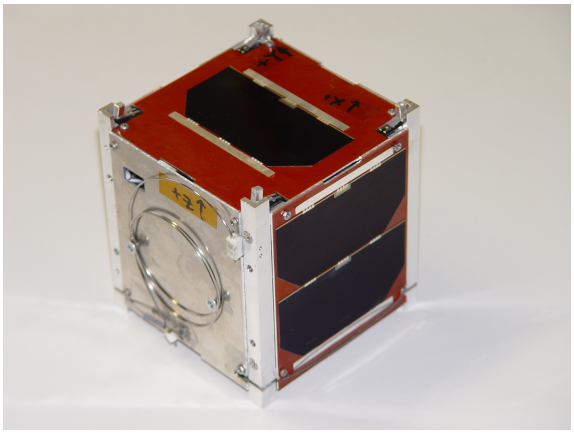


Figure 2.1: CubeSat UWE-1 from the University of Würzburg

Benefits of the small satellite concept

Building extremely small satellites has several benefits: The most prominent argument for reducing the mass of a space vehicle are the reduced launch costs. The price to bring an object into space directly correlates to the mass of that object, therefore less mass means less costs [Karabeyoglu et al., 2005]. That is also the main reason why the CubeSat standard is so successful in educational context, as the launch cost for a 1 kg satellite is affordable for a university. Meanwhile many funding sources are available to get an educational pico or nano-satellite into orbit [Schilling and Brieß, 2008].

Another benefit, which is closely related to the reduced mass (and consequently also reduced complexity), is the short development time. Several projects demonstrated already the development of a pico-satellite in less than one year. For some application fields, e.g. technology demonstrations, it is necessary to demonstrate or test a component in space without large delays. The small satellite concept enables realization in extremely short time frames, bottlenecks are currently rather launch opportunities, which quite often delay the operation of a satellite much longer than the actual development time.

Another important benefit is, that a costumer can conduct an experiment on a dedicated satellite and is therefore independent of other parties. This enables tailoring the satellite bus individually for one specific experiment or mission. Typical missions are related to the demonstration of a enabling technology or component testing [Barza et al., 2006].

Very interesting for future satellite missions is the usage of distributed satellite sys-

tems for new application fields. The innovative idea is the utilization of several small satellites instead of a single large satellite to perform a task or mission. Possible applications fields are for example multi-point measurements to increase the spatial and temporal resolution of a measured parameter (e.g. in space weather research) or efficiency improvement with interferometry (e.g. in astronomy). For a detailed discussion of distributed space missions and applications refer to section 2.3. Such distributed missions require the possibility to launch a large amount of vehicles into the desired orbit, here the small satellite concept shows its strength. Due to reduced size and mass it is possible to deploy many satellites with a single launch vehicle in orbit, like demonstrated already several times with the launch of CubeSat batches. Currently are missions planned with 50 nano-satellites integrated into a single launch vehicle [Muylaert, 2009].
Of course, the reduced size and weight of these satellites induces also restrictions: The most obvious drawback is the fact that the payload budget is very restricted. Therefore, huge devices or heavy components can not be accommodated on a pico-satellite, for example high resolution cameras or large amounts of propellant. Thus, current small satellite missions are restricted to low Earth orbits. Another issue is the restricted lifetime, due to restricted power resources or missing redundancy concepts. Admittedly, CubeSats have been operated for more than 5 years already, but the lifetime of most small satellites lies only between 6 month and 2 years. Many research groups try to overcome these issues be implementing more sophisticated redundancy concepts, but for some applications is a lifetime of 1 year fully satisfying. In summary, the small satellite concept is still a very new field of interest and many researchers focus currently on miniaturization and enabling technology. Nevertheless, the scientific potential leads to a steadily increasing number of planned small satellite missions, demonstrating the interest in this emerging field. This new concept of small, lightweight, orbital vehicles complements the traditional approach of satellite engineering and offers new strategies in niche applications, e.g. by utilization of distributed satellite missions.

2.1.2 Evolution of the pico and nano-satellite approach

The introduction of the CubeSat specification in 1999 initiated worldwide a wave of pico-satellite projects at universities. A short survey on national and worldwide activities is given in the following.

Figure 2.2: Pico and nano-satellite projects in Germany

Small satellite projects in Germany

In the last years several pico and nano-satellites were launched from German universities. The first German pico-satellite, UWE-1, was already launched in 2005 (see figure 2.1, for more details about UWE-1 refer to section 2.1.3). Several CubeSat missions followed in 2008 and 2009 (COMPASS-1 [Scholz et al., 2009], UWE-2 [Schmidt et al., 2009] and BEESAT [Kayal et al., 2008]). Further projects are currently under development (HEIDELSAT, SOMP) or are already acknowledged for launch (MOVE) (see figure 2.2). The mission objectives of these satellites comprise a wide range of experiments, ranging from technology demonstration (BEESAT) to telecommunication experiments (UWE-1). Furthermore, the educational aspects are promoted from all these projects [Schmidt and Schiling, 2009]. More information about the corresponding projects and their objectives as well as an extensive analysis of applications fields can be found in the PiNaNuPo study [Schilling and Brieß, 2008].

Meanwhile a strong community originated from the above mentioned projects. The yearly, between Würzburg and Berlin alternating, "pico and nano-satellite workshop" [1] brings researcher together and promotes knowledge exchange between them. The strong interest from outside Germany turned the workshop in the last years into

[1] http://www7.informatik.uni-wuerzburg.de/conferences/pina2011/

an international event with participants from all over Europe. Additionally strengthening the small satellite development in Germany is the established funding program [2] from the German space agency (DLR), which provides funding for launch costs and equipment for educational satellite projects. It is therefore expected that also in future the number of small satellite projects in Germany will increase.

Small satellite development worldwide

A rapid growth of initiated small satellite projects was observed in the last ten years. Beside advances in miniaturization also the publication of the CubeSat specification played a major role in this context. Currently, main contribution to the small satellite community comes from North-America, Asia and Europe. The most active community is located in the US, which launched so far more than half of all pico-satellites [Schilling and Brieß, 2008]. From Asia mainly Japan and China brought their own small satellites into orbit. Especially the robust CubeSats XI-VI and XI-V from the University of Tokyo are very remarkable, which are meanwhile operated for more than 5 years [Funase et al., 2005] [Funase et al., 2006]. In Europe a wide range of institutions (from Denmark, Netherlands, France, Italy, Switzerland etc.) are involved in small satellite projects, for more information about specific activities please refer to [Schilling and Brieß, 2008].
A survey of launched pico-satellites of the last decade is depicted in figure 2.3. It is easy to see that the number of launched pico-satellites raised significantly in the last 5 years. In this graph are only launched vehicles considered, the number of projects started or in progress is significantly higher. It is in general not possible to determine this number exactly, estimations range from 50 [Nguyen, 2007] to 180 projects [Schilling and Brieß, 2008]. Nevertheless, more than 50 launched pico-satellites in the last 5 years emphasize the growing interest in this innovative satellite concept.

2.1.3 The UWE satellite series

The computer science department of the University of Würzburg develops small satellites for education as well as for research purposes, they are the counterpart to the ground segment which is subject of this work. The University of Würzburg

[2] $http://www.dlr.de/rd/desktopdefault.aspx/tabid-2265/3376\_read-12596/$

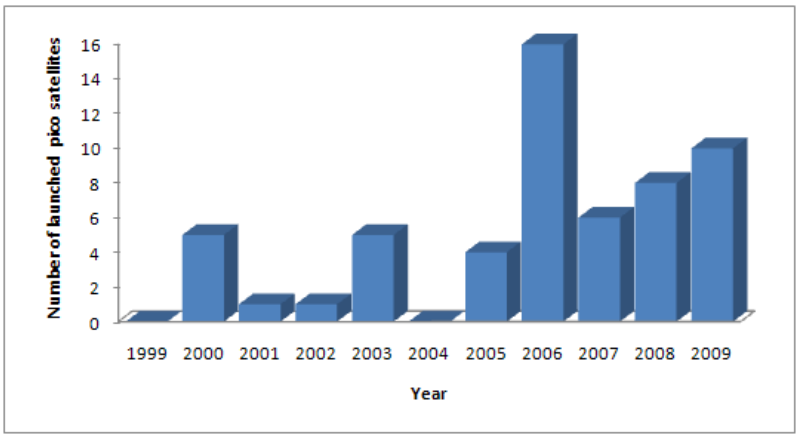


Figure 2.3: Launched pico-satellites from 1999 to 2009

Experimental satellite (UWE) project was started in 2003 and launched meanwhile 2 satellites successful into orbit, the third satellite is currently under development and will be launched prospectively begin of 2012. Pico and nano-satellites are still a very new field of interest, main focus of the research work in Würzburg are small satellite components, robust system design and control of satellite swarms and formations [Schilling, 2009c]. The UWE satellites are introduced here mainly for two reasons: First, the satellites are a good example of state of the art technology, which is beneficiary to understand the special requirements and needs for the corresponding ground segment. For example is the limited link budget of current small satellites one of the most important arguments for the usage of ground station networks. Second, it gives a good survey on space related research activities at the University of Würzburg, which are closely related to this work. Especially the future plan to operate swarms of satellites in orbit is one of the driving forces of this work.

The pico-satellite UWE-1

UWE-1 was built at the Institute of Robotics and Telematics in Würzburg in about 1.5 years project time in the scope of an educational work. The UWE project from Würzburg showed how a pico-satellite can be successfully used as an experimental platform to test new hardware and software components. The satellite bus contains the common subsystems, for example Onboard Data Handling (OBDH) and Power Supply Subsystem (PSS). To enable a flexible and robust architecture design, it was decided to use a microprocessor on UWE-1 instead of the commonly used solution of a microcontroller. UWE-1 uses for this purpose a 16 bit Hitachi H8S 2674 processor,

which has low power consumption. Under normal conditions the complete OBDH system requires less than 300 mW. The communication is handled from a modified commercial of-the-shelf (COTS) transceiver called PR430. The PR430 device combines a TNC and radio equipment within a small device. As operating system (OS) is the *uClinux* Linux distribution used. The complete UWE-1 architecture is described in detail in [Herbst et al., 2005] and [Schilling et al., 2006]. Additional to the satellite platform a complete ground station was implemented, which enables communication with UWE-1 and other satellites communicating in the VHF/UHF frequency bands. The ground station is mainly composed of COTS components and is connected to the Internet. More details about the specific architecture of such a ground station follows in section 2.2.1.

Mission objectives

The UWE-1 project had two major mission objectives: First objective was a technology demonstration of new hardware and software components. The integrated triple junction GaAs solar cells from industry were tested first time under space conditions [Barza et al., 2006]. The implemented OBDH software based on the *uClinux* distribution was also first time utilized on a pico-satellite in orbit. Second mission objective of UWE-1 was performing telecommunication experiments: The aim of these experiments was the characterization and optimization of the communication link. Especially the influence of disturbances on the radio link was investigated. The UWE-1 platform is an ideal platform for a broad spectrum of telecommunication experiments, as the system architecture is very flexible. The Linux OS provides a huge amount of available software components and standardized protocols. The optimization of the communication link was done in several steps: The first important optimization was the elimination of driver inefficiencies. The next step was than the optimization of the channel access algorithm regarding special radio link parameters. The application layer protocols of the communication link were optimized individually under consideration of cross layer dependencies with the AX.25 protocol. For more detailed information about the mission objectives of UWE-1 and the conducted experiments refer to [Schmidt and Zeiger, 2006].

Internet Protocols (IP) in space

One of the key aspects of the UWE-1 mission was the demonstration of IP in space on a pico-satellite. The terrestrial Internet grows more and more each day and therefore acts as a global interface for all kinds of technical devices, like mobile phones, laptops and TVs. Furthermore, many different components and protocols relying on IP are freely available and standardized. This is one of the reasons why in the last years a development towards IP in space was observed. Nevertheless, it is known that performance problems with TCP are not avoidable in space and that new solutions need to be found (compare section 2.4.2). The challenge for the UWE-1 mission was to combine IP in space with the restricted resources of a pico-satellite. As UWE-1 is running a Linux operating system, a complete IP protocol stack is provided from the operating system. This is a basic requirement to integrate the satellite in a global IP network. Additionally, several upper layer protocols are available to realize more sophisticated communication links, e.g. TCP, UDP, HTTP and TFTP. UWE-1 demonstrated successfully that it is possible to use small satellites for IP communication experiments, the conducted experiments contain file transfer over TFTP and HTTP data exchange [Schmidt et al., 2007]. This result advertises new opportunities to integrate small space vehicles in a worldwide IP based network.

The pico-satellite UWE-2

The successor UWE-2 was developed in 2007 and had the objective to demonstrate the capabilities of extremely small satellites in the field of attitude determination, which is a preparatory step for Earth observation applications or for coordination of satellite swarms. Because of the non-linearity of attitude dynamics and kinematics, an Extended Kalman Filter (EKF) is used to fuse gathered sensor data. The design of UWE-2 also follows the CubeSat specification, it includes all standard subsystems (e.g. power, OBDH, telecommunication), but was optimized with respect to size and weight. The architecture was inherited from UWE-1, the Hitachi H8 microprocessor runs as well the *uClinux* operating system [Schmidt and Schilling, 2008a]. The communication subsystem also uses the amateur frequency bands (VHF/UHF). Main emphasis of the UWE-2 mission was laid on the sensor suite for position and attitude determination, including a space qualified GPS receiver.

Attitude determination system of UWE-2

Due to the limited dimensions of the satellite, it is not possible to integrate highly accurate attitude determination sensors, like star trackers or horizon detectors, because of volume, mass and power consumption demands. Nevertheless, position and attitude determination is a basic capability needed for many applications. To enable an appropriate attitude determination on UWE-2, the pico-satellite was equipped with redundant sensors. The sensor equipment is mainly based on commercial micro-electromechanical system (MEMS) components, which were selected with respect to low mass and power consumption.
The UWE-2 satellite carries six pairs of perpendicularly mounted individual sun sensors, one on each panel. Therefore, the Sun will be continuously in the field of view, when the satellite is not in eclipse. For complementary measurements three miniature gyros are included, contributing changes in attitude and a direct measurement of turn rates. An accelerometer complements the gyros to form the inertial navigation system.
On-board the absolute position of UWE-2 is determined by a GPS receiver. It provides position data at a higher accuracy compared to a accelerometer/TLE solution. A specialized Phoenix GPS receiver [Montenbruck et al., 2006] was integrated in UWE-2 providing accurate time and position information. The mission objective was to demonstrate the performance of the ADS. More details about the ADS system of UWE-2 are given in [Schmidt et al., 2009].

Future prospects of the UWE project

The third satellite of the UWE series was completely redesigned. The new modular, standardized satellite bus supports rapid and robust development for future missions [Busch and Schilling, 2010]. The scientific objective of UWE-3 is the in orbit demonstration of attitude control on a pico-satellite platform [Schmidt and Schilling, 2010a]. The ADCS is one prerequisite for the long term goal to use satellite formations and swarms for advanced space missions. The upcoming satellites in the UWE series will therefore extend the capabilities of the satellite bus in each generation to finally achieve the realization of distributed space mission.

2.2 A new concept of ground station networks

The idea to combine ground stations in a network is of course not a new invention. Since the first space missions the ground segment consisted of several entities, exchanging information between these parts can be considered already as networking. The aggregation of different stations was already performed in the beginning of the space era to achieve better coverage or to increase redundancy. A good example for this "classic" approach is the ESA ESTRACK system [Maldari and Bobrinskiy, 2008], which was initiated already in the early 70's. Also NASA started quite early with the combination of different stations to networks, e.g. the Deep Space Network (DSN) was established in 1958 [Fisher et al., 1999]. Nevertheless, a new approach respectively concept of ground station networks originated from the small satellite community and developed in the last years. In this "new" concept the ground stations from the individual small satellite projects are combined to extend the access time to space vehicles. On the first glance the "classic" and the "new" ground station network approach seem fairly similar, nevertheless significant differences exist:

The most obvious difference between the classical and the new approach is the topology. The classic ground station network contains only a few, but therefore highly specialized stations, to support a broad spectrum of space missions. On the other side possess the newly emerged networks a high degree of distribution, composed of many low-cost ground stations (comparable to the architecture of the Internet), supporting very similar types of space missions (namely small satellites in LEO). This fundamental difference in topology affects the characteristics of the complete network, with all its requirements and objectives, but also constraints and bottlenecks.

Hence, the terms of "classic" and "academic" ground station networks are used throughout this work to differentiate between the above mentioned network topologies respectively ground network concepts. These differences are handled in more detail in section 2.2.1. Of course is the concept of academic ground station networks not restricted to the academic environment, it rather indicates the origin from the small satellite projects at universities.

Before addressing the new approach of *Academic Ground Station Networks (AGSN)*, an important aspect related to the taxonomy of the term *ground station* has to be

clarified: In literature is the space segment typically divided into mission control and ground station network [Wertz and Larson, 1999] [Wittmann and Hanowski, 2008]. The mission control is aggregated in control center (Mission Control Center (MCC), Spacecraft Operations Control Center (SOCC)) and is responsible for mission planning and mission operations, e.g. monitoring and commanding the spacecraft. The ground station network, composed of different ground stations, is on the other side dealing with signal reception and transmission, orbit tracking etc. The mission control and the ground station network are not only logically divided, also often geographically separated from each other. A typical ground station contains in this context several receiving stations including different types of antennas as well as corresponding hardware equipment. Exemplary, the ground station in Weilheim (part of the ESTRACK system) contains 6 different antennas ranging from 6 m to 30 m to support deep space missions as well as near Earth missions. So, a ground station describes in a classic sense a sophisticated system containing a larger number of facilities for satellite communication.
Contrary, in the context of small satellites projects, a ground station is considered rather as a single entity used to access one single satellite. This ground station is in contrast to the classical definition a standalone system, comprising all components to communicate with the satellite, i.e. antenna, transceiver, tracking hardware, data distribution system, etc. There is no clear distinction any more between mission control and ground stations. Furthermore, a ground station consists here typically only of one antenna system and corresponding equipment for communication with a single satellite in LEO.
This work refers in general to a ground station of the latter type, a complete system, capable to communicate with one satellite at a time (such a ground station is also available in Würzburg and was used in different parts of this work). The term *Academic Ground Station Network* refers to a network of individual ground stations interconnected through the Internet.

2.2.1 Classic and academic ground station networks

Architecture of academic ground station networks

The reason why academic ground stations are so similar in architecture is, that they are designed for the communication links typically used in small satellites. Due to

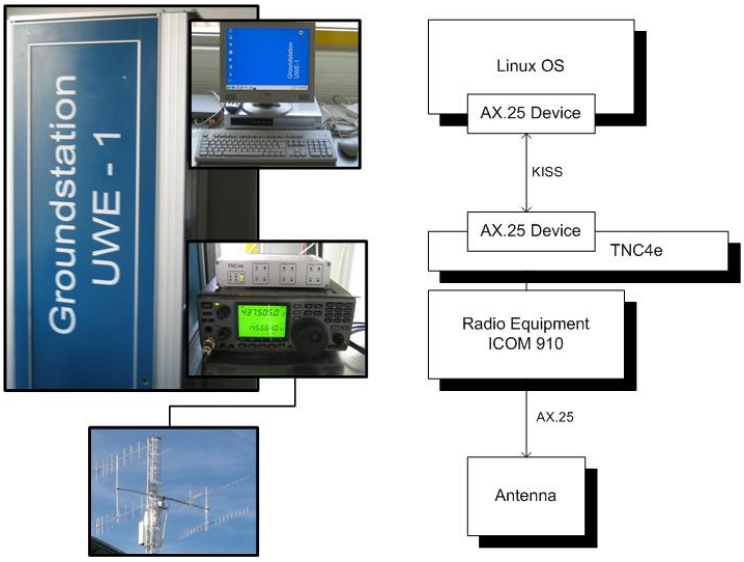


Figure 2.4: Architecture of ground station in Würzburg

the limited mass and power budget and the restricted pointing capabilities are primarily UHF and VHF transceivers used for communication. The utilized frequency bands, 70cm and 2m, are part of the amateur radio bands and are under the supervision of the International Amateur Radio Union (IARU[3]). Higher frequency bands, e.g. S-Band, will be the next step in the evolution process of small satellites, but are still rarely used [Kayal, 2000] (compare section 2.4.2).

This effects of course the architecture of academic ground stations: As hardware components for UHF and VHF are commercially available, many ground stations are built up from low cost commercial of-the-shelf components. Typically are standard radio transceiver and Terminal Node Controller (TNC) components used, which are connected to simple desktop computers (example architectures are described in [Hsiao et al., 2000], [Bester et al., 2003] and [Tuli et al., 2006]). That standard computer is normally connected to the Internet for data exchange. A variety of antennas and suitable tracking systems are offered. To control the antenna and radio equipment, several software solutions are available (open source as well as proprietary). A schematic diagram of the ground station in Würzburg is shown in figure 2.4. For a more detailed description of the ground station in Würzburg please refer to [Schmidt and Schilling, 2008b].

The ground stations of an academic network are connected through the Internet, i.e.

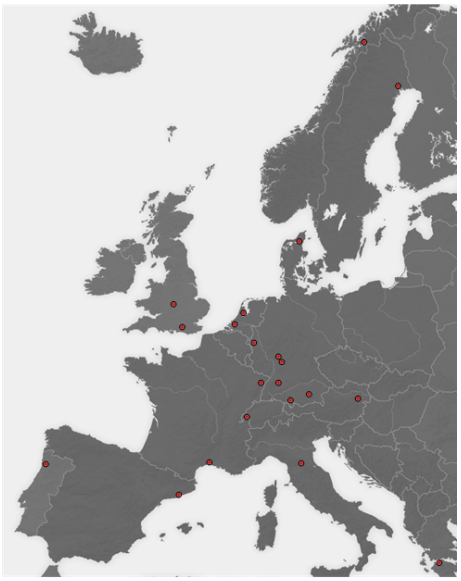
[3]http://www.iaru.org/

the Internet Protocol (IP) is used on the network layer, on the transport layer the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) is used, depending on the application on top (see also appendix B). Each ground station in the network can be seen as an access node to a satellite, which is in contact range. Typically only one communication link to a satellite can be established at the same time, but it is possible to track one satellite with several ground stations in parallel to achieve a more robust connection or to increase redundancy. Currently, IP is only for the data exchange between the ground station computers used, there is no real end-to-end communication between satellites and distant ground stations on basis of IP realized. IP in space is an approach pursued from several researches and will be further discussed in section 2.4.2.
The primarily used protocol for data exchange between ground station and satellite is AX.25 [Beech et al., 1997], which is conform to HDLC ISO standard 3309. AX.25 originates from the X.25 protocol and was adapted for the special needs of the radio amateur community. It is used as data link layer protocol in packet radio mode. Error detection is possible due to a 2 byte checksum attached to each frame, corrupted packets are discarded by default from a TNC device. In many ground stations is the TNC replaced by a software modem, emulated by a standard sound card [Rodriguez-Osorio et al., 2008]. This enables more control over the radio link and delivers new opportunities for post processing of corrupted data. Combinations of IP and AX.25 are possible and were demonstrated already with UWE-1 in space [Schmidt and Zeiger, 2006]. Studies showed that a HDLC conform protocol in combination with IP could perform equally to CCSDS standards [Schnurr et al., 2002] and is therefore at the current stage a suitable alternative. Furthermore is AX.25 accepted in the small satellite community as standard communication protocol, i.e. all major ground station network projects (see page 23) support AX.25. Nevertheless, a migration to CCSDS protocols might be on a long term view reasonable for compatibility reasons.

Differences between classic and new ground station approach

The distinction between "classic" and "academic" ground station networks is very important, due to crucial differences in architecture and topology and therefore different capabilities and characteristics. To express these differences in facts and numbers, the ESA ESTRACK system [Maldari and Bobrinskiy, 2008] as a typical

(a) European core network of the ESTRACK ground segment (Santa Maria Island, Azores not on map)



(b) Involved ground station of the GENSO system according to [Melville, 2009] [Page et al., 2010]

Figure 2.5: Comparison of ESTRACK and GENSO

representative of a classic ground station network is compared to the actual stage of the GENSO system, to illustrate the contrast between both ground station network types.

Comparing the size of academic and classic ground station networks is difficult, as a classic ground station network, like ESTRACK, evolved more than 40 years under the supervision of a space agency, while GENSO is still in the final development phase. Nevertheless, some numbers can highlight the difference in size: The ESTRACK core ground network consists currently of 9 ground stations located in Australia, Africa, Europe and South America. Since 1968, ESTRACK supported more than 60 missions. In contrast to that, more than 20 ground stations will be already involved in the testing phase of GENSO [Gil Biraud et al., 2009]. Mission concepts like QB50 foresee the operation of 50 nano-satellites, launched within a single mission [Muylaert, 2009]. So, it can be expected that in the next years a considerable amount of stations will cooperate in the GENSO platform. An illustration of geographical ground station distribution in Europe is shown if figure 2.5. In summary one can say that academic ground networks might support more missions in total, but those missions have typically only a short lifetime and do not require

a certain level of service, while classic ground station networks are used for less missions, but predefined services and support is guaranteed.

As pointed out in the previous section, ground stations at universities are low-cost standalone systems, mainly composed of COTS components. The individual nodes of the network can therefore be exchanged, as almost all small satellite missions are in LEO and the communication links use nearly the same frequency bands. Of course the ground stations are not replaceable when considering contact times, which are depending on the location of the ground stations, but in general it is possible to replace one ground station with another when the hardware enables access to the desired satellite. This effects for example scheduling requirements. On the other side, classic ground station networks contain highly specialized stations. Thus, dedicated stations for deep space missions and geostationary satellites exist and can in general not be replaced with each other. The ESTRACK system has for example dedicated 35m diameter antennas for deep space missions, for near-Earth missions 15m diameter antennas are provided. The frequency bands range from S-Band to K_a-Band [Maldari and Bobrinskiy, 2008], thus all kind of missions can be supported by ESTRACK, but dedicated stations for specific type of missions need to be requested.

Major emphasis in academic networks is placed on the need for flexibility. For classic ground station networks are operation plans created for long time spans to maintain a high utilization and to satisfy the needs of the customers. In the field of academic ground station networks the situation is totally different: Most participants are from non-commercial institutions, involved personnel is not available 24 hours a day, ground stations are sometimes unexpectedly not available. The result is a high dynamic topology in academic ground station networks, the need for flexibility results for example in different scheduling objectives compared to classical ground station networks.

Classic ground station networks are centrally organized, i.e. there is typically one controlling organization on top, for example ESOC in the case of the ESA ESTRACK system, which is able to manage or control the complete network. For example is maintenance or migration to new software or hardware managed centrally by ESOC [Maldari and Bobrinskiy, 2008]. For academic ground station networks this is not the case, the nodes of the network belong to individual organizations, therefore the coupling between the nodes is much weaker. So, there is no central entity able to

take decisions for the global network. That means, if a ground station has a failure, only the corresponding organization can undertake any actions. Also the usage of policies, standards and protocols is not always properly defined. Migration to new hardware or software is only possible if participating ground stations are willing to react. Therefore, controlling an academic ground station network is a more complex task, which imposes additional measures for new soft- and hardware solutions.
Another crucial point is, that classic ground station networks are open to paying customers. To run a huge ground station network a large part of the available budget comes from facility fees. Thus, the typical average cost of a tracking hour ranges between 300 and 450 Euros [Maldari and Bobrinskiy, 2008]. At the current stage in academic ground station networks there is no commercial interest, participants are contributing with their own ground station and can use other stations of the network free of charge. Costs occurring for a ground station, e.g. maintenance, are directed to the owning institution. This influences of course the objectives of the network: Rather than increasing the utilization of the network (which makes sense if it has to be commercially successful), the aim of academic networks is to share the sparse resource of ground stations hardware respectively access time. Therefore, also scheduling needs differ a lot (for more details related to scheduling see section 3.2.1). Finally, all peculiarities of both ground station network types are summarized in table 2.1 for a better overview.

Due to all this differences, it is not possible to simply transfer strategies for efficiency improvement from classic to academic ground networks. The topologic difference leads to new objectives and problems in data management, the altered requirements in flexibility and redundancy bring up new challenges in satellite scheduling. The main focus of this work is related to such efficiency improvements and is handled in detail in chapter 3 and 4.

2.2.2 Projects establishing academic ground station networks

Finally, some examples will present the current state of the art in academic ground station networks, technical issues will be handled in a short manner to emphasize the special characteristics of these networks.

Classic Ground Station Network (CGSN)	**Academic Ground Station Network (AGSN)**
Clustered in operation center	Highly distributed network
Supports wide range of satellite missions	Supports mainly small satellite LEO missions
High end devices and stations	Low-cost stations
Specialized, dedicated stations	Generic, replaceable stations
Strictly defined network topology	Dynamic topology
Centrally controlled and managed	Distributed organized
Provided to paying customers	No commercial interest
Utilization increase desired	Sharing resources to extend communication time desired

Table 2.1: Comparison of ground station network concepts

Global Education Network for Satellite Operations (GENSO)

The Global Education Network for Satellite Operations (GENSO) was started in 2006 from the International Space Education Board (ISEB), which consists of the educational departments of CSA, JAXA, NASA and ESA. Objectives of the GENSO project are to allow remote access for operators to their satellites over the network, provide remote control of the participating ground stations and to define and implement a global standard for educational ground stations.
The GENSO software provides a distributed platform based on a client and server architecture. A server application, the so called *Ground Station Server* (GSS) is installed on each ground station participating in the network, *Mission Control Clients* (MCC) are used from the satellite operators to bring the data from the satellite back to the owner [Page et al., 2008]. Secure access is granted from *Authentication Servers*.
From a technical point of view is the GSS software responsible for handling the interfaces to the hardware devices, i.e. rotor and radio equipment. Software libraries like the *Ham Radio Control Libraries*[4] were integrated to support a broad spectrum of COTS components [Shirville and Klofas, 2007]. The current software release supports already standard equipment like *ICOM* transceivers or *Yaesu* rotor

[4]http://sourceforge.net/apps/mediawiki/hamlib/index.php?title=Main_Page

controllers [Melville, 2009]. The MCC software is used from an operator to control the configuration of a connected GSS server. It contains a graphical user interface, but also provides interfaces for customized software solutions for satellite operation. The public response to GENSO is very remarkable and many small satellite developers announced their interest to join the GENSO network. More than 30 universities and radio amateurs are currently involved in the early operational phase [Page et al., 2010]. The project issued already a first version of its software, but is still under testing phase. For further information about GENSO and participation possibilities please refer to [Shirville and Klofas, 2007] and [Gil Biraud et al., 2009].

Ground Station Network (GSN) from the UNISEC group

The ground station Network project of the UNISEC group was already initiated in 1998. As JAXA also takes part in the ISEB, there is close cooperation between developers from the GSN and the GENSO projects. The GSN network was established on a national level in Japan. Objective of GSN is to construct a network-based ground station system with functionalities for remote control via Internet. More than 10 ground stations in Japan are participating in the project [Nakamura and Nakasuka, 2006]. Conducted experiments demonstrated already successful how the operation of pico-satellites can be improved by the usage of several ground stations [Oda et al., 2008].
The GSN system bases on the Ground Station Management Server (GMS), a package of software functions which are necessary to remotely control ground station hardware. As each ground station has a different hardware architecture, specific device drivers need to be implemented to interface the GMS client software, a detailed description can be found in [UNISEC-GSN, 2006]. The data exchange between client and local ground station hardware uses the Web Services technology, a W3C term for communication between devices using web standards, in this way is a platform independent remote control concept realized. An operator using GMS can directly control hardware devices like TNC or radio equipment.

Mercury

The Federated Ground station Network (FGN), initiated at Stanford University, envisions an autonomous, global distributed ground station installation to increase coverage and redundancy as well as enabling multi satellite support. The proto-

type implementation of FGN is called Mercury Ground Station Network (MGSN) [Cutler et al., 2002] and was published as open source project. It combines a three-tiered web architecture to enable remote control over the Internet.
A very interesting approach of Mercury is the concept of ground station virtualization [Cutler, 2003]. Here, virtualization describes the encapsulation of the provided functions of a single ground station, the result is reduced complexity of the overall system. Main driver for this design is the idea of transferring the end-to-end principle to ground networks. The end-to-end principle is a very common strategy for data exchange in distributed systems and handles protocol communication between end nodes in a network. The close relationship to the Internet is obvious: Ground stations could act similar as routers in the Internet and open new possibilities for space operations. The end-to-end principle enables a more flexible architecture for end user applications in space context. [Cutler and Bhasin, 2002].

A new approach of the ground segment, introduced as academic ground station networks, is still in an early stage. Ambitious projects like GENSO are progressing fast, but are still at the beginning. Currently the networks focus mainly on infrastructure and interfaces, the resources are shared with a kind of reservation system, which is ineffective with respect to utilization. This work starts from this points and proposes improvements for such networks on a global scale. Only with proper data management as well as scheduling, a more efficient usage of ground resources is possible. Especially for the operation of satellite swarms or formations, i.e. a highly distributed environment, such approaches are vital.

2.3 Distributed / multi satellite systems

The previous sections introduced the small satellite concept as well as the academic ground stations established in their scope. New opportunities due to reduced size and mass were pointed out and efforts in the ground segment to establish networks were discussed. Altogether are these developments strong indications towards a new generation of distributed space missions in the future.
What exactly is a distributed system? Andrew S. Tanenbaum defines a distributed system as network of independent computers that interact with each other to achieve a common goal [Tanenbaum, 2003]. A distributed system everyone is familiar with

is the Internet, but also a group of cooperating robots is a distributed system. Of course, this term can also be applied to space context, thinking of several satellites in orbit to perform a common task (e.g. the GPS satellite constellation). Meanwhile several applications were realized in space with distributed systems and topological terms exist to distinguish different kinds of distributed systems in space.
Most important terms describing the topology of distributed satellite systems are formations, constellations, clusters and swarms [Sandau, 2009], unfortunately these terms are not used consistently in literature. Especially in the scope of satellite range scheduling (see chapter 3), the topology of a distributed satellite system influences the difficulty of a problem instance.
Formation: Describes a multi-satellite system, which motion is controlled on-board with a closed loop to maintain relative distances between the vehicles in the same or very similar orbit [Scharf et al., 2003] [Leitner, 2004] [Schilling, 2009b]. In this way coordinated measurements can be performed or a single large instrument can be emulated by several small instruments (e.g. by interferometry). Formations are envisioned by many scientists as opportunity to realize new innovative mission concepts, as the size and mass of a single satellite is limited by launch vehicle restrictions. Up to now only small formations were demonstrated in space and mainly with satellites in the same orbit, flying behind each other (e.g. [Guinn, 2002][Serrano et al., 2009]).
Constellation: Is related to a multi-satellite system with its satellites distributed in different orbits. The vehicles are placed in space to achieve better coverage or higher temporal resolution. They are typically controlled individually from ground to maintain a coordinated ground coverage, the relative distance between the satellites is not controlled. Most prominent example of a satellite constellation is the GPS system, consisting of 32 satellites in 6 orbit planes. But also telecommunication and Earth observation constellations are already implemented (e.g. Iridium, Disaster Monitoring Constellation).
Cluster or **swarm:** Is used for a distributed system of spacecrafts cooperating to achieve a common goal without fixed absolute or relative position. The term swarm originates from flocks or school of fish, having in mind individuals acting together to achieve a common behavior. Satellites flying very close together and moving in almost the same orbit are often considered as swarms or clusters.

Throughout this work are the terms multi-satellite systems, distributed satellite

Figure 2.6: Distributed space system

systems and satellite networks used interchangeable, and describe in general a space segment with more than one space vehicle, which are used to achieve a common goal. When referring to a formation, constellation or cluster, the definitions above are used to distinguish between different topologies or control strategies.

Beside the space segment also the ground segment is integral part of each space mission. In a simple view, one could already see one satellite and one ground station as a distributed system (consisting of two nodes communicating with each other during contact). But considering a more general case, having a multi satellite system connected to a ground station network, is more suitable. Therefore, one can identify two different distributed systems, one in space and another one on ground, both in close relationship with each other (see figure 2.6). This viewing angle is sometimes neglected when multi-satellite systems are discussed, but of course the topology of the ground segment plays an important role for telemetry and telecommand. Due to the movement of the satellites on their orbits, the communication links between ground stations and satellites change frequently. These adds new challenges to the operation of a satellite network, careful planning and scheduling is necessary (see also section 2.4).

2.3.1 Distributed satellite systems as a new paradigm for satellite missions

In the last years a paradigm change in satellite engineering was observed worldwide. Since the first satellites have been launched, the dimensions and mass of space vehicles steadily increased. But in the last 10 years a new strategy of reducing the size of space vehicles has been pursued. Tiny satellite buses, like the SSTL micro satellites platforms [Jason et al., 2000], are commercially available and demonstrate the advances in miniaturization techniques. The reduction of size and mass was not only an effect of advances in miniaturization of electronic components (like in computers or mobiles), it also relates to the benefits of reduced launch costs (see section 2.1.1).

Beside the cost savings, a very interesting side effect of reducing the mass is the ability to launch several satellites together with a single launch vehicle, which is an important ability for future space infrastructure [Raymond et al., 2004]. This was performed already several times in the CubeSat community, several of these 1 kg satellites were piggybacked on the primary payload to further lower the launch costs. It is still a very common strategy to attach small satellites to larger ones as a dedicated launch vehicle would be to expensive. Meanwhile this launch strategy was extended to bring multi satellites into space.

The result is, that distributed satellite systems can be realized now by affordable costs. The current evolution to extremely small satellites will promote even more to highly distributed systems. The paradigm change from single large satellites to distributed space missions with a large number of satellites is currently going on [Schilling, 2009a]. An often envisioned scenario is a fully connected terrestrial and space Internet [Burleigh et al., 2002][Hogie et al., 2005]. The most critical question is now, why should we use distributed systems in space? A simple answer is, to realize distributed applications! Two exemplary applications fields are discussed in section 2.3.2. Main drivers for a distributed application in space is higher spatial or temporal resolution, or the usage of concepts like interferometry.

But not only new application scenarios are the driver for establishing networks in space, the concept of distributed intelligence was successfully demonstrated in the last years in various fields, for example Internet or sensor networks. The idea is always to have a huge number of nodes, each with restricted capabilities and knowl-

edge about the environment, but the complete network behaves "intelligent" and can fulfill a task more effective than one single complex system. Furthermore, the decentralized organization or control makes the system robust and reliable. Graceful degradation is a capability strongly desired in the case of failures.
Altogether this paradigm shift from large single satellites to distributed small satellites is apparent in many fields and a hot research topic discussed in many workshops and conferences .The concept of distributed satellite systems will not replace the classical approach of launching single satellites, it will be rather a possibility for specific mission scenarios.

2.3.2 Application scenarios for distributed satellite systems

The most critical question is of course what benefits have distributed systems in space? As it was already indicated, main driver for multi satellite systems are distributed applications. There are many good reasons why an application should be implemented in a distributed way, for example to achieve higher robustness if a failure occurs or to increase resolution. Additionally, approaches like interferometry can be used to achieve better performance than achievable with only a single satellite. Two exemplary applications will be discussed in the next sections to emphasize the benefits of distributed space applications. Of course many more applications and algorithms take advantage of distributed systems, but will not be elaborated in this work. For a more comprehensive overview of applications for distributed satellite systems please refer to [Schilling and Brieß, 2008], which contains a more extensive description of possible applications fields.

Distributed satellite systems for Earth observation

One application field with high potential for improvement through swarms of small satellites is Earth observation. Satellites in Low Earth Orbit (LEO) are able to monitor environmental changes without large delays. The low height of the orbit enables high spatial resolution on Earth and provides therefore huge potential for applications like disaster monitoring. Even with small satellites can environmental changes like natural disasters be observed. But due to the orbit, these satellites are orbiting with a high velocity around Earth, resulting in short observation periods in desired areas and large gaps until the ground track repeats. The solution for that

problem is a higher temporal resolution, achievable through several satellites in the same orbit. This can be easily achieved with the launch of several pico or nano-satellites. The reduced launch costs enable a high temporal resolution, resulting in a contact time roughly every 10 minutes. The future development of small satellites for Earth observation is therefore very promising [Sandau, 2009].
Another interesting application could be the creation of 3-dimensional images out of data collected by a swarm of satellites [Schilling, 2009a]. Through different viewing angles of satellites in the same orbit, flying in a leader-follower constellation, this could be achieved with simple means. A formation which demonstrated already such a constellation was implemented in 2000 with the Landsat-7 and EO-1 mission [Guinn, 2002]. Originally it was not intended to use these satellites for a formation, but by accident it was possible to demonstrate a formation flying due to the extended lifetime of Landsat-7. The next step is now to demonstrate a small satellite swarm for Earth observation purposes, challenges are especially the demonstration of accurate formation keeping in space.

Multi point measurements for space weather research

A very interesting application field for small satellite swarms is space weather research. Current missions related to that topic face two problems: To better approximate atmospheric models, a more detailed knowledge about relevant parameters is necessary. A single satellite delivers only single measurements at a given time, to increase the accuracy of current atmospheric models, the spatial resolution of the measurement has to be increased. The simplest way to realize that kind of multi-point measurements is a multi satellite system. A second crucial problem is the severe environment of the lower atmospheric layers, which reduces the lifetime of a satellite in low altitude Earth orbits dramatically. Hence, the usage of low-cost satellites to obtain information about the composition of these lower atmospheric layers is favorable.

2.3.3 Example mission

To illustrate the potential of distributed satellite systems an example missions is introduced. Of course many more missions are planned, but can not be described in detail within this work. More distributed mission concepts can be found in [Schilling and Brieß, 2008] [Sandau, 2009] and [Curtis et al., 2003].

QB50

A very promising mission relying on a multi satellite system is the QB50 project [Muylaert, 2009], initiated from the Karman institute and supported from many European experts. The scientific objective of QB50 is the in situ measurement of spatial and temporal variations of key parameters in the lower thermosphere. The measurements are performed from 50 double CubeSats, launched together and separated consecutively from the launch vehicle. All satellites are equipped with identical sensors, which obtain data approximately 100 km separated from each other. Furthermore, the re-entry process of the distributed satellite system is studied. Interesting is the fact that the 50 satellites are provided from 50 different institutes, so the satellite swarm itself is very heterogeneous. For operation is the academic ground station network GENSO foreseen (for more details see page 25). The project is a promising mission with scientific value, nevertheless suitable strategies for its operation are not available yet. One key aspect for the proper operation of 50 satellites flying near to each other in nearly the same orbit is efficient scheduling. Additionally, communication between such a highly distributed system and proper data transmission of measurements to Earth has to be addressed.

2.4 Challenges and technologies for distributed space missions

The potential of distributed applications was already pointed out in section 2.3.2. Nevertheless, from a technological point of view are still several challenges to be faced to realize such distributed missions in orbit. To conclude this chapter, major challenges as well as existing related technologies are discussed.

2.4.1 Challenges for highly distributed satellite missions

Formation flying

Many experts consider formation flying as the next necessary step to progress in application fields like VLBI or x-ray astronomy, which benefit from long baselines between sensor devices and are needed to establish so called virtual instruments [Raymond et al., 2004] [Xiang and Jorgensen, 2005]. Coordinated space vehicles can realize in that way performance unachievable using a monolithic approach [Sandau, 2009]. But on the other side, formation flying is due to stringent system requirements a very challenging research field [Scharf et al., 2004] [Schilling, 2009b]. Formations of conventional satellites would be to expensive for implementation in space. The small satellite concept is more cost-efficient due to reduced launch costs, but limited mass and power budgets restrict the utilization of state of the art components and subsystems to provide necessary capabilities: Especially attitude and orbit control functions are mandatory to enable precise formation flying. Miniaturized AOCS components designed for small satellites are indeed in focus of different research teams, but only have been rarely demonstrated in orbit. Also relative distance measurements are required to efficiently control formations, but advanced sensors and actuators are required to achieve the desired accuracy (and still, desired accuracy for missions like DARWIN are not realizable).
Beside hardware devices, proper control algorithms are needed to maintain a formation. Here, the individual satellites as well as the complete formation need to be controlled with respect to attitude and orbit, to keep the relative distances between the vehicles constant. For more details related to formation control refer to [Alfriend et al., 2010], [Wang and Hadaegh, 1996] and [Radice et al., 2010]. A major issue for formations as well as for swarms is communication in highly distributed networks [Raymond et al., 2004]. Especially inter satellite communication rises several problems in a highly dynamic environment, for example routing, data flow optimization and link interruption handling.

Self organization and autonomy

The term self organization is used in various research fields, for example computer science (decision algorithms, neural networks) and economics (socio-spatial systems), to describe a coordination process in complex systems [Unsal, 1993]. In technical

context summarizes self-organization functions or techniques, which increase the performance, efficiency or flexibility of (distributed) systems through autonomous behavior or reactions. Furthermore, self organization is an important issue for scalability and robustness. This can be applied to robot agents, sensor networks or satellites.
A distributed satellite system does not necessarily need self organization capabilities, for example does the GPS constellation contain more than 30 satellites, nevertheless are they individually controlled from ground. But with a growing number of vehicles, the effort for individual operation increases dramatically. Even with only a small number of assets in the network it might be necessary to include self organization techniques, if fast actions or decisions are desired, e.g. for collision avoidance. Only a few aspects of self organization in distributed satellite systems are highlighted here, more details can be found in [Wang and Hadaegh, 2000] [Pinciroli et al., 2008]. Self organization is an important topic in the context of communication, which needs to be adapted due to topology changes, transmission errors and link failures. Another important aspect is collision detection and avoidance in swarms or formations. A manually controlled satellite might not be able to detect a critical situation early enough to avoid the collision. Therefore, autonomous reactions should be implemented to guarantee safe modes, e.g. in close distance formations. Autonomous reactions are also desired in the case of failures: If a single satellite of the system can no longer provide a service, the distributed system should self organize to compensate the missing resource (soft degradation capability).

Operation of distributed satellite systems

Having a highly distributed formation or constellation in space also induces requirements for the ground segment. Missions with a large number of satellites impose a lot of work for operators. Considering a future space mission, which contains more than 100 nano-satellites, would increase the workload enormously. Commanding 100 different satellites individually from ground, which in case of a satellite cluster have almost the same contact times on a local ground station, is nearly impossible. Additionally, interferences between them might even more perturb proper mission operations. A possible solution could be a fully autonomous satellite network with no interaction of any human operator. Due to several reasons is a fully autonomous system not foreseen for the next years, at least an operator monitoring the scenario

to interact in failure cases is desired. Nevertheless, such a semi-autonomous strategy brings up new challenges and tasks for operation: Which satellites should establish a communication link with the ground stations? What happens if dedicated downlink satellites fail? How can data be collected from a distributed space system and transferred efficiently to a network of ground stations? All such issues are summarized in this work under the term "distributed satellite operation".
While section 2.4.1 is dedicated to general challenges of distributed satellite systems, this work contributes in the field of distributed small satellite operation. Especially problems from the field of scheduling and data management are handled to increase the efficiency and performance of distributed space missions. Scheduling issues, i.e. assignment of satellites to ground stations are handled in chapter 3. Aspects related to data management to coordinate data streams in ground station networks are discussed in chapter 4.

2.4.2 Technologies for distributed space missions

Enabling Technology

The ability to launch a satellite formation or swarm on a single launch vehicle was enabled by the decreasing mass and size of actual satellite platforms. On the other side, the small satellite concept suffers from limited capabilities due to mass and power restrictions. Even if the research focus of many teams lies on further miniaturization of satellite components, there is still a strong need for enabling technologies:
In nowadays pico and nano-satellite platforms two major bottlenecks origin from the mass and power restrictions: First, the Attitude and Orbit Control System (AOCS) contains not yet the capabilities of conventional large space vehicles. The tasks of the AOCS include attitude determination and control as well as orbit determination and control. For all these disciplines exist flight proven subsystem solutions, unfortunately not in the scale of a pico or nano-satellite: State of the art star trackers for highly accurate attitude determination are too large for small satellites, COTS propulsion systems for attitude or orbit control are to heavy for small satellites. Many researchers from the small satellite community try to overcome that issues by the development of miniaturized components and subsystems especially designed for small scale space vehicles. Meanwhile a variety of innovative systems are available even for CubeSats, including miniaturized star trackers, micro reaction wheels

or tiny propulsion systems (e.g. cold-gas thrusters or pulsed plasma thrusters). A dedicated investigation of enabling technology in the context of AOCS is given in [Schmidt and Schilling, 2010a].
The second above mentioned critical bottleneck is the communication subsystem. The limited power budget in combination with less accurate pointing accuracy led to the utilization of lower frequency bands (mainly UHF or VHF) in current satellite platforms. Furthermore, a broad spectrum of COTS radio amateur hardware is available and proved its performance in several missions. Main drawback is the low transmission rate provided from those COTS components, many pico and nano-satellites are restricted to 9600 bps or even less. Migration to higher frequency bands are planned, but ongoing activities are still in development phase. Most developers envisage S-band communication as possible technology for higher data rates, single missions already accommodated such systems [Minelli et al., 2008]. Furthermore, there was no Inter Satellite Link (ISL) demonstrated so far in a pico or nano-satellite, but especially for satellite formations is an ISL essential. More information about communication issues in small satellites are presented in [Marszalek et al., 2011] and [Klofas et al., 2009]

Communication protocols and IP in space

Beside the hardware devices for the physical communication link, also a suitable protocol for communication between network entities is needed to enable efficient data exchange, i.e. communication protocols need to be utilized. Of course network communication and protocols for distributed systems were addressed already in various other fields, but due to the special peculiarities of the space environment they can not directly be transferred. Migration of state of the art protocols to space context was therefore investigated extensively from many research groups.

An approach already tested in simulations and in real space missions is the utilization of Internet Protocol (IP) for space vehicles. When speaking about Internet protocols, one mainly refers to the TCP/IP protocol stack, which works on the network and the transport layer of the ISO/OSI reference model (see appendix B). The network layer is in all cases occupied from the Internet Protocol (IP), which guarantees the interoperability in the network. Beside the interoperability, the routing mechanism of IP made it so successful worldwide. The routing mechanism is in most cases not directly transferable for usage in space environment, due to the high

dynamics of the vehicles in orbit. But interoperability is a major argument for utilization of IP, as many other protocols use the IP interface to exchange information. A second, but quite simple reason to utilize IP in space is a commercial aspect: Designing a communication link is time intensive, referring to an existing solution reduces development time and costs (and IP is a good candidate as its interface is accepted worldwide as a standard). The last point is more or less the vision to have in future a global network, enabling the exchange of data between arbitrary nodes via IP (computers, mobiles, cars, space vehicles). This complies with the paradigm of having highly distributed space missions, each representing a small network in space, which is again connected to one global network. Of course this vision will not be realizable in near future, but the evolution of the Internet in the last years demonstrated very impressive the capabilities of highly distributed systems.

Problematic for the usage in space missions is in general not IP, rather connection oriented higher layer protocols. Especially the Transmission Control Protocol (TCP), occupying the transport layer, is susceptible to performance decrease. Main reasons for that are delays and transmission errors: The performance of TCP is restricted by its bandwidth delay product, requiring a transmission window buffer of equal or greater size [Hogie et al., 2005]. Therefore large delays decrease the maximum throughput. Nevertheless, for satellites in LEO this is not a problem at all, as the signal propagation delay is only a few milliseconds. Currently, small satellites are only used in LEO orbits and the delay is here not an issue for the application of TCP in space.

A second restricting factor of TCP is the misbehavior of the congestion avoidance algorithm in space. As there is currently no mechanism to distinguish loss due to congestion from loss due to noise, the flow control interprets the loss of an individual packet as a congestion situation and reduces the transmission rate. This means that increasing noise will reduce throughput as retransmission timeouts increase [Hogie et al., 2005]. To avoid the affects of the flow control, proper FEC algorithms need to be employed or a connection-less protocol like UDP has to be used.

Additionally, issues like path asymmetry and slow start performance should be mentioned in this context, for more information regarding IP in space with an extensive discussion of the influence on the complete ISO/OSI layer is given in [Hogie et al., 2005], [Akan, 2002] and [Steele, 2004].

To overcome the performance problems of TCP in space links, several protocol ex-

tensions were proposed. For example is a better delay handling implemented in the TCP Westwood Extension. Other extensions try to make TCP less sensitive to transmission errors, for example TCP/Peach. The concept of TCP extensions remedies some of the performance limitations, but the compatibility is not longer guaranteed. An survey on TCP extensions is given in [Durst et al., 1996].

First experiments with Internet protocols on satellite links were conducted in 1996 [Blott and N., 1996]. In the scope of the Operating Missions as Nodes on the Internet (OMNI) project a TCP/IP implementation was uplinked on UoSAT-12, a small satellite developed by SSTL [Jackson et al., 2001]. The first satellite solely relying for the communication on IP was the CHIPSat satellite from the University of Berkeley [Janicik and Wolff, 2003]. The IP protocol was here mainly employed to save development and implementation time. The provided FTP/TCP file transfer satisfied the needs of the mission operators [Hogie et al., 2005]. Advanced IP experiments were conducted in the scope of the Communication And Navigation Demonstration on Shuttle (CANDOS) mission, including protocols like SSH, SCP, FTP, MDP and Mobile IP [Israel et al., 2004]. The first experiments related to IP in space on a pico-satellite were conducted in the scope of the UWE-1 project (see section 2.1.3).

For all these mission experiments was static IP routing used (except for CANDOS), which is always problematic in highly dynamic network topologies. Solutions like Mobile IP are used in terrestrial scenarios to overcome the routing problem, a promising solution for space application are Mobile Ad-hoc Network (MANet) protocols, which were already utilized successful in robot formations [Zeiger et al., 2008] [Zeiger et al., 2009b] [Zeiger et al., 2009a].

Chapter 3

Redundant scheduling for ground station networks

One of the most limited resource of a satellite in a low Earth orbit is for sure the communication time. The contact window between a satellite and ground station can be determined from the orbit elements and the location of the ground station. For a LEO satellite are typically 6 to 8 [Cakaj et al., 2007] contact windows between 5 and 15 minutes available each day. This implies two major drawbacks: First, the satellite is only visible for a few minutes each day, which limits the amount of transferable data dramatically. Moreover, the ground station is not utilized for a large fraction of each day. Second, the contact windows have fixed start and end times due to orbit geometry and an academic ground station can serve only a single satellite at a time. Therefore, overlapping contact windows of individual satellites (see figure 3.1) lead to the question which of these satellites should be operated first.
To handle conflicts respectively overlapping contact windows, scheduling comes into play. The term scheduling describes in general the assignment of scarce resources to activities, with the objective to optimize one or more performance measures [Leung, 2004]. Throughout this work it is distinguished between planning and scheduling, even if in literature sometimes both fields become indistinct. Planning is a very general term and involves formulating a sequence of actions to achieve a desired objective, for example path or task planning. According to [Smith et al., 2000], scheduling is a special case of planning, where the actions are already chosen, leaving only the problem of determining a feasible order. This is consistent with the definition of scheduling in [Leung, 2004].

Many systems and applications make use of scheduling, e.g. process scheduling in

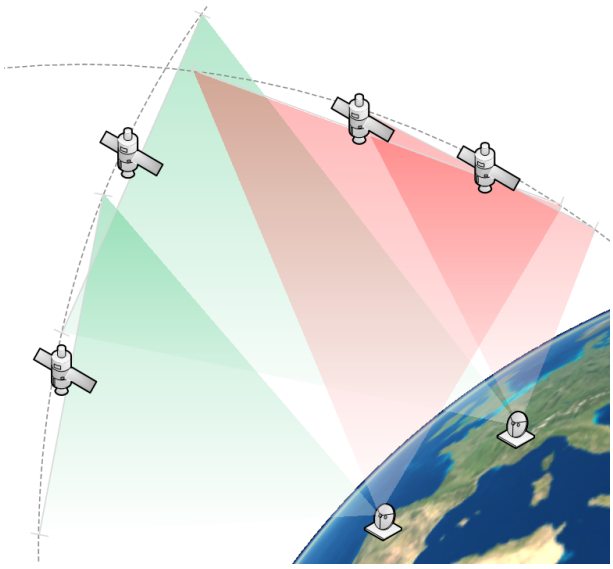


Figure 3.1: Overlapping contact windows for adjacent ground stations

operating systems or job scheduling in manufacturing processes. Within this work, scheduling is considered as the problem of finding an assignment of satellites to ground stations, i.e. to schedule contact windows in a given time horizon. This kind of problem is in general referred as Satellite Range Scheduling, or just SRS problem. The general aim in this context is to optimize the scheduling process for better utilization of restricted resources. In the case of academic ground station networks, scheduling follows the objective to share ground station resources and to extend communication time. In classic ground station networks, the main objective is rather to increase the utilization of the ground stations.

This chapter starts with the introduction of state of the art scheduling approaches used in current space systems. A new scheduling problem was derived from the demand of recent satellite missions and is presented in section 3.2. An approach tailored for academic ground station networks, which is able to satisfy the special requirements of distributed satellite missions, is proposed in section 3.3. The chapter concludes with an evaluation of the implemented scheduling system.

3.1 State of the art

In the following, several problems and systems are introduced, which deal with the problem of scheduling in space applications. Scheduling is in space context mainly used to assign contact or observation windows to one or more resources. Several studies related to optimal assignment of contact windows to ground station networks were conducted in the last years.
In certain environments is the process of scheduling still performed manually and is therefore very time consuming. It takes for example already 5 hours to create a preliminary schedule for a small number of tracking stations of the Air Force [Howe et al., 2000]. In larger networks, the creation and optimization of suitable schedules can take weeks. The presented approaches in this work focus exclusively on autonomous creation of schedules.
A variety of systems for automatic scheduling in space applications were already investigated in the past, the following sections concentrate on the satellite range scheduling domain. But also the Earth observation scheduling problem has similarities to the problem definition and will be introduced briefly.

3.1.1 Satellite Range Scheduling

The term scheduling is used on wide field of problems, in space context the Satellite Range Scheduling (SRS) problem was investigated in deep. It is defined as the generic problem of scheduling task requests for communication antennas, introduced from Schalck [Schalck, 1993]. A satellite can only communicate with a ground station when it is within the transmitting horizon of the ground station, depending on the orbit geometry this occurs periodically. Creating an efficient schedule is very sophisticated due to the huge number of combinatorial possibilities of satellites and ground stations. Furthermore, resolving conflicts of overlapping contact windows is a major issue. Also a problem is, that the SRS problem is typically oversubscribed. Oversubscribed scheduling means, that there are not enough resources available to satisfy all requests for resources, so a certain number of requests are suspended from the schedule.
The SRS domain can be further divided into the Single Resource Range Scheduling Problem (SiRRSP) and Multi Resource Range Scheduling Problem (MuRRSP), where SiRRSP describes the case of a single ground station (respectively antenna)

allocation to several satellites. The general problem of assigning a ground network to several satellites is contained in the MuRRSP domain.
It was shown from Burrowbridge, that the SiRRSP (i.e. only one ground station) can be solved in polynomial time when only LEO satellites are considered. Further details to the algorithm and a comparison of different scenarios can be found in [Burrowbridge, 1999]. In contrary, in the MuSRRSP domain (i.e. more then one ground station needs to be assigned), the contained problems are NP complete [Barbulescu et al., 2004c]. The scheduling problem in academic ground station networks can be ranked as MuRRSP problem, thus it is also NP complete.
Currently, there is no accepted state of the art algorithm for the SRS problem [Barbulescu et al., 2002], due to the unsteady performance of promising algorithms on individual problems. Two real world applications and its problem characteristics are discussed in the following, the oversubscribed AFSCN and AMC scheduling problems. Other relevant works in the scope of SRS domain are [Preindl et al., 2010], [Zufferey et al., 2008] and [Pemberton and Galiber, 2000]

Air Force Satellite Control Network (AFSCN)

An extensive investigated problem from the SRS domain is the AFSCN scheduling problem. The Air Force Satellite Control Network (AFSCN) is responsible for the communication needs between users on the ground and satellites in space [Barbulescu et al., 2002]. Key mission is to accommodate the requests from users into a consistent schedule. The AFSCN scheduling problem involves on average more than 100 satellites, which have to be assigned to 9 stations containing in total 16 antennas. Typically about 500 requests for communication windows are issued each day and need to be accommodated in a 24 hour schedule. Due to the restricted resources the problem is oversubscribed.
The main objective in the AFSCN scheduling system is to reduce the number of unsatisfied requests for a given day. Unsatisfied requests describe the number of requests, which can not be accommodated into the final schedule due to conflicts. A special characteristic of this scheduling domain is the distinction in low altitude and high altitude requests: A requested communication window for a low altitude satellite, i.e. LEO, is characterized by fixed start and end times, while the high altitude (GEO) communication windows can be scheduled more flexible, as GEO satellites are permanently visible.

First deeper investigation of that problem was performed by Gooley [Gooley, 1993] and Schalck [Schalck, 1993]. Both approached the scheduling problem with a Mixed Integer Programming (MIP) algorithm. Gooley reported with his system a ratio of successfully scheduled requests of at least 91 %. An extension of the approach presented from Schalck increased the ratio to a success rate between 96 % and 99 %.

For better comparison of the developed algorithms, a set of benchmark problems were defined [Barbulescu et al., 2007b], containing the requests submitted to the AFSCN administration in a 7 day period, each day occupied with approximately 300 requests. Barbulescu et al. [Barbulescu et al., 2004a] [Barbulescu et al., 2004b] performed an extensive comparison of search algorithms on the AFSCN benchmark problems, with focus on hill climbing, heuristic search and genetic algorithms. The first obtained results reveal strong performance of the genetic algorithm GENITOR [Barbulescu et al., 2002]. In further research, the effectiveness of large leaps in the search space was shown [Barbulescu et al., 2004c], which applies for the GENITOR algorithm as well as for the squeaky wheel optimization. The research of Kramer et al. [Kramer et al., 2007a] [Kramer et al., 2007b] compared, similar to the work of Barbulescu, search strategies for the AFSCN problem and derived hypothesis to explain their performance. Barbulescu at al. addressed later the issue of finding suitable benchmark problems for the AFSCN scheduling problem, which are needed to compare the performance of individual scheduling algorithms. Generating artificial benchmark problems is very difficult, since the produced problem instances have to reflect the complexity (e.g. level of oversubscription) of real world scenarios [Barbulescu et al., 2007b]. In summary, the following characteristics were shown for the AFSCN problem: Main difficulty in creating a schedule are the high altitude requests, because their flexible start and end times enlarge the search space exponentially. The conducted experiments show furthermore, that the performance of the individual algorithms strongly depend on the benchmark problem instances and the problem representation. The problem instances have to be chosen carefully to obtain expressive results, especially interesting is the result that problem instances from the real world scenario were solved from the split heuristic algorithm with a small number of unsatisfied requests, while artificial benchmark problem instances revealed for the same algorithm poor results [Barbulescu et al., 2002]. Nevertheless, problem generators are necessary to evaluate an algorithm statistically, a large num-

ber of problem instances is needed to obtain average values.

Air Mobility Command (AMC)

The scheduling problem faced by the USAF Air Mobility Command (AMC) is referred as the AMC scheduling problem, which is as well oversubscribed and very similar to the AFSCN scenario. Indeed, the AMC problem is not directly a satellite range scheduling problem, nevertheless it has very similar characteristics and the techniques applied to the AMC problem are transferable to problems in space [Kramer and Smith, 2004b]. Moreover is the AMC problem especially interesting, as it utilizes a priority scheme for task assignment. In the context of academic ground station networks is a priority scheme often desired, as the ground station resources belong to individual institutions, and those want to assign a higher priority to their own satellites than to other satellite missions.
The objective in the AMC scheduler is the allocation of resources to missions to minimize the no-productive flying time [Becker and Smith, 2000], or from a more theoretical point of view to increase resource utilization, which is equivalent to accommodate as many tasks as possible [Kramer and Smith, 2003]. Kramer and Smith presented a task swapping algorithm to solve the oversubscribed problems of the AMC domain and showed that the contained retraction heuristic [Kramer and Smith, 2003] is suitable for dynamic scheduling domains. Later they further improved the algorithms with respect to runtime and search efficiency [Kramer and Smith, 2004a]. A problem generator for the AMC domain was introduced in [Kramer and Smith, 2005]. Kramer concludes, that the proposed task swap algorithm shows good performance at comparatively moderate runtime. Unfortunately, the obtained results can't be transferred to similar scheduling problems like the EOS domain, due to different problem characteristics [Kramer and Smith, 2004b]. In a comparison between the very similar AFSCN and AMC scheduling domain it was shown that algorithm performance depends also on problem representation itself [Barbulescu et al., 2007a]. Small changes in the problem formulation change the performance of an algorithm significantly. The AFSCN and AMC domains differ by a priority scheme applied to the contact windows. It was shown that repair based algorithms outperform priority based algorithms in AFSCN, in the AMC domain the situation is vice versa, i.e. repair based strategies perform worse [Barbulescu et al., 2007a].

3.1.2 Earth Observation Scheduling (EOS)

Another scheduling problem faced in space context is the Earth Observation Scheduling (EOS) problem (or just observation scheduling problem). The objective of the EOS domain is to accommodate a number of predefined observation targets on a restricted number of resources (satellites). Because of the movement of the satellites, the targets are only observable during a time window, to satisfy all imaging requests from customers and researchers a schedule needs to be created. The daily management of one Earth observation satellite is already a challenging combinatorial optimization problem [Bensana et al., 1999]. The ground segment is often only indirect involved (as an additional constraint with respect to downlink capacity). Nevertheless, the problem is very similar to the SRS domain.

The EOS problem appears in different application fields, for example in commercial services, where imaging orders coming from clients need to be accommodated [Bensana et al., 1999] or in science observations, when astronomers request observation targets from space telescopes. Therefore, different variations of the EOS problem are described in literature, differing in modeling or complexity of constraints, a short survey is given in the following:

Wolfe and Sorensen compared the performance of three algorithms for the EOS domain. The EOS problem instances were modeled with a window constrained packing (WCP) problem, i.e. how jobs can be packed on a time-line in a factory schedule [Wolfe and Sorensen, 2000]. They used in their investigation a simple dispatch algorithm and a look-ahead algorithm in comparison to a novel genetic algorithm. They come to the general conclusion that the simple dispatch scheduler is relatively fast but obtains only poor schedules. The more sophisticated strategies (like genetic algorithms) deliver much better performance, but increase the runtime significantly. Therefore a tradeoff between performance and runtime needs to be defined.

A different aspect was handled from Bensana et al. for the daily management of a larger EOS scenario. In [Bensana et al., 1996] describes Bensana the scheduling problem of the Spot5 satellites, which need to take photographs from a predefined set of locations on Earth and are limited by a number of physical constraints. The scheduling problem is formulated here as a Variable Valued Constraint Satisfaction Problem (VVCSP). The main objective is to find a schedule which maximizes

the number of taken photographs, which are incorporated with different weights in an objective function. Bensana distinguishes between exact and approximated methods and compares for given problem instances a *tabu search*, *multi greedy* and *branch and bound* algorithm [Bensana et al., 1996]. He concludes that the exact methods are not applicable to large problems, approximate methods find solutions of good quality, but waste time to further improve the result to an optimal solution [Bensana et al., 1999].

A very similar problem from the EOS domain was investigated by Damiani et al. [Damiani et al., 2004] [Damiani et al., 2005a] [Damiani et al., 2005c]: An Earth watching satellite constellation consisting of identical satellites is proposed to detect and observe forest fires and volcanic eruptions. In contrast to Bensana, the focus is rather on planning methods. The goal is to realize an autonomous on board planning system which can satisfy the requirements of a global satellite detection system. The objective is not to create optimal schedules, rather to optimize the observation process by autonomous decision planning of the satellite agents [Damiani et al., 2005b].

From the field of telescope observation scheduling is especially the research work of Drummond [Drummond et al., 1994] [Drummond et al., 1995] and from Bresina [Bresina et al., 1995] [Bresina et al., 1996] relevant: Their work deals with management and scheduling of ground-based, remotely located, fully automatic telescopes. The objective is to accommodate observation requests submitted from astronomers for desired targets [Bresina et al., 1996]. The proposed strategy is to create an initial schedule with a dispatch algorithm, and to utilize the Heuristic-Biased Stochastic Sampling (HBSS) method [Bresina, 1996] for further improvement. Bresina showed, that the HBSS method outperforms greedy search for the investigated problem instances. Later he further optimized the schedule generation process by improving the search heuristic with the GenH technique [Bresina et al., 1997].

Globus et al. initially investigated for a constellation of Earth observation satellites the usage of genetic algorithms [Globus et al., 2002] and compared later genetic algorithms, *simulated annealing*, *squeaky wheel optimization* and a stochastic hill climbing search strategy [Globus et al., 2003]. A very interesting result is, that the genetic algorithm performs worse than the other investigated search algorithms on the underlying EOS problem. The results are in strong contrast to the results from the AFSCN problem, where the genetic algorithms outperform other algorithms [Globus et al., 2004]. This demonstrates the dependence on problem modeling, even

when the problem domains are very similar.
Further research related to scheduling in the context of the EOS domain can be found in [Lin et al., 2005], [Ruan et al., 2005] and [Wang et al., 2007].

3.1.3 ESA tracking stations - ESTRACK

Even for large ground station systems like the ESA ESTRACK system (see section 2.2.1) was planning and scheduling performed manually from operators until 2006 [Damiani et al., 2006], only supported by a set of tools, for example to build initial allocation plans. Meanwhile, a very sophisticated planning and scheduling system was implemented in order to coordinate the growing number of missions to be operated by the ESTRACK system. Due to the variety of supported satellite missions, ranging from Earth observation over interplanetary to high elliptic orbit missions, an automated planning system demands a variety of functionalities and services. Responsible for these tasks is the ESA Management System (EMS), which is further divided in the ESTRACK Planning System (EPS), ESTRACK Coordination System (ECS) and ESTRACK Scheduling System (ESS). Objective of the EPS planning system is the creation of a valid plan, which implements the Mission Agreements of all missions on a finite planning period. An extensive description of the different entities of the complete system can be found in [Damiani et al., 2006]. Damiani et al. use as basis for scheduling the mission needs of each supported mission and formulate them as a constraint network. The resulting Constraint Satisfaction Problem (CSP) contains temporal binary constraints, linear constraints and disjunctions of binary constraints. The consistency is checked by solving Disjunctive Temporal Problems (DTP) with a branch and bound approach, Linear Programming algorithms are utilized to manage remaining linear constraints. More information about the algorithmic part is given in [Hoffmann and Theis, 2009].
The first operational results obtained with the EPS system are promising: The used scenario for demonstration contained 7 satellites in a 10 days scheduling horizon. The resulting problem contains more than 325 contact windows creating more than 2800 constraints [Damiani et al., 2007]. The EPS system was able to solve the problem with 23 repair steps on the created schedule, but the consistency checks are very time consuming.
Unfortunately it is not possible to estimate the capacity of the system, i.e. the number of missions which can be supported, at the current stage, as it is still un-

der development. Furthermore is the modeling of the scheduling problem relatively complex, due to huge number of mission types involved in the ESTRACK system.

3.1.4 Summary

The problem of creating allocation schedules for satellite or ground station resources was investigated for different applications and scenarios in space context. Due to the complexity of such problems it is not possible to obtain an optimal solution in reasonable time. In general one has to make a tradeoff between solution quality and runtime. Another important point is, that the performance of an algorithm depends strongly on the problem formulation and structure. Some algorithms proved to obtain good results in a specific domain and fail to get comparable results in similar scheduling problems, so heuristics and strategies can not be transferred from one application to a modified one. There is no state of the art algorithm which delivers good average result on all scheduling problems in space context.
For the special needs of scheduling in low cost ground station networks was no dedicated investigation performed, yet. The GENSO network plans to use scheduling mechanisms to improve the utilization of the participating ground stations. In this scope compared Preindl [Preindl et al., 2010] different scheduling algorithms. He uses in his comparison a reservation system as basis, which restricts the efficiency of the overall system. A dedicated investigation for scheduling in academic ground station networks still needs to be done to better utilize available resources. To cope with the specific particularities of small satellite missions and highly distributed, low-cost ground station networks, a more detailed view on the scheduling requirements as well as the underlying problem structure is necessary.

3.2 Scheduling for low cost ground station networks

Various scheduling approaches for the SRS domain exist and were introduced in the previous section. Unfortunately they can not be transferred to the scheduling problem occurring in academic ground station networks. The main reason why state of the art algorithms are not suitable, is the fundamental differences in topology, application and organization (compare section 2.2.1). These differences result in scheduling objectives and requirements, which can not be satisfied from the current

available systems. A short example illustrates that: The UWE-2 satellite from the University of Würzburg was launched in September 2009 together with 3 other pico-satellites (BEESat, Swisscube and ITUpSat). All operators requested directly after launch to track their satellites to collect beacons for health monitoring. The operators did not specify how much communication time is requested for their satellites, they rather asked to track their satellites as often as possible. From the scheduling point of view this implies an important problem: The jobs (tasks) to be scheduled are not strictly defined, neither in number nor in length. Current approaches (see section 3.1) refer always to a predefined set of jobs for schedule creation. A second problem is, that those satellites were launched with the same launch vehicle and the ejection sequence was only a few minutes in total, therefore all satellites are in nearly the same position in orbit. Thus, all satellites have during the first weeks the same contact times to all ground stations.

This example illustrated the particularities in academic networks, a detailed elaboration of scheduling requirements and their differences to the classic scheduling approach is described in the next section. The mathematical formulation of the proposed problem is given in section 3.2.2 in terms of the Redundant Request Satellite Scheduling (RRSS) problem.

The policy of the operators, to request as much communication time as possible, is denoted as *redundant scheduling*: This term expresses that at least one contact window is desired per request from a user, more windows should be assigned if possible. Or in other words, one communication window is mandatory, redundant scheduled windows are welcome. The example visualized in figure 3.2 explains this strategy: The scenario contains three satellites, for each of them was a request issued. On ground station 1 are satellite 1 and satellite 2 in conflict (contact windows are colored red and blue). The classic scheduling approach would try to assign exactly one contact window to each satellite. For an academic ground station network this solution would not be satisfying, as more contact windows are available and not used. Therefore the remaining contact windows need to be assigned. The objective is now to avoid an arbitrary assignment of free windows to requests, rather should the redundant windows be distributed fairly between the individual requests. Figure 3.3 shows a situation where this was not done appropriately, satellite 2 received three contact windows while satellite 1 and 3 received only a single window each. A fairer schedule is depicted in figure 3.4, where two of the satellites received two

contact windows, the remaining window was assigned to the third satellite.

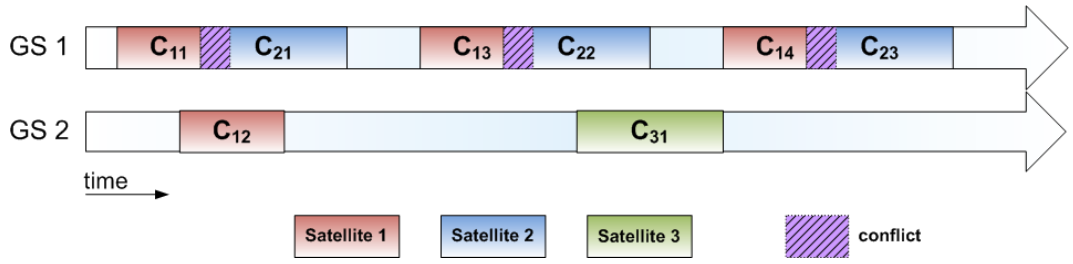


Figure 3.2: Redundant scheduling example scenario

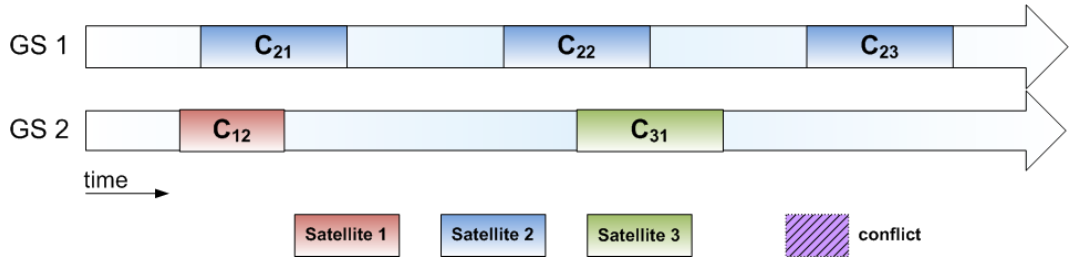


Figure 3.3: Unfair assignment of redundant contact windows

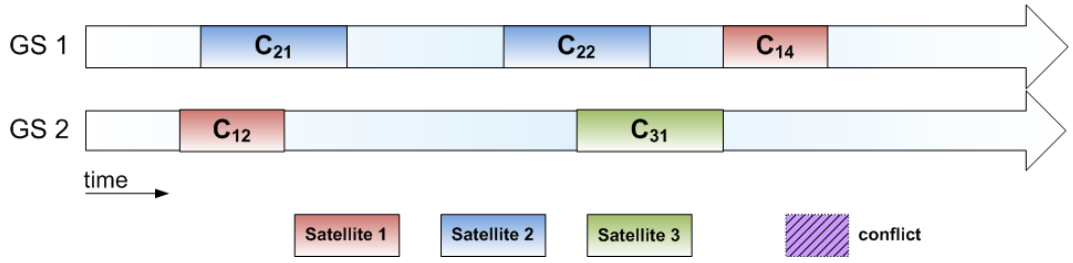


Figure 3.4: One example of a fair assignment of redundant contact windows

In summary, *redundant scheduling* means that the requests of the scheduling problem are not automatically satisfied when a single contact window for communication was assigned, it is desired to assign redundant windows if possible. Furthermore, a fair distribution of redundant windows should be achieved. The term *redundant scheduling* is used throughout this work to describe this special scheduling demand of academic ground station networks.

3.2.1 Scheduling requirements of academic ground station networks

In classic satellite range scheduling is the objective to increase the utilization of the ground network, this holds not necessarily for low cost ground station networks, here the objective is rather to extend the contact time with the satellites by sharing resources. The participants of these academic ground station networks are faced with the following problem: Very often the ground stations were built up in the scope of small satellite projects, but as these satellites are normally launched to low Earth orbit (LEO), the available communication time is restricted to a few minutes a day. By sharing the ground stations between two institutions, the contact time with the satellites can be doubled on both sides. This results in an important difference to classical ground station networks, which are sharing the ground stations with commercial interest, increasing the utilization is therefore necessary for economic reasons. Sharing ground stations in academic networks has no financial benefit, the educational and research aspects play the leading role.

Major emphasis in academic ground station networks is placed on flexibility, i.e. a flexible scheduling process is desired. Classical ground station networks on the other side have to create schedules for long time horizons, which results in a lot of work for operators, sometimes still working with paper to accommodate all requests in the schedules. A replanning is very complicated when such a schedule changes or a ground station failure occurs. In the field of academic ground station networks the situation is quite different: Most participants are associated with non-commercial institutions and also students are very often involved, so a more dynamic environment is present. A need for rescheduling in such an environment is very likely. The experience from the satellite projects at the University of Würzburg [Schmidt et al., 2007] showed this behavior many times, almost every day a ground station joined or left the network. Thus, the need for flexibility results in different scheduling objectives compared to classic ground station networks: It is more important to be able to reschedule in a fast manner than to find an optimal solution to a given scheduling problem.

As already pointed out before, a very critical aspect is the capability of *redundant scheduling*. This means to integrate redundant contact windows for all requests of the participants. In classic ground station networks a request is satisfied if exactly

one contact window of desired duration was assigned in the final schedule, the allocated time slot is enough to perform the suggested experiment or to download data. In academic ground station networks the standard scenario differs. Many small satellite developers want to have as much communication time as possible, especially in the first days of operation a seamless monitoring of the satellite is necessary, but this is hardly realizable due to contact windows of only a few minutes. By sharing of ground stations it is possible to retrieve data each orbit. Therefore, many satellite operators in the small satellite community formulate their request as "every free station should track my satellite". This behavior was introduced before as *redundant scheduling*. Classic scheduling algorithms in the SRS domain totally neglect this aspect. Classical ground station networks try to satisfy a scheduling request by assigning exactly one contact window as specified in the request.
Furthermore, the commercial character of classic ground networks obliges the network provider to maximize the number of satisfied requests. In academic ground station networks the participants share resources to have access to their satellites. Thus, it is not necessary to satisfy as many participants as possible, rather the resources should be shared equally. Especially when redundant scheduling is desired, a fair distribution of assigned contact windows should be assured.

3.2.2 Problem description

The problem of ground station scheduling can be simply described with a few words: To find an optimal assignment of ground stations to a number of satellite requests, submitted by different users. This very vague description will be more formalized in this section. The entities of a problem instance are ground stations, users and satellites, which are related to each other through communication requests. The most important entity and central to this problem description is the definition of a communication request R (further only referred as request), it describes the request of an user U for a contact window with a satellite S. The ground station, on which this contact takes place is not defined from the user beforehand, the ground station network itself guarantees seamless data flow through the connected ground stations (for example over Internet). Therefore, a request R_i is clearly defined by

$$R_i = \{S, U, t_s, t_e, dur, Rd\} \tag{3.1}$$

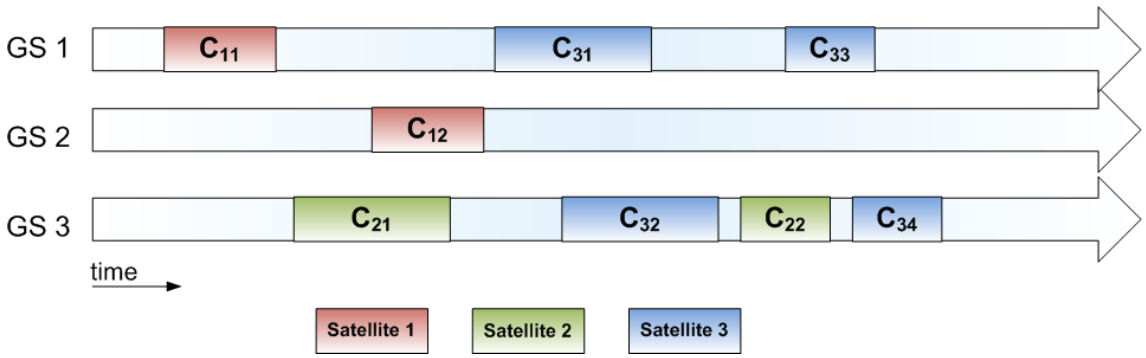


Figure 3.5: Calculated contact windows C_{ij}

with the index i numbering consecutively all requests. The parameters of a request R_i are the requested satellite S, which is moving on its orbit, an user U, who submitted the request, an earliest start time t_s and a latest end time t_e, which describe the time horizon desired from the user (for example in the next two days). The duration *dur* describes the minimum required length of the requested contact window (in minutes), i.e. this length determines if an available contact window is suitable, for example to perform a specific experiment. The intention of the parameter *dur* is to exclude too short contact windows from the schedule, this could be necessary when a command needs to be send to the spacecraft and a certain length of the contact window has to be guaranteed to ensure a safe telecommand within that time interval. If only beacons should be collected from a satellite, the parameter *dur* can be set to a minimum value, which qualifies all available contact window for assignment. The last parameter *Rd* describes the maximum degree of redundancy and determines how many contact windows should be assigned at maximum. As explained in section 3.2.1, the aspect of including redundant contact windows is essential for this problem formulation. It is possible to define an upper bound for the number of redundant scheduled contact windows. If no upper bound is defined, it is possible to assign as many contact windows as available to a given request. The possible values for *Rd* are defined as

$$Rd = \begin{cases} n \in N_+ \text{ maximum of n contact windows} \\ -1 \text{ if no upper bound is set} \end{cases} \tag{3.2}$$

All basic entities of the problem description, i.e. users, ground stations, satellites and requests, are associated with priorities. A priority scheme was introduced to

weight the contribution of an user or satellite to the ground station network. In the further text is the priority P marked with a subscript for the corresponding component, so P_S stands for the priority value of the satellite, P_R for the request, P_G for the ground station and P_U for the user respectively. With these definitions of requests, satellites and users it is possible to calculate all available contact windows C_{ij}. Each request R_i has a certain number of contact windows associated, the contact windows are numbered with the subscript j. Each request has a start and end time (t_s and t_e), the orbit geometry and the location of the ground station determines when a contact is possible. The j contact windows of a request R_i are distributed over several ground stations, a contact window C_{ij} is defined through

$$C_{ij} = \{R_i, t_{AOS}, t_{LOS}, G\} \tag{3.3}$$

Each contact window C_{ij} has an associated request R_i and can be used to accommodate a time interval of *dur* minutes. Furthermore is C_{ij} only valid for ground station G. The values t_{AOS} (acquisition of signal) and t_{LOS} (loss of signal) describe the time interval when contact between the satellite S and ground station G is possible. It should be emphasized that t_{AOS} and t_{LOS} are not the same as the start and end times t_s and t_e of the request: For example, a request could be related to a 10 minute contact time in the time from now (t_s) until next week (t_e), an appropriate contact window C_{ij} could start tomorrow morning (t_{AOS}) and end 15 minutes later (t_{LOS}). Figure 3.5 illustrates a scenario with 3 satellites, the contact windows are distributed over 3 ground stations. The first index describes the associated request (e.g. all blue contact windows belong to request number 3), the second index numbers consecutively the contact windows belonging to the same request. The calculation of these contact windows can be performed automatically with an orbit prediction software (for example *STK* [1] or *predict* [2]). After calculation of the available contact windows C_{ij}, the objective is to find an assignment of these contact windows to the requests R_i. The difficulty originates from the fact that normally many contact windows overlap, i.e. are in conflict with each other. This assignment is not trivial and an algorithm to find an optimal solution in reasonable time is not known, due to the NP-complete characteristic of that problem [Barbulescu et al., 2004b].
An important issue, which should be emphasized again, is the assignment policy for

[1] http://www.agi.com/products/by-product-type/applications/stk/
[2] http://www.qsl.net/kd2bd/predict.html

the contact windows C_{ij}. In classical ground station networks, a request is satisfied as soon as one contact window out of the j available contact windows C_{ij} was included in the final schedule. In the RRSS problem domain the aim is to include more than one contact window (if possible) in the final schedule.

3.2.3 Scheduling objective

The scheduling objective of the introduced RRSS problem is twofold: The first objective (similar to the SRS domain) is to minimize the number of unsatisfied requests, i.e. as many requests as possible should be included in the final schedule. That is reasonable, as contact time to a satellite is a valuable resource and should be utilized as efficient as possible. An important difference to other scheduling problems from this domain is an integrated priority scheme. In the scope of academic ground station networks share many participants their resources free of charge. Nevertheless, each participant wants at least his own satellite reliably tracked from a ground station, which can be resolved with the assignment of a higher priority. Additionally, one could think of higher priorities for participants contributing with a ground station, participants which use only the network without bringing own resources to the network could get a lower priority. Hard priorities are not required, soft priorities satisfy the needs of small satellite missions.
The second objective deals with the distribution of redundant contact windows. The special property of the RRSS problem is the desired equal distribution of redundant contact window (compare section 3.2). Hence, additionally to including as many contact windows as possible it is also important to assign the ground station resources in a fair way. Both objectives are integrated into an objective function γ, which is used to compare the quality of individual schedules with each other or to optimize the schedules to a given criteria.
As already pointed out in the previous section, the focus is to find a solution to the specified objective function within reasonable time. It is rather important to create a new schedule within a short time frame, to react on sudden changes in the network topology, than to create an optimal solution for a given scenario. The runtime aspect is not included in the objective function γ, but it is used at the end of this chapter as a criterion for evaluation.

3.2.4 Classification

To distinguish different kinds of scheduling problems, they are grouped in problem classes. A common notation to describe scheduling problems respectively classes was introduced by Graham et al. [Graham et al., 1979]. The notation consists of three fields and is primarily used to classify theoretical scheduling problems. The notation has the form $\alpha \mid \beta \mid \gamma$, where

- α describes the machine environment and contains only a single entry
- β provides details about job characteristics and scheduling constraints. This field can contain several entries
- γ contains the objective function to optimize

The machine environment α specifies number and type of available processing units, e.g. *single machine*, *Parallel and Identical Machines*, *Job Shop*, *Flow Shop* etc. Possible job characteristics of the β field could be *Preemption*, *Release Dates*, *Processing Times* etc. The objective function γ describes an optimization variable, like minimizing the *Maximum Lateness* or *Total Completion Time*. For an extensive survey of scheduling classes and further parameters of the Graham notation please refer to [Leung, 2004].
In literature, the classic SiRRS satellite range scheduling problem (compare section 3.1.1) is considered as a $1 \mid r_j \mid \sum U_j$ problem, i.e. jobs are processed on a single machine, release time of each job is denoted by r_j and the objective is to reduce the number of not scheduled jobs ($\sum U_j$) [Barbulescu et al., 2004b]. It is known that the decision version of the $1 \mid r_j \mid \sum U_j$ problem is NP-complete and therefore it can be shown that also the MuRRS case is NP-complete, as SiRRS is a subproblem of MuRRS with R_m unrelated machines ($R_m \mid r_j \mid \sum U_j$).
The introduced RRSS problem in section 3.2.2 complies with respect to machine environment and job characteristic with the classification of Barbulescu, main difference is the objective function. Additionally to minimizing the number of not scheduled jobs are priorities introduced, which act as weights w for requests and satellites to favor individual participants (see explanation in previous section). Furthermore, a second objective is introduced, which is responsible for an equal distribution of redundant contact windows. The penalty value P is added if redundant windows are not equally distributed. In this case the introduced RRSS problem can

be denoted as $Rm \mid r_j \mid \sum w_j U_j + P$.
However, the Graham notation is a bit problematic when applied to satellite range scheduling. The principle difficulty is the assignment of an appropriate processing time to each job. For a satellite can the processing time be calculated from the amount of data to be transferred to Earth. The fixed transmission rate of the communication system determines the average time needed to transfer that data. Unfortunately, this time can not be used in the same sense as the processing time in the Graham notation, as the orbit geometry of the satellites restrict the assignment to an arbitrary ground station. Hence, it is in general not possible to exchange two equal ground station resources, as the visibility to the satellites differs (except when both ground stations are located at nearly the same longitude and latitude). The notation of Graham is here only used to refer to the complexity of equivalent problems.
Also important in this context is the formulation of the scheduling problem. Very common is the problem representation as a Constraint Satisfaction Problem (CSP), which comes from the field of Constraint logic Programming (CP) to solve combinatorial optimization problems. A CSP consists of a given set of variables together with finite domains and a set of constraints. Each constraint is related to a subset of the variables, the objective is to satisfy all constraints. A detailed survey on CP can be found in [Leung, 2004]. The implementation of the proposed scheduling approach uses also a CSP representation of the objective function γ, more details are presented in section 3.3.6. Open source solvers and libraries dedicated to CSP solving are available.

3.3 Scheduling approach

The previous described RRSS problem was derived from the scheduling needs of current small satellite projects. To satisfy these needs, a comprehensive system was designed and implemented to create consistent schedules for real world scenarios. This section focuses rather on the scheduling approach for the RRSS domain, than on the implemented scheduling system. To express the origin from the small satellite environment, the scheduling system was called *CUbesat Scheduling System* (*CUSS*). The heart of the *CUSS* system is the objective function γ, which is used to evaluate the performance of a given schedule. The novelty of the proposed scheduling ap-

proach is the capability of *redundant scheduling*. Therefore only the relevant parts, the problem formulation (see section 3.2) and the objective function (see section 3.3.2) are handled in detail within this work. Nevertheless, a short overview on the complete system is given to better illustrate the relationship between the different components of the *CUSS* system. The utilized search algorithms are explained in section 3.3.5 and a few important implementation details are discussed in 3.3.6.

3.3.1 System overview

The *CUSS* scheduling system is divided in different modules, but in principle is only the core module needed to create for a given problem a corresponding schedule. Additional modules were developed for visualization and to support operators (see block diagram in figure 3.6). The core module gets as input a number of contact windows and a list of requests. The contact windows are clearly defined by the satellites and ground stations (c.f. problem description in section 3.2). This input defines the search space of a problem instance. Further integral parts of the core module are the search algorithms and the already mentioned objective function γ.

An important component required is the orbit predictor module: The calculation

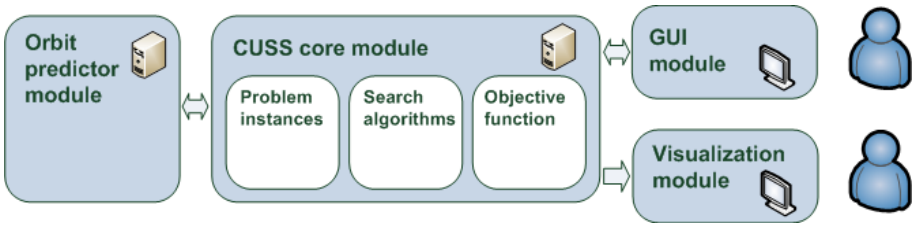


Figure 3.6: Block diagram of CUSS scheduling system

of contact windows, defined by the satellite orbits and the ground station locations, is not performed by the *CUSS* core module itself, it relies on open source software and standardized orbit element sets (more details about the format are presented in the implementation section 3.3.6).

Additionally, there is a module provided for visualization and a GUI module for facilitating the operation. The visualization module is only intended to show the obtained result to a human operator (see figure 3.6). The GUI module is needed to control and parametrize the schedule creation process, for example by choosing the desired search algorithm or to modify the priority weights. Both, the visualization

and GUI module, are not subject to this work and are not further addressed, for more information to those refer to [Schmidt et al., 2008].

3.3.2 Scheduling objective function

The objective of the *CUSS* scheduling system is defined as a cost function γ (see equation 3.4), which is to be maximized. This cost function calculates a value for a given schedule σ, consisting of all calculated contact windows C_{ij}. Each available contact window C_{ij} has an attribute, which determines if it is part of the final schedule. The cost function γ contains two terms, γ_1 and γ_2, representing two different scheduling objectives.

$$\gamma(\sigma) = \gamma_1 - \gamma_2 \tag{3.4}$$

γ_1 is the weighted sum of assigned priority values. Maximizing γ_1 means to include as many contact windows C_{ij} as possible, it is defined as

$$\gamma_1 = \sum_{\forall C_{ij}} \left(\pi(C_{ij}) \cdot C_{ij}^b\right) \tag{3.5}$$

with C_{ij}^b specified as

$$C_{ij}^b = \begin{cases} 1, \text{ if } C_{ij} \text{was integrated into the final schedule} \\ 0, \text{ otherwise} \end{cases} \tag{3.6}$$

and

$$\pi(C_{ij}) = w_R \cdot P_R + w_G \cdot P_G + \ w_S \cdot P_S + w_U \cdot P_U \tag{3.7}$$

The function $\pi(C_{ij})$ calculates the total priority value from the weighted entity priorities (P_R, P_G, P_S, P_U). Thus, it is possible to control the contribution of the different scheduling entities in a very fine grain. For this work the weights (w_R,w_G,w_S,w_U) were chosen to a value of 1.

Maximizing γ_1 implies to prefer high priority contact windows, low priority requests are only included if not overlapping with higher priority requests. As the problem formulation states, it is possible that a request gets more than one contact window assigned. Therefore, it could be possible that one request receives all available contact windows and a conflicting request not even one, even if both request could be accommodated in the final schedule. The problem is, that redundant

scheduled contact windows of high priority requests will preempt contact windows of low priority requests. To avoid this "unfair" distribution of contact windows, the γ_2 term is introduced: The aspect of fairness, here interpreted as an equal distribution of contact windows for requests, is integrated by subtraction of γ_2 (see equation 3.4). So, γ_2 can be interpreted as a penalty for unfair distribution of redundant contact windows. In this way are situations avoided, where two equal requests are treated unfair with respect to overlapping windows. The scenario depicted in figure 3.2 shows a situation where two satellites fly almost in the same orbit shortly after each other, which results in conflicting contact windows each revolution at ground station 1 (GS 1). If both satellites have the same priority, the γ_2 term guarantees that the associated requests receive the same amount of contact windows. The γ_2 term is defined as:

$$\gamma_2 = \sum_{1 \leq k \leq i} \left(\lambda^{\kappa(R_k)}\right) \tag{3.8}$$

where λ is a positive integer > 1 and the function $\kappa(R_k)$ is defined as

$$\kappa(R_i) = R^b_{max} - R^b_i \tag{3.9}$$

$$R^b_i = \sum_j (C^b_{ij}) \tag{3.10}$$

R^b_i describes the actual number of assigned contact windows C_{ij} for request R_i in a schedule. The value R^b_{max} describes the maximum number of available contact windows of a single request, with respect to the complete set of requests $R_1, .., R_i$. So R^b_{max} is a specific constant for a set of given requests: Or in other words, from the set of given requests (R_1,..,R_i), the request having the maximum number of contact windows available, that value is defined as R^b_{max}. For better illustration, an example is given below. Therefore, it is obvious that $\kappa(R_i)$ will be always ≥ 0 as the actual amount of assigned contact windows of R_i will never be greater than the maximum amount of available contact windows over all requests.

So far not mentioned is the parameter λ, which influences the "importance" of the γ_2 term on the overall objective function γ. If λ is selected as a small integer, the objective to distribute the redundancy equally between requests has only small influence, γ_2 will be small in comparison to γ_1 . But if λ is set to a large integer, the

objective to equally distribute redundancy overwhelms the γ_1 objective. Therefore a tradeoff between these two objectives has to be found, in the experiments was a suitable value of λ empirically determined as 3.

Example

To illustrate the introduced terms and definitions a short example is given: Lets con-

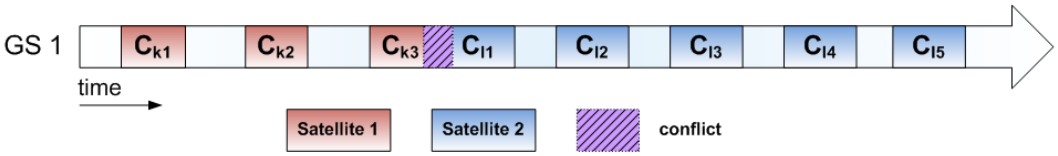


Figure 3.7: Example with two requests

sider two request, R_k with 3 possible contact windows (C_{k1}, C_{k2}, C_{k3}) and another request R_l with 5 possible contact windows $(C_{l1}, C_{l2}, C_{l3}, C_{l4}, C_{l5})$. One conflict can be observed between R_k and R_l, the contact windows C_{k3} and C_{l1} are overlapping and only one of these two contact windows can be assigned in the final schedule (see figure 3.7). In this scenario the constant R^b_{max} is equal to 5, as request R_l has the maximum number of 5 available contact windows. For the final schedule two options are possible:

Schedule 1: R_k gets all of his 3 available contact windows assigned, so $R^b_k = 3$ and $\kappa(R_k) = 5 - 3 = 2$. Due to the conflict in one of the contact windows, only 4 windows can be assigned to R_l, which results in $R^b_l = 4$ and $\kappa(R_l) = 5 - 4 = 1$ (see schedule option 1 in figure 3.8). The penalty value for this schedule would then be calculated as $\gamma_2 = \lambda^2 + \lambda^1 = 12$.

Schedule 2: The conflict will be resolved with the other option, R_k receives only 2 contact windows, therefore 5 contact windows will be assigned to R_l (see schedule option 2 in figure 3.8). The parameter R^b_{max} is again equal to 5 (this value is a constant for a given set of requests). Now one can calculate $\kappa(R_k) = 5 - 2 = 3$ (only 2 assigned contact windows for $R_k \rightarrow R^b_k = 2$) and $\kappa(R_l) = 5 - 5 = 0$ (because 5 assigned contact windows for $R_l \rightarrow R^b_l = 5$). As a result, γ_2 is determined as $\lambda^3 + \lambda^0 = 28$. The value of γ_1 is neglected in this example, assuming equal priorities for all requests, satellites and ground stations and considering the same amount of contact windows assigned (7 in total), which leads to an equal value of γ_1 for both

scheduling alternatives.
So in this scenario it is clear that schedule option 1 is preferred, due to its lower penalty value γ_2. In the further section will be analyzed, if the penalty function γ_2 is suitable to achieve an equal distribution of redundancy in general. In the next

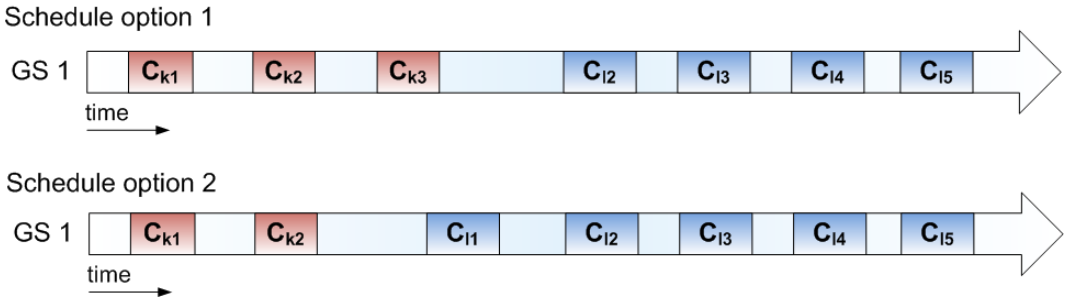


Figure 3.8: Schedule options for the example

section it is shown, that the penalty function γ_2 can be used for any combination of two requests R_i and R_j, i.e. that γ_2 will be in any case minimal if the redundant contact windows are equally distributed.

3.3.3 Behavior of the penalty function for two arbitrary requests

The aim of this section is to prove that the penalty value γ_2 is minimal, if two requests R_i and R_j have equally distributed redundant contact windows. Two arbitrary requests R_i and R_j are assumed, with a number of n, respectively m, contact windows available. The proof can be divided in two distinct cases:
Case 1: Both requests R_i and R_j have no overlapping contact windows. This case can be handled quite simple, because the absence of conflicts means, that the maximum amount of contact windows can be assigned ($R_i^b = n$ and $R_j^b = m$, both are maximal), therefore $\kappa(R_i)$ and $\kappa(R_j)$ are both minimal. Thus, also the result of the penalty value γ_2 is minimal.
Case 2: Both requests R_i and R_j have at least one conflict, i.e. overlapping contact window. Thus, for a conflicting pair of contact windows, only one of these will be assigned in the final schedule. Lets assume w.l.o.g. that the request R_i gets in total k contact windows, which leads to $\kappa(R_i) = R_{max}^b - k = a$. Request R_j on the other

side receives l contact windows, resulting in $\kappa(R_j) = R^b_{max} - l = b$, with $a, b \geq 0$. The penalty for this assignment is calculated as

$$\gamma_2 = \lambda^a + \lambda^b \tag{3.11}$$

When the assignment of contact windows is changed by transferring exactly one conflicting contact window to R_j instead of R_i, the penalty function will change to

$$\gamma_2 = \lambda^{a+1} + \lambda^{b-1} \tag{3.12}$$

The explanation is, that an additional contact window for R_j decreases the κ value by 1 (This was also the case for the example in section 3.3.2, where the penalty altered from $\lambda^2 + \lambda^1$ to $\lambda^3 + \lambda^0$). Equations 3.11 and 3.12 are now used to derive how the penalty function behaves in general, when the contact window assignment is changed. The following inequality is the starting point:

$$\lambda^a + \lambda^b < \lambda^{a+s} + \lambda^{b-s} \tag{3.13}$$

It will be shown here, that for a given schedule the penalty increases, when the distribution of contact windows is changed in an unfair assignment. This is expressed in equation 3.13, where the penalty value of a schedule should be greater ($<$), if an unfair distribution of contact windows was created by removing a contact window from request R_i to R_j. Furthermore it will be proven, that equation 3.13 holds, if a fair distribution of redundant contact windows was integrated in the schedule. In the first step, the inequality in equation 3.13 is modified to

$$\lambda^a \cdot (1 - \lambda^s) < \lambda^b \cdot (\lambda^{-s} - 1) \tag{3.14}$$

furthermore holds

$$\lambda^{a-b} \cdot (1 - \lambda^s) < (\frac{1}{\lambda^s} - 1) \tag{3.15}$$

and finally is formed to

$$\lambda^{a-b} \cdot (1 - \lambda^s) < (\frac{1 - \lambda^s}{\lambda^s}) \tag{3.16}$$

the term $1 - \lambda^s < 0$, as the parameters $\lambda > 1$ and $s > 0$. The last step is

$$\lambda^{a-b} > \frac{1}{\lambda^s} \tag{3.17}$$

To show under which condition inequality 3.17 is satisfied, one has to distinguish three further cases:

Case 2.1: $a > b$. Then inequality 3.17 is always true, because $\lambda > 1$ and $s > 0$. This means, if request R_i has less contact windows assigned than R_j (which is equal to $a > b$), then a further decrease of contact windows for R_i (is equal to an increase of a) leads automatically to a higher penalty, due to the simultaneous increase of contact windows for R_j. Or in other words, if R_i has already less contact windows than R_j, the distribution will be even more unfair when additional contact windows will be assigned to R_j instead to R_i

Case 2.2: $a < b$. Assuming that a is smaller than b, equation 3.17 can be modified to

$$\frac{1}{\lambda^{b-a}} > \frac{1}{\lambda^{s}} \tag{3.18}$$

which holds for the case that $a - b < s$. This means, that the amount of s contact windows can be assigned to R_j instead R_i without an increase of the penalty value. $a < b$ means, that R_j has less contact windows assigned than R_i, i.e. the redundancy is not yet equally distributed. By moving one contact window from R_i to R_j, the schedule becomes more fair, the penalty value decreases.

Case 2.3: $a = b$. Then inequality 3.17 is again always true, as $1 > \frac{1}{\lambda^s}$. So, if R_i and R_j have an equal amount of contact windows assigned, any further rearranging of assigned contact windows will raise the penalty due to unfair distribution of contact windows.

It was shown in this section, that the penalty function γ_2 can be used to prevent an unfair distribution of redundant contact windows. A minimal value for γ_2 is always achieved, when redundant contact windows are equally assigned over two requests.

3.3.4 Redundancy distribution for more than two arbitrary requests

Before proving the correct behavior of the penalty function for more than two requests, a formal definition of *equal distributed redundancy* with respect to a schedule is necessary. When considering only two requests, an equal distribution of redun-

dant windows is simply achieved by balancing the number of assigned, conflicting contact windows. But how can equal distribution be assessed for more than two requests? This is achieved by introducing the term *distance*: The *distance* of schedule σ (see equation 3.19) refers to the maximum number of assigned contact windows r_{max} (with respect to all requests from schedule σ) minus the minimum number of assigned contact windows r_{min}(considering all requests from schedule σ).

$$Dist(\sigma) = r_{max} - r_{min} \tag{3.19}$$

$$r_{max} = \max\left(R_i^b, \forall i\right) \tag{3.20}$$

$$r_{min} = \min\left(R_i^b, \forall i\right) \tag{3.21}$$

Using this equation makes it possible to define that the redundancy of schedule σ_1 is more balanced than schedule σ_2, if distance of schedule σ_1 is smaller than the distance of schedule σ_2

$$Dist(\sigma_1) < Dist(\sigma_2) \tag{3.22}$$

As this definition seems not so clear from the first impression, an example is given for illustration.

Equal redundancy distribution for an example scenario

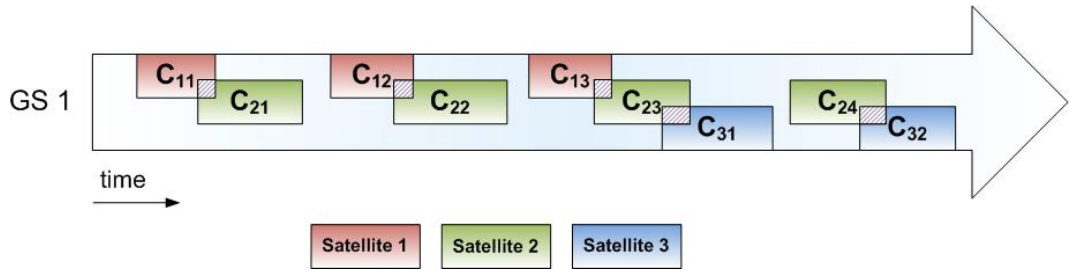


Figure 3.9: Example Scenario with 3 requests

The scenario in figure 3.9 depicts a situation with three requests. The conflicts between these requests are expressed with the overlapping, hatched areas. Therefore, it is not possible to assign all 9 available contact windows. Two schedules, which

could be a result from the scheduling process are shown in figures 3.10 and 3.11. It can be seen quite easily, that the amount of assigned contact windows for schedule σ_1 and σ_2 are equal (5 in total). The question is now, which of these schedules better satisfies the criterion of equal distributed redundancy? From the upper definition of *distance* this is the case for schedule σ_2, because $Dist(\sigma_1)$ can be calculated as the maximum amount of contact windows $r_{max} = 3$ (satellite 2) minus the minimum amount of contact windows $r_{min} = 1$ (satellite 1 and 3), so $Dist(\sigma_1) = 3 - 1 = 2$. The same evaluation for schedule σ_2 leads to a value of $Dist(\sigma_2) = 2 - 1 = 1$ (satellite 1 and 3 have two contact windows assigned). Therefore, schedule σ_2 has a fairer distribution of redundant contact windows. It has to be shown now, that the penalty value is minimal, if the distance in a schedule with more than two requests is minimal.

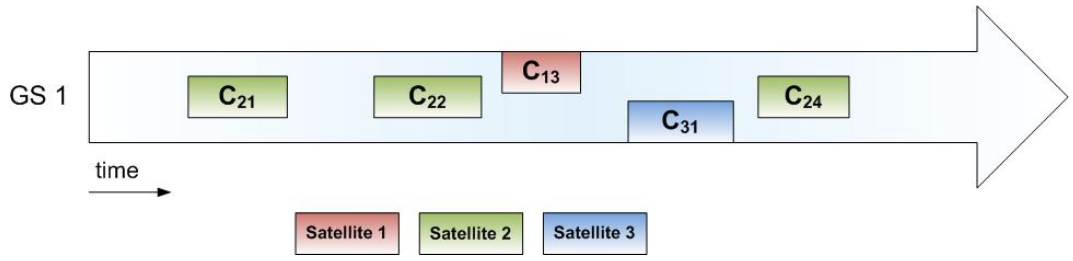


Figure 3.10: Schedule σ_1

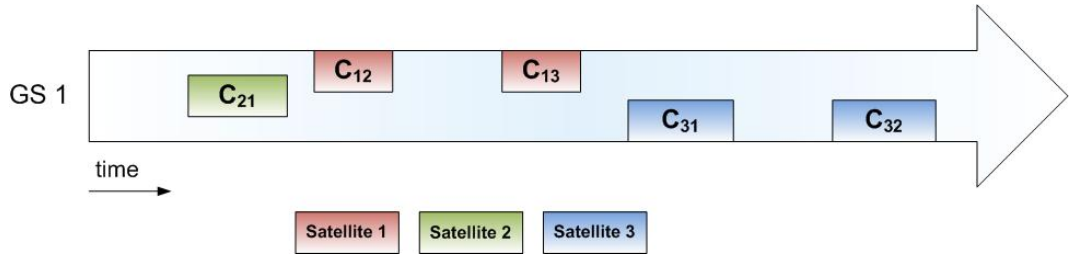


Figure 3.11: Schedule σ_2

Relation between the penalty function and distance

With the mathematical formulation of the term *distance* (see equation 3.19) it will be proven now, that the penalty function is minimal, if the *distance* of a schedule is minimal, i.e the redundancy is equally distributed for a given problem instance with

more than two requests. In a real world scenario, each orbit results in an unique pattern of available contact windows, thus the number of available contact windows for each request is different. The interesting question is then, how the total amount of assigned contact windows interacts with the objective to equally distribute redundant contact windows. Therefore, two cases are distinguished to prove that the objective function behaves correct, i.e. it is minimal if the redundancy is equally distributed between the requests:

Case 1: Schedules σ_1 and σ_2 have the same total amount of n assigned contact windows. To prove that the penalty function behaves correctly, it has to be shown that the penalty value of schedule σ_2 is smaller than the penalty value of schedule σ_1, if the distance of schedule σ_2 is smaller than the distance of schedule σ_1. Mathematically formulated:

$$\gamma_2(\sigma_2) < \gamma_2(\sigma_1) \tag{3.23}$$

$$\text{if} \quad Dist(\sigma_2) < Dist(\sigma_1) \tag{3.24}$$

These equations express the intuitive expectation, that the penalty value is smaller, if the redundancy is fairly distributed, i.e. the distance between the schedules is smaller.

Case 2: Schedule σ_1 has in total n contact windows assigned, Schedule σ_2 has m contact windows assigned, with $n \neq m$. For this case it has to be shown, that there exists at least one penalty value for schedule σ_2, which is smaller than the penalty value of schedule σ_2, if $n < m$. Expressed in equations:

$$\exists\gamma_2(\sigma_2) < \gamma_2(\sigma_1) \tag{3.25}$$

$$\text{if} \quad \sum_i \kappa(R_i) < \sum_j \kappa(R_j) \tag{3.26}$$

Note: Equation 3.26 expresses, that schedule σ_2 has more contact windows in total assigned than schedule σ_1. Intuitively one expects, that the penalty value for

schedule σ_1 should be greater, as it has less contact windows in total assigned. But having more contact windows assigned does not necessarily imply a fair distribution of redundant contact windows, for example could many conflicts in the calculated contact windows result in a very unfair redundancy distribution. But when the conflicts are neglected, there should exist at least a schedule σ_2, which has a smaller penalty value then schedule σ_1, as it has more contact windows assigned than σ_1.

Proofs

This section proves the correct behavior of the penalty function for the two cases above:

Case 1: If $Dist(\sigma_2) < Dist(\sigma_1)$, then the penalty value $\gamma_2(\sigma_2) < \gamma_2(\sigma_1)$:

This proof uses the fact, that a decrease of $Dist(\sigma_1)$, automatically results in a decrease of the penalty $\gamma_2(\sigma_1)$. The penalty value of schedule σ_1 can be calculated by

$$\gamma_2 = \lambda^{a_1} + \lambda^{a_2} + \lambda^{a_3} + ... + \lambda^{a_n} \tag{3.27}$$

It is assumed without the loss of generality, that $a_2 = R^b_{max} - r_{max}$ and $a_1 = R^b_{max} - r_{min}$. That means $Dist(\sigma_1)$ is given by $a_2 - a_1$. If the distance of that schedule is reduced be removing one contact windows from R_2 to any other request, one obtains a new schedule σ_2, and the penalty is altered from $a_2 \rightarrow a_2 + 1$ and $a_n \rightarrow a_n - 1$. That means in equation 3.27 exactly two terms of the sum are altered, namely λ^{a_2} changes to λ^{a_2+1} and λ^{a_n} to λ^{a_n-1}. As it was already shown in section 3.3.3, the inequality

$$\lambda^{a_2+1} + \lambda^{a_n-1} < \lambda^{a_2} + \lambda^{a_n} \tag{3.28}$$

is true when the redundancy is distributed fair. This means that also the penalty value for schedule σ_1 will decrease, as the following characteristic holds

$$\lambda^a + \lambda^b < \lambda^{a+s} + \lambda^{b-s} \tag{3.29}$$

Thus, $\gamma_2(\sigma_1)$ will be reduced by decreasing the distance by 1. Therefore, $\gamma_2(\sigma_2)$ is also smaller than $\gamma_2(\sigma_1)$, as $Dist(\sigma_2) < Dist(\sigma_1)$.

Case 2: For the second case it has to be shown now, that there exists at least one schedule σ_2, which has a smaller penalty value than schedule σ_1, if the total amount of assigned contact windows for schedule σ_2 is greater than the amount of assigned contact windows for schedule σ_1.
This is shown through proof by contradiction. If there would exist no $\gamma_2(\sigma_2)$, that is smaller than $\gamma_2(\sigma_1)$, the penalty value of schedule σ_1 is minimal and can be expressed as

$$\gamma_2(\sigma_1) = \lambda^{b_1} + \lambda^{b_2} + \lambda^{b_3} + ... + \lambda^{b_n} \tag{3.30}$$

The schedule is now modified to yield a new penalty value of

$$\gamma_2(\sigma_1') = \lambda^{b_1-1} + \lambda^{b_2} + \lambda^{b_3} + ... + \lambda^{b_n} \tag{3.31}$$

It can be seen easily, that $\gamma_2(\sigma_1') < \gamma_2(\sigma_1)$, because $\lambda^{b_1-1} < \lambda^{b_1}$. Furthermore, $\sum_i \kappa(R_i) < \sum_{i'} \kappa(R_i')$ holds, as the redundancy distribution was changed from b_1 to $b_1 - 1$. This means a schedule σ_1' exists, which fulfills the requirements in equation 3.25 and 3.25 and is therefore a contradiction.

In summary, the proofs of section 3.3.3 and 3.3.4 show that the proposed scheduling objective function γ_2 guarantees an equal distribution of redundant contact windows. While the γ_1 objective tries to maximize the number of contact windows included in the final schedule, the γ_2 objective prefers the schedules where a fair distribution of contact windows exists. This capability is a novel concept, which is not available in other state of the art scheduling systems. This characteristic better satisfies the demands of the small satellite community (compare section 3.2.1).

3.3.5 Search algorithms

The before described objective function is needed to compare schedule solutions of a problem instance in the search space. The implementation of the approach, the *CUSS* system, contains currently two search algorithms, a hill climbing algorithm and a branch and bound strategy using depth first search. The choice for this algorithms was made from a practical point of view, the search algorithms do not need to find an optimal solution for a given problem formulation. In the scope of academic ground station networks is optimality a less important criterion, it is

rather desired to fulfill the special scheduling requirement described in section 3.2.1, i.e. flexibility and short runtimes. Thus, simple search strategies were chosen, which delivered in the very similar SRS domain reasonable results.

Hill climbing

A very popular and often implemented search strategy is the hill climbing algorithm [Michalewicz and Fogel, 2004]. It is a simple, heuristic optimization method which tries to improve an initial guess by local search. This process runs until no further improvement can be achieved. In the case of the RRSS problem domain, the system starts with an initial schedule and assigns next the contact window, which increases the objective function γ most. This is repeated until no further increase of the objective function is achievable.
The hill climber search was chosen for mainly two reasons: First, it is a very simple search strategy and short runtimes can be expected due to limited search space traversal. The second reason is, that in similar scheduling domains very good results were obtained with that strategy (compare with EOS problem on page 45). Nevertheless, the hill climbing algorithm has a major drawback, it normally finds only local minima. An investigation of that phenomenon is discussed in section 3.4.3. The problem of local minima can be attenuated by integrating random steps in the schedule to reach different plateaus in the search space.

Branch and Bound / Depth First Search

The second algorithm utilized from the *CUSS* system is a combination of the Branch and Bound ($B\&B$) strategy and Depth First Search (DFS). The well known $B\&B$ strategy [Michalewicz and Fogel, 2004] is a general algorithm for optimization problems utilizing decision trees. The idea behind $B\&B$ is the division of the search space in several subproblems (branch) and to define a upper (respectively lower bound) of the quantity being optimized (bound), to identify which subproblems will not lead to a reasonable result. In this way is the optimization problem only limited to a subset of problems and only a fraction of the search space needs to be traversed, i.e. runtime is reduced.
Finding a solution to a given subproblem requires a search algorithm: For the proposed RRSS problem is the DFS algorithm a reasonable candidate. It is an uninformed search which creates a search tree starting from an initial root and adds

child nodes along each branch before backtracking. A more detailed description of DFS is given in [Cormen et al., 2009].
The branching step is realized using the priority scheme. A priority is defined for each contact window (see page 59, the priority is used from the $B\&B$ algorithm to decide which contact should be included next after branching). The upper bounds for the bound condition are described with the objective function γ, i.e. a subproblem is not further traversed if the resulting objective function value γ is to low. For more details about the realization of the search algorithms refer to the implementation section 3.3.6.

3.3.6 Implementation

A system overview of the *CUSS* system was already given in section 3.3.1, this section describes important details about the software implementation. Nevertheless, the implementation is only cursory described in here, but it might be helpful to understand how the different modules interact with each other to create a schedule for a given problem instance. Furthermore, knowing the implementation details illustrates the boundaries of the scheduler from a systematic point of view.

Core module

The core module was implemented in *Java* [3], it is therefore platform independent. It was implemented as a standalone system which creates for a given problem instance a schedule satisfying the demands defined in section 3.2.1. The input of the system is the problem instance in XML format, the corresponding schedule is also saved in XML and forms the output of the system. The problem formulation contains a list of satellites, ground stations, requests and contact windows. If the contact windows are not given inside the input XML file, the orbit predictor module is used to calculate all available contact windows for the specified requests. The output file contains the same information as the input file. Additionally, each contact window has an attribute assigned, which determines if it is contained in the final schedule. The search procedure uses a default parameter set to obtain a schedule (search algorithm, orbit predictor, etc.), the parameters can be changed with the GUI module. Second important component of the *CUSS* system is the objective function, which

[3] http://java.sun.com/docs/overviews/java/java-overview-1.html

is directly integrated in the core source code. Due to the definition of the objective function (see equation 3.4), the resulting value of the function grows with increasing problem sizes. Especially the size of the penalty values can increase dramatically, hence, the implementation of the objective function was slightly modified: The λ value of equation 3.8 is scaled down for large scenarios to prevent overflows during the calculation process. The characteristics of the objective function are not affected from this rescaling procedure.
The last integral part of the *CUSS* core module is the implementation of the search algorithms. As the hill climber search strategy is relatively simple, the implementation details are not further addressed. More interesting is the implementation of the branch and bound strategy. To benefit from the performance of specialized solver software, it was decided to translate the RRSS problem formulation for the CHOCO framework. CHOCO is a Java library for solving CSP problems and uses an event-based propagation mechanism with backtrackable structures [Menana and Demassey, 2009]. It is open source software and can be obtained from the CHOCO website [4]. The translation of the objective function γ in a CSP representation originated from this step.

Orbit predictor module

The orbit predictor module is required to calculate for a given set of satellites, ground stations and requests the contact windows. The available contact time is defined by the orbit geometry of the satellites, a orbit predictor can derive from the orbit elements and the locations of the ground stations the visibility. A broad spectrum of orbit predictor software is available, the *CUSS* system relies on on the open-source solution *predict*. The latest version is only available for Linux operating systems, it can be downloaded freely from the Internet [5]. Main reason to use *predict* for orbit calculations was the provision of a very simple UDP interface for orbit calculation requests. The *predict* application can run on any machine in server mode and the *CUSS* core module asks per Internet for any desired contact windows. To satisfy the demands of *CUSS*, the *predict* UDP interface was slightly modified to support multiple ground stations in the orbit prediction process.

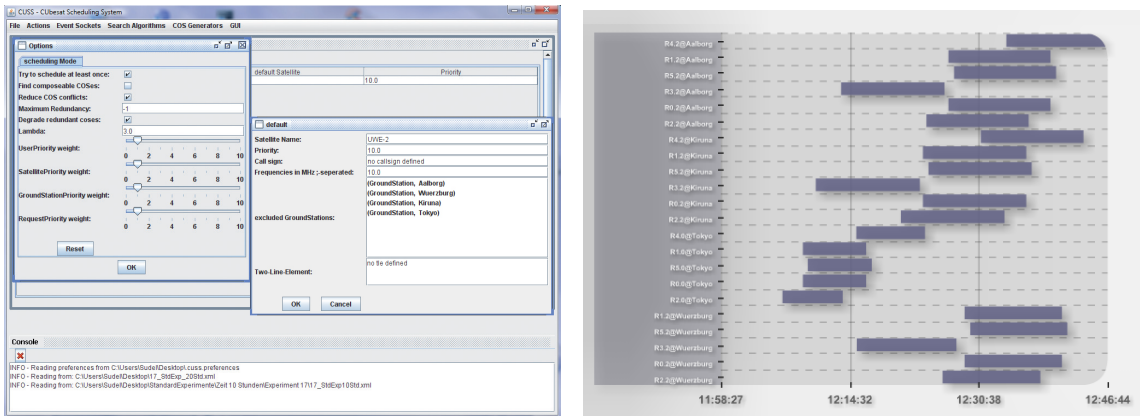
[4] http://www.emn.fr/z-info/choco-solver/
[5] http://www.qsl.net/kd2bd/predict.html

For the calculation of contact times are accurate orbit elements needed. Like many other orbit predictors relies *predict* on orbit elements in the form of TLE data sets [Hoots and Roehrich, 1998], i.e. uses the SGP4 model [Vallado, 2008] for orbit propagation. TLE data sets are only valid for 30 days, therefore it is recommended to create schedules with the *CUSS* system for time horizons less than 30 days. From the accuracy point of view are the deviations of the contact times after 30 days still acceptable, but the system was anyway designed to cope with flexible scheduling and short time horizons. The aimed time horizon is approximately one day, a more detailed discussion about implications on appropriate time horizons is presented in section 3.4.2.

GUI and visualization module

To support operators of academic ground stations two modules are provided: The first one, the so called visualization module, is only responsible for graphical interpretation of the created schedules, which are contained in the output XML files. The resulting bar diagrams (see figure 3.12(b)) helps the operator to have a quick overview on the contact window assignment in the complete network. Furthermore, the visualized schedules can be used to distribute the scheduling information on a website of a ground network.
The second module is used to control and parametrize the *CUSS* core module, the so called GUI module is the direct interface to the core. It can be used to initialize the schedule creation process, it provides functionalities to adapt the search parameters and can modify the problem instance by adding and removing satellites, ground stations and requests. The GUI module is as well implemented in Java, a picture is shown in figure 3.12(a). The communication to the *CUSS* core module was realized via event sockets based on TCP, so the software can be remotely controlled through the Internet.

The GUI interface implements so far no dedicated authentication service. Anyway should *CUSS* rather be seen as a scheduling system aiding academic ground station networks with providing a service for scheduling. Moreover, it would be reasonable to connect the *CUSS* system for authentication issues with already established infrastructure, e.g. the authentication server used in the GENSO network ([Page et al., 2010]). Further critical points related to operation of a scheduling system by a group of heterogeneous institutions are provided in section 3.5.



(a) GUI module (b) Visualization module

Figure 3.12: CUSS operator interface

3.4 Performance evaluation

To assess the applicability and limitations of the developed approach, the performance of the *CUSS* scheduling system was extensively evaluated. The first step is of course the definition of the evaluation criteria, which were derived from the scheduling requirements in section 3.2.1. The experiments focus mainly on the schedule creation time and the number of unsatisfied requests, a description of both criteria follows in the next subsection. A fair redundancy distribution of contact windows is already guaranteed from the objective function γ (compare section 3.3.2), therefore no dedicated experiments for its evaluation are necessary. Furthermore, the number of available redundant windows depends on the orbit geometry. Hence, it can be hardly expressed what degree of redundancy can be realized in a given problem instance. For more information on the redundancy distribution behavior, please refer to section 3.3.3.

A variety of different experiments were created to test the developed approach on typical small satellite mission scenarios. Here it is on the one hand important to cover different problem sizes and types and on the other hand to investigate the influence of different scheduling parameters (e.g. choice of scheduling algorithm etc.). The obtained results as well as a conclusion is presented at the end of this chapter.

3.4.1 Evaluation criteria

For evaluation of the CUSS scheduling system different performance criteria were defined, these are: The total schedule creation time (runtime) and the number of unsatisfied requests. It is important not to rely on a single criterion for evaluation to get a comprehensive overview of the performance. Moreover, the calculated schedule is often a tradeoff between different objectives. Depending on the application it might be rather important to have a short schedule creation time instead of an optimal schedule. In the following the utilized criteria are described in more detail:
Schedule creation time (runtime): Describes simply the time needed from CUSS to generate a schedule for a given scenario. This sounds like a trivial issue, but due to the NP-complete nature of the ground station scheduling problem is this measure very important [Barbulescu et al., 2004b]. Especially in the field of academic ground station networks, which are consisting of ground stations belonging to different institutions, flexibility is a strong demand, especially with respect to scheduling. Sudden failures of single ground stations might require a fast re-scheduling without long time delays. Therefore, the needed time to create a schedule is an important performance parameter.
Number of unsatisfied requests: This criteria describes the number of requests, which could not be included in the generated schedule. This could be the case if the scenario is very oversubscribed and it is not enough time available to accommodate a contact window for each request. Furthermore, situations are imaginable where low priority requests are preempted due to collision with high priority requests. In many publications related to the field of ground station scheduling is the objective to minimize the number of unsatisfied requests (e.g. [Barbulescu et al., 2007b]). Background is the wish of commercial ground networks to include as many users as possible in a given time horizon. In academic ground station networks is the number of unsatisfied requests an important measure, but not that critical as in classic networks as the participants of academic networks are more flexible and have no financial interest. Nevertheless, the criterion of unsatisfied requests can be used to compare the implemented system with other scheduling systems.
Other research works refer sometimes as performance criterion to an absolute percentage value of scheduled tasks. This was not done in this work for two reasons: First, an absolute value of scheduled tasks is in this case not very expressive, as it strongly depends on the level of oversubscription (see analysis in section 3.5). It is

more suitable to use the number of unsatisfied requests to have a relative performance measure of scheduled tasks (for example to compare two search algorithms). Second, a percentage measure of scheduled tasks is not always applicable to the RRSS domain, especially for the scenarios with a superior number of ground stations where redundant scheduling is desired.

3.4.2 Experiments

To get an comprehensive view on the performance and limitations of the system, it is necessary to evaluate the before mentioned criteria on various problems. To cover different aspects, like problem size and resource distribution, several experiments classes were designed, which reflect the conditions of individual mission or application scenarios. The experiments are divided in three problem classes (A, B and C).

Experiments class A

This class of experiments contains problems with a constant number of ground stations and increasing number of satellites, the problem size grows with increasing experiment. Experiment A.1 starts with 4 satellites and 4 ground stations, the last experiment evaluates the performance of the scheduler with 36 satellites on 4 ground stations (see also table 3.1). This problem instances correspond to scenarios where a distributed satellite system is launched and only a limited number of ground resources is available. For example, when a mission like QB50 is launched and the principal investigator owns only a small number of ground stations, which he can continuously use without restrictions.
The difficulty in this scenario is to include all satellites into the final schedule, while only a limited time horizon is available, i.e. these problems are oversubscribed.

Experiments class B

Very similar is the second experiment class B defined, but instead of increasing the number of satellites, the number of ground stations is increased (see table 3.2). The number of satellites is kept fix in class B at a value of 4. This corresponds to a situation nowadays observed in academic ground station networks like GENSO,

Experiment No.	# of Satellites	# of GS
A.1	4	4
A.2	6	4
A.3	8	4
A.4	10	4
A.5	12	4
A.6	14	4
A.7	16	4
.	.	.
.	.	.
.	.	.
A.17	36	4

Table 3.1: Class A experiments

Experiment No.	# of Satellites	# of GS
B.1	4	4
B.2	4	6
B.3	4	8
B.4	4	10
B.5	4	12
B.6	4	14
B.7	4	16
.	.	.
.	.	.
.	.	.
B.17	4	36

Table 3.2: Group B Experiments

where only a small number of satellites is operated, but many institutions are sharing their resources on a voluntary basis. Currently, only a small number of pico and nano-satellites are active, due to short lifetimes and irregular launch opportunities. In this scenario it is less difficult to accommodate all satellites within a schedule, it is rather desired to distribute the redundant contact windows equally between the space vehicles. Here, the novel concept of redundant scheduling comes into play.

Experiments class C

The last experiment class contains problems with increasing number of satellites and ground stations, i.e. those problems have a much larger search space than experiment class A and B. This represents a combination of the previous described application scenarios. Also, it can be expected that in future the number of small satellite missions will grow (compare section 2.1.2).The largest experiment of class C comprises 36 satellites and 36 ground stations (see table 3.3)
This experiment class was designed to test the performance of the *CUSS* scheduling system on large scenarios and to identify limits of the system.

Experiment No.	# of Satellites	# of GS
C.1	4	4
C.2	6	6
C.3	8	8
C.4	10	10
C.5	12	12
C.6	14	14
C.7	16	16
.	.	.
.	.	.
.	.	.
C.17	36	36

Table 3.3: Group C Experiments

Ground station locations and satellite orbit data

Important for the experiments of the three classes is also the choice of satellites and ground stations. Satellites in similar orbits in combination with geographically adjacent ground stations lead to more difficult scheduling problems due to overlapping footprints, than other combinations with more separated orbits and locations. Therefore, different data sets were created to test the performance on equal problem sizes in dependence of different orbits and ground station locations. In the following evaluation of the experiments, three different data sets are used to illustrate the influence on orbit geometry, these data sets differ only in the input set of satellites and ground stations. Many more data sets were used for an extensive evaluation of the *CUSS* system, the presented graphs represent only a subset of performed experiments, which were chosen to visualize certain aspects of the obtained results. The satellite orbit data was obtained from NORAD in the form of TLE files, ground stations with various locations were used, taking into account the distribution in current academic ground networks with concentrations in US, Europe and Asia. It would be possible to use any satellites from LEO orbits, but pico and nano-satellites are often launched in batches or clusters comprising several satellites, the corresponding orbit elements are almost equal and vary only slowly over time (a detailed

discussion can be found in [Ravandoor et al., 2010]). Thus, for the conducted experiments only the orbit data of pico and nano-satellites was included (even if not active anymore), as they better reflect real world problems, like clustered satellite structures in polar orbits. A list of the satellites used for the experiments can be found in appendix C.

Scheduling time horizon

Finally, the dependence of the scheduling time horizon needs to be evaluated, i.e. the time span a schedule is valid for. The time horizon was chosen to values between 5 and 30 hours, which is relatively short compared to other scheduling systems. But in the case of redundant scheduling it is especially important to evaluate the performance in such short time frames: First, the system was designed to be used in a very dynamic environment, so it should be able to react immediately if a new ground station joins or leaves the network. One could think of a situation where a ground station can not be used due to technical problems and a fast replanning is needed to have a valid schedule for the same day. Therefore, a short time horizon correlates directly with the flexibility of the system. Second, it is easier to accommodate all requests in the schedule when a large time horizon is used, as more contact windows will be available. Hence, the problems with short time horizons are more constraint and needs to be investigated. The last point for choosing short time horizons was already indicated in section 3.3.6: As the *CUSS* scheduling system uses TLE sets for orbit propagation, the accuracy of the contact window calculations decreases with time. This is especially true for the time directly after launch, where the TLE sets are updated typically every 12 hours, as the identified orbit elements are not very accurate at this stage. A new schedule creation process would be anyway initialized when new orbit sets are available. So, the schedule time horizon should be in the order of the minimum update rate of the TLE data sets.

3.4.3 Results

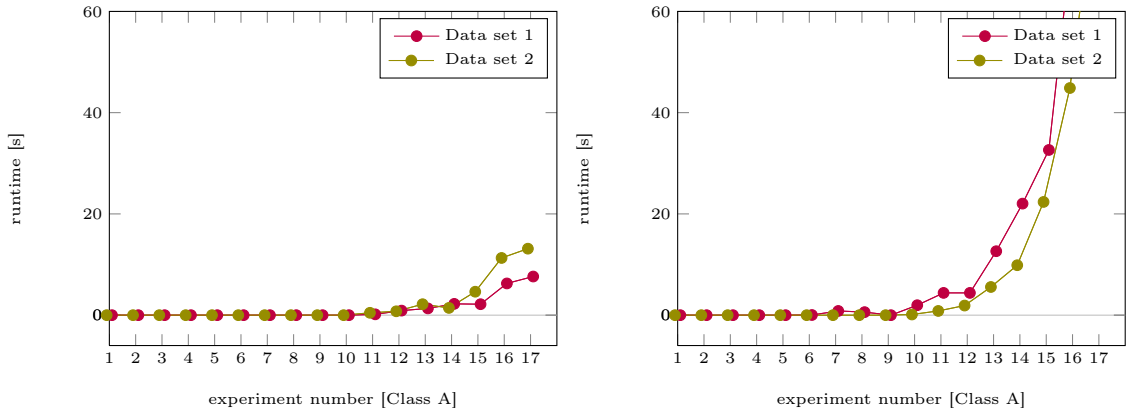
This section summarizes the most important results from the performed experiments. All the experiments were conducted on a desktop machine, equipped with an Intel Core 2 Duo processor and 2 GB RAM, running Windows XP. The *CUSS* system uses no multi threading, therefore only one core of the processor is utilized.

The priority of all entities (see equation 3.7) was set to the same value, thereby it is easier to interpret the obtained results. All requests refer to a desired contact time of at least 10 minutes, as it is assumed that a certain minimum contact length is required to transfer data during the contact window. Furthermore, it is allowed to assign as many redundant contact windows as possible (i.e. the maximum degree of redundancy is infinite, compare section 3.2.2).
All results were obtained by averaging each value over 30 trials. Both search algorithms of the *CUSS* system (*HC* and *B&B*) contain random steps in the search process. Using average values was especially important as the *HC* search algorithm is affected by local minima.

Runtime evaluation

The runtime of a scheduling system is an important performance parameter, but many factors influence the time needed to search for a valid schedule, which can not be directly controlled. Not only the hardware characteristics (like processor and RAM) play a major role, also the used programming language and the implementation of the algorithms itself influences to a large extend the result. Hence, the intention of this section is rather to elucidate the relation between runtime and problem size than to determine exact absolute values.
In all experiments conducted with the hill climbing algorithm was the runtime negligible. Even in the largest scenario (C.17) was the total time to create a schedule less than one second. The explanation is, that the number of available contact windows in the largest experiment is still manageable for the hill climbing strategy (about 3500 contact windows on average in a 10 hour time horizon, containing ca. 18000 conflicts). As the hill climbing strategy is not backtracking for better solutions, the search process terminates relatively fast. Thus, also in larger scenarios the runtime will be relatively small as the hill climbing algorithm is neither complete nor optimal.
The further evaluation of runtime properties is dedicated to the search process based on the *B&B* algorithm.

Class A: The measured runtime of the *B&B* algorithm is significantly higher: The CSP solver of the *CHOCO* library needs adequate time to find a good performing solution. In figure 3.13, the runtime of experiment class A is depicted for a time



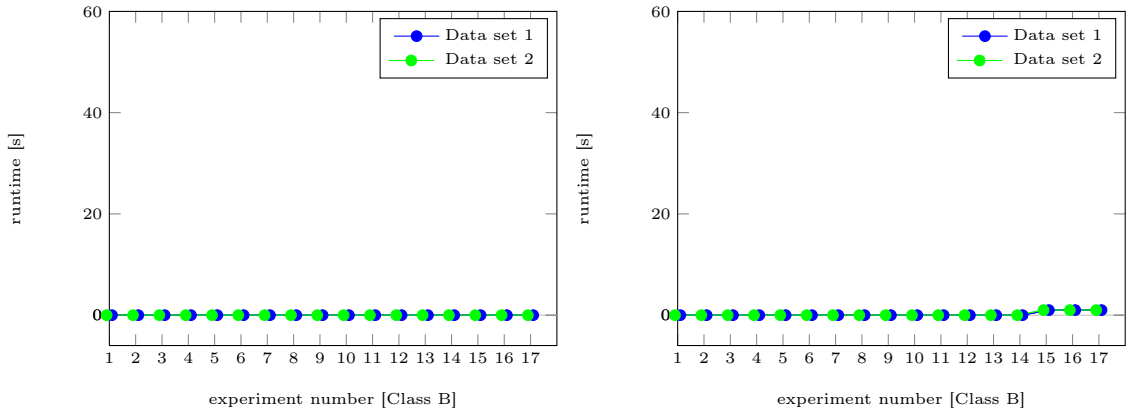
(a) Schedule creation time for 5 hour time horizon

(b) Schedule creation time for 10 hour time horizon

Figure 3.13: Runtime comparison for experiments of class A

horizon of 5 hours and 10 hours. One can see, that the resulting runtimes in both graphs stay relatively constant for experiment A.1 to A.12, but increase dramatically when more than 30 satellites are involved (experiment A.13 and higher). This is owed to two factors, on the one side, the number of calculated contact windows is steadily increased by the larger number of satellite. On the other side, it gets more and more difficult to accommodate all satellites in the schedule when only a small, constant number of ground stations can be utilized. The different data sets (input satellite orbits and ground station locations) have only a minor influence, as both data sets have almost equal amount of possible contact windows. The reason for the similar amount of contact windows is, that most of the satellites were launched into low Earth, sun synchronous polar orbits, which results in characteristic contact window patterns each day. Thus, also the search space of data set 1 and data set 2 have comparable size. The runtime shown in graph 3.13(b) is significant higher than the runtime in graph 3.13(a), both differ only with respect to the time horizon (5 versus 10 hours). Here, a strong influence of the time horizon on the runtime is observed, as doubling the time horizon doubles also the amount of contact windows.

Class B: In contrast to the experiments in class A, the task to find a good solution in class B is less difficult. The superior number of ground stations provide enough receiving points on Earth to accommodate all satellites in the final schedule. So,



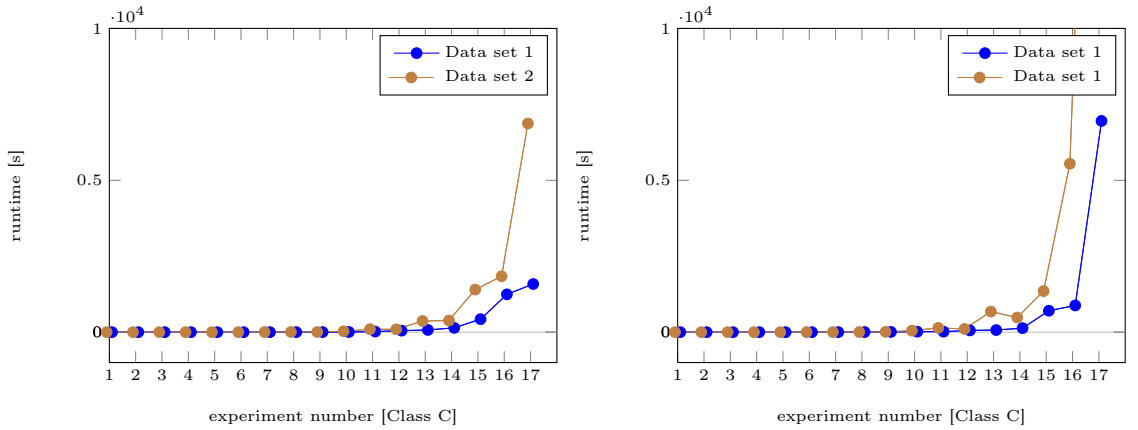
(a) Schedule creation time for 20 hour time horizon

(b) Schedule creation time for 30 hour time horizon

Figure 3.14: Runtime comparison for experiments of class B

a schedule satisfying all requests can be obtained relatively simple, the remaining task is then to distribute redundant contact windows fairly between the requests. It is surprising how strong this different condition affects the runtime of class B. In figure 3.14 is the runtime depicted for time horizons of 20 and 30 hours. The average result of the $B\&B$ algorithm for the largest scenario (B.17) is less than 2 seconds, which is comparable with the hill climber search performance. So, in this context is the runtime not a limiting factor, especially when the system is used for short time horizons of only a few hours.
Fortunately, this situation also reflects the environment in current academic ground station networks. This makes the presented approach a very promising candidate for recently initiated space missions and applications.

Class C: The results from this section are very interesting, as this class contains the largest problem instances (increasing number of satellites and ground stations). Indeed, enough ground station resources are available to accommodate all the requests, but the search space is significantly larger than experiment class A and B. This is also the explanation for the extremely high runtime values shown in figure 3.15. Compared to the previous results, they are more than an order of magnitude larger and constitute the constraints of the system, as it is not practical to solve large problems with the $B\&B$ algorithm. But it should be remarked, that the hill climbing strategy finds a solution also for this large scenarios in a few seconds. For



(a) Schedule creation time for 5 hour time horizon

(b) Schedule creation time for 10 hour time horizon

Figure 3.15: Runtime comparison for experiments of class C

the application in a real ground station network environment, it is therefore important to consider the problem size and find a tradoff between flexibility (i.e. runtime) and fairness (i.e. optimality).

Runtime for larger time horizons Finally, the runtime of the $B\&B$ algorithm for even larger time horizons was investigated, as it might be desired to create schedules for whole days to simplify the management of a ground station network. The results from experiment class C showed already, that the schedule creation time for the 10 hour time horizon exceeds 1 hour, the resulting runtime for even larger time horizons restricts the flexibility of the system to much for application in real world scenarios. The runtime in experiments of class B is also for large time horizons negligible when the $B\&B$ algorithm is used (see graphs 3.14(a) and 3.14(b)). Thus, the influence of larger time horizons on the runtime was performed for experiment A.17, the largest problem size from class A. In figure 3.16, the results for a time horizon up to 24 hours are shown. From this point of view, the $B\&B$ algorithm is still a reasonable candidate for experiments of class A, as the runtime does not exceed critical limits.

Unsatisfied request evaluation

Unsatisfied requests, i.e. requests which have no contact windows assigned in the final schedule, occur if not enough ground station resources are available. This is

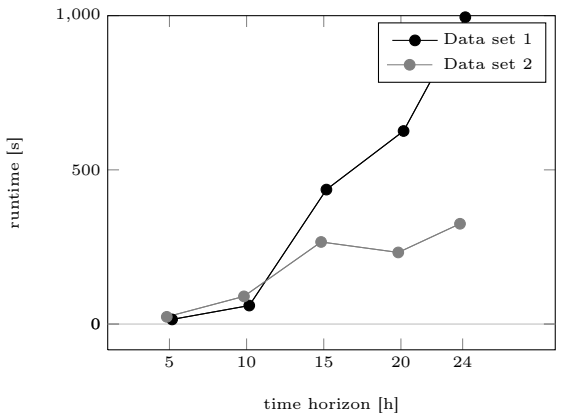
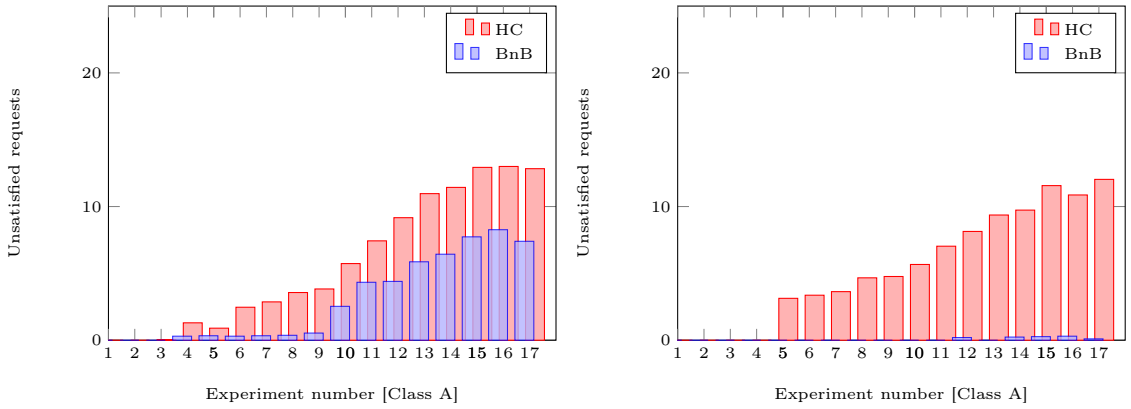


Figure 3.16: Runtime comparison for larger time horizons

also reflected in the following results, where mainly the experiments from class A are affected by unsatisfied requests. The evaluation of unsatisfied requests is reasonable to compare the performance of different search algorithms with each other. Furthermore, it is used as the main optimization criteria in similar scheduling applications from the SRS domain (compare section 3.1), the following evaluation gives an insight of the relevance for the RRSS problem domain.

Class A: In figures 3.17 and 3.18, the results from experiment class A are depicted for time horizons of 5 and 10 hours. The $B\&B$ algorithms performs better than the hill climbing strategy on all problem instances. It was expected, that the number of unsatisfied requests increases with growing number of satellites, when the number of ground stations is fixed. For example in problem A.1 only 4 satellites need to be incorporated in the final schedule, which can be solved from both search algorithm without observing any unsatisfied requests. In contrary, in problem A.10 it is not possible to integrate all requests in the final schedule (in data set 1 the HC algorithm suffers 6 unsatisfied requests on average, the $B\&B$ algorithm only 3, compare figure 3.17(a)). A simple way to reduce the number of unsatisfied requests is to enlarge the time horizon, which is depicted on figures 3.17(b) and 3.18(b). In this way more resources are available and the $B\&B$ strategy accommodates unsatisfied requests in the resulting gaps. Unfortunately, this is not the case for the HC algorithm, the obtained schedules are also for larger time horizons frequently affected from unsatisfied requests.

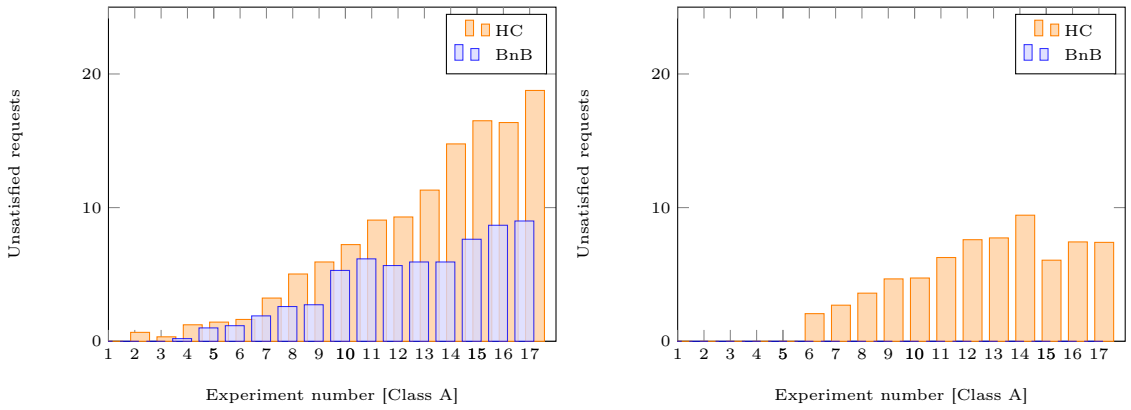
Very interesting is the performance of the hill climber algorithm in certain problem



(a) Unsatisfied requests for 5 hour time horizon (Data set 1)

(b) Unsatisfied requests for 10 hour time horizon (Data set 1)

Figure 3.17: Unsatisfied request class A



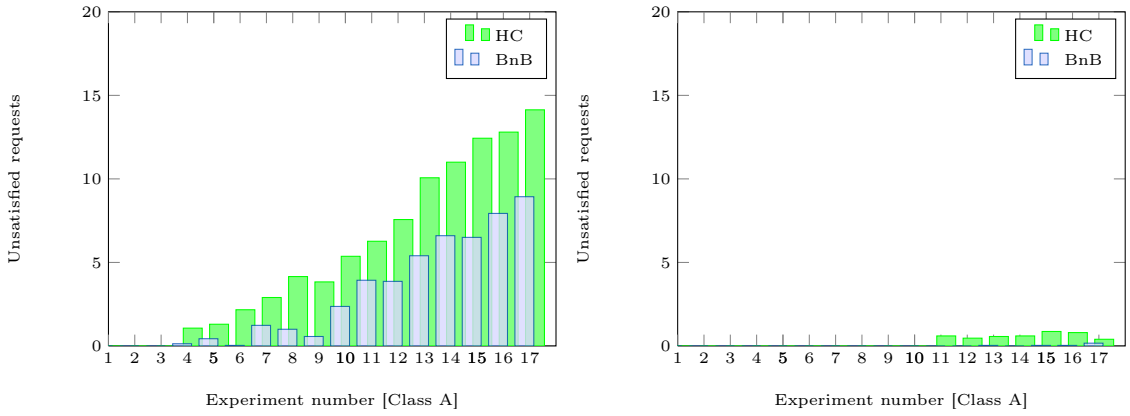
(a) Unsatisfied Requests for 5 hour time horizon (Data set 2)

(b) Unsatisfied Requests for 10 hour time horizon (Data set 2)

Figure 3.18: Unsatisfied request class A

sets. A major drawback of the hill climber search is, that it resides in local minima. Such a situation is depicted in figure 3.18(b), where the problem A.14 is subject to a relatively high number of unsatisfied requests. Surprisingly, experiment A.15 is affected by less unsatisfied requests, although it contains more satellites and the same amount of ground stations. This performance of the hill climber search yields to counterintuitive situations, the vulnerability of the hill climber becomes obvious,

as enforcing the problem with more satellites can lead to better results. Nevertheless, in some problem instances also the *HC* algorithm obtains very good schedules with a negligible number of unsatisfied requests (see figure 3.19(b))



(a) Unsatisfied Requests for 5 hour time horizon (Data set 3)

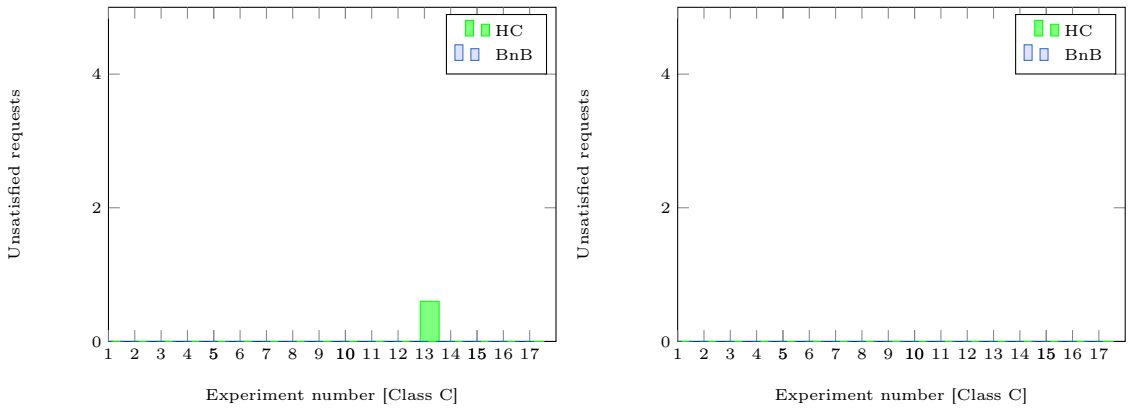
(b) Unsatisfied Requests for 10 hour time horizon (Data set 3)

Figure 3.19: Unsatisfied requests class A

Class B: All experiments of this class showed no occurrence of unsatisfied requests, as the increasing number of ground stations provide enough resources to accommodate all satellites. This is again a very comfortable situation, as both search algorithms are applicable without restrictions to the scheduling problem faced nowadays.

Class C: The last presented bar diagrams are from experiment class C, i.e. with increasing number of satellites and increasing number of ground stations. As shown in figure 3.20, unsatisfied requests are only observed sporadic when the hill climber search is used. The explanation is again the problem of local minima. The branch and bound strategy was able to accommodate in all experiments at least one contact window from each satellite.

In conclusion it should be mentioned, that the number of unsatisfied requests is a good indicator for the performance of the implemented algorithms. It can be used to identify the bottlenecks of a given problem instance. Nevertheless, it is problematic for a performance comparison with other scheduling systems, as the resulting value



(a) Unsatisfied Requests for 5 hour time horizon (b) Unsatisfied Requests for 10 hour time horizon

Figure 3.20: Unsatisfied requests class C

strongly depends on the given mission scenario. In other scheduling systems is the objective mainly to optimize the number of unsatisfied requests. The problem cases of the RRSS domain are typically flexible enough to avoid unsatisfied requests by increasing the time horizon. A more detailed discussion follows in the conclusion.

3.5 Conclusion and discussion

This chapter introduced a novel scheduling approach for academic ground station networks. It was tailored for the special demands of small satellite projects and better satisfies their special demands in flexibility and redundancy, than other state of the art satellite scheduling algorithms. The proposed algorithm incorporates the assignment of redundant contact windows, which is necessary to better utilize the resources of academic ground networks. So, additional value from the distributed architecture of academic ground station networks can be obtained.

The implemented *CUSS* system is the first scheduler adopting the concept of *redundant request satellite scheduling* (RRSS). It provides all functionalities to create schedules for highly distributed ground station networks and can be remotely operated via Internet. The *CUSS* system itself is a standalone system to create schedules for small satellite missions, but its application is intended in a broader context of a comprehensive ground station management system.

The performance of the system in the evaluated scenarios (based on real satellite

orbit data) is very promising: With respect to current academic ground station network implementations (like GENSO or GSN) is the proposed approach and the *CUSS* system applicable without restriction. The results from the experiments were evaluated regarding two main aspects: First, the time to create a new schedule (runtime) was measured, which is necessary when changes in the network topology occur or new orbit data was issued. The hill climbing search algorithm is able to create a new schedule within less than 5 seconds for all the conducted experiments. This is especially useful if sudden events enforce the creation of a new schedule, for example when a ground station fails due to technical problems. The obtained runtime values for the hill climber strategy fully comply with the requirement of a flexible scheduling system. The branch and bound strategy is also able to calculate new schedules within short time spans, but the applicability is limited to scenarios of moderate problem size. For very large scenarios with more than 30 satellites and 30 ground stations, the measured runtime is particularly larger than 1 hour. This might be still suitable when a new schedule is regularly created in a 24 hour interval, but might not by appropriate to react flexibly on sudden changes. Nevertheless, the real world problem sizes imposed nowadays on academic ground station networks are still small enough to rely on the branch and bound algorithm. The second investigated evaluation aspect was the number of unsatisfied requests, which describes how many requests could not be integrated in the final schedule. Unsatisfied requests are only observed when a problem is heavily oversubscribed, i.e. when many satellites need to be accommodated on a small number of ground stations within a short time horizon. The branch and bound strategy performs here better than the hill climber search, due to the more extensive search strategy. Additionally, the hill climber search stucks sometimes in local minima, which leads sometimes to unexpected unsatisfied requests in some problem instances. Thus, it is recommended to use the branch and bound strategy for schedule creation if no runtime limitations apply. Furthermore it should be mentioned, that the operator of the scheduling system can choose the time horizon arbitrarily, so it is in general possible to avoid a high number of unsatisfied requests by enlarging the time horizon. But one has to take into account, that extending the time horizon will also increase the runtime of the schedule creation process.

The presented approach opens new opportunities for small satellite missions as redundant windows are incorporated in the final schedule, which can be used e.g. for

data management (see chapter 4). For larger distributed space missions, containing more than 100 space assets, a tradeoff between runtime and performance (unsatisfied requests) has to be found. More sophisticated search strategies (like branch and bound) obtain better schedules, but have a significant higher runtime. However, in the context of academic ground station networks is the optimality of the generated schedules not required, so one might rather prefer short runtimes.

An extension of the *CUSS* system with additional search algorithms is planned, promising seems the HBSS algorithm from Bresina [Bresina, 1996]. In other SRS application fields have genetic algorithms proved to obtain better solutions than other standard search methods. An interesting comparison would be, if these algorithms perform as well on the RRSS problem, as it differs in the scheduling objectives and problem representation.
A very important issue, which was not handled within the scope of this work, is organization and administration of scheduling in academic ground station networks. The proposed system focuses only on the scheduling approach itself and does not consider any administrative issues. Nevertheless, the coordination of a scheduling system is from an administrative view a very critical point, and should be handled from one or more operators for the complete ground network. These operators have to initiate scheduling and rescheduling processes, define priorities for satellites and ground stations, add new participants etc. Basically there are two choices how the administration could be organized: A centralized administration from a single organization or a distributed solution where several participants coordinate and manage the scheduling system. In a decentralized organization arises the question, who has the permission to modify or parametrize the scheduling process. Especially when the participants share their resources on a voluntary basis, a centralized coordination instance might be difficult to realize. First attempts with a centralized administration was implemented in the GENSO project.
Another point for discussion is the integration and automation of scheduling systems in ground network infrastructure. An automated execution of the generated schedules is feasible from a technical point of view, but from a legal point of view can be problematic: For example requires the Amateurfunkgesetz (AFuG [6]), which is valid in Germany for ground stations using the amateur radio bands (UHF/VHF),

[6] http://www.gesetze-im-internet.de/afug_1997/index.html

that an registered station needs to be operated or supervised from a licensed radio amateur. Such administrative issues need to be taken into account for the migration to fully autonomous ground station networks, compliance with international regulations should be further elaborated.

Chapter 4

Data management for information recovery in ground networks

In classic ground station networks, mission operation center are responsible for collecting science and operations data. In academic ground station networks, the highly distributed topology makes it more sophisticated to bring the data from the source to the destination (satellite data has to find its route back to the operator), which is realized in different ways (solutions are for example provided by the GENSO project [Shirville and Klofas, 2007] [Page et al., 2008]).
The term *data management* is used in a broad context and comprises all disciplines related to optimal usage of data resources. In this chapter, a special aspect from the field of data management is handled: Due to the restricted size and mass of small satellites, this communication systems are rather simple, using lower frequency bands in UHF/VHF or S-Band, than state of the art technologies available for larger space vehicles (e.g. Ka-Band and X-Band). UHF/VHF systems utilize often dipole antennas, which typically have a limited directivity, therefore the footprint of those satellites is very large (see figure 4.1 with the footprint of UWE-2). Thus, for a satellite over Europe a large number of ground stations are able to receive its signals in parallel at the same time. Quite often, several ground stations track in parallel a satellite and forward the received data to the satellite operator (like performed from Oda et al. [Oda et al., 2008]). In classic ground station networks, parallel tracking is not performed, on the one hand because the satellite beam is relatively narrow and on the other hand, reserving several ground stations from a space agency can be very expensive. Academic ground station networks share their resources without

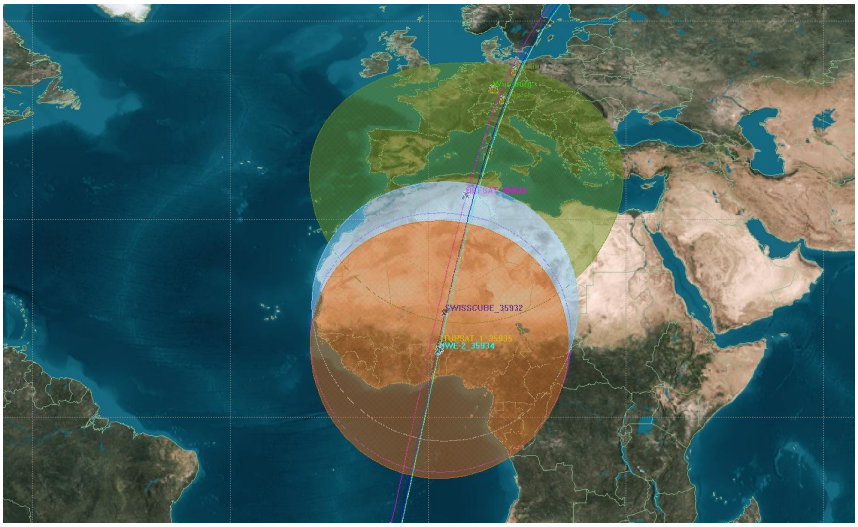


Figure 4.1: Footprints of UWE-2, BEESAT, ITUpSAT-1 and Swisscube, launched together on 23. September 2009

commercial interest and promote in this way the parallel reception of a data stream from a singe satellite. Receiving downlinks from a satellite in parallel contains opportunities as well as challenges: The opportunity is to take advantage of redundant ground station links, e.g. for robustness. The challenge is the realization of such a system, using proper data management and synchronization methods, which is the content of this chapter.

The data management system introduced in this chapter evolved from the idea to combine several data streams from the same satellite, received at geographically distributed ground stations. Theoretically, these data streams received in parallel on terrestrial stations should be identical, in reality received data differs mainly due to two factors:

- The contact time between the satellite and individual ground stations differs for overlapping footprints. When two overlapping, but geographically distant ground stations are taken into account, there is still a small period of time where only one of these will be in contact with the satellite. This results in different sets of data frames received at the individual stations.

- The received data can be corrupted, resulting in bit errors or even missing packets. Transmission errors are caused by atmospheric disturbances, low

signal to noise ratio or inaccuracies at the receiver side. So, the content of the data frames can differ, even if corresponding to the same data packet.

This leads to the situation that the received data streams (or data packets) are in a large fraction identical for overlapping ground stations, only a small portion differs in corrupted or lost information.

Starting from the previously described circumstances, the idea was to develop a system which automatically combines the different data streams received in a ground station network to one single data stream using proper data management. A satellite operator would then monitor only a "virtual" single data stream, composed from the information of the streams received in the network. Combining several data streams from the same satellite received at geographically distributed, overlapping ground stations opposes the following challenges:

- Bringing data frames in the correct order on a global timescale. Due to unsynchronized clocks on the ground stations and transmission delays in space and on Earth, the temporal ordering of the packets can be altered.
- Identifying identical data packets, if redundant packets were received from the ground network.
- Reconstruct data gaps with redundant information.

These challenges comprise further problems, which need to be addressed: Ordering data frames on a global timescale implies proper synchronization procedures. Reconstructing data gaps relies on redundancy identification. Such issues are handled in this chapter under the context of data management, the focus lies on the problem of time and data synchronization as well as data combination in academic ground station networks.

4.1 Problem definition and state of the art

The motivation of the proposed data management system was described before in a general way, a more formalized representation of the problem statement will be developed in this section. Additionally, suitable state of the art approaches to handle the problem of data management in academic ground station networks are presented and their applicability and restrictions are analyzed.

4.1.1 Problem description

The formalization of the problem statement is presented in this section. The problem itself can be divided in two separated subproblems, which are handled separately: First, participating ground stations of the network have to be synchronized with each other. Synchronicity between participating ground stations is required to order the received data frames on a global timescale. Synchronization is here related to synchronizing computer clocks, as well as synchronizing data streams. Second, the information from the synchronized data streams (respectively frames) has to be combined to reconstruct data gaps. In the following is the term data stream considered as a sequence of data packets transmitted from the satellite to the ground station. Furthermore,the terms data packet and data frame are used interchangeable (but the correct term for data on the link layer is *data frame*, see also appendix B). The underlying scenario contains a single satellite, which transmits data frame F_1 to frame F_l to Earth. A network of n ground stations is able to communicate with that satellite, the stations are consecutively numbered with capital letters $A, B, C...$

Synchronization problem

It is assumed that a certain fraction of the n ground stations in the network is able to receive the data frames simultaneously. Nevertheless, these simultaneoulsy (or duplicate) received frames can be corrupted by transmission errors, therefore the data content of the duplicate frames might differ. The data content of a received data frame is denoted by F_i^{gs}, which describes the frame number i received at ground station gs. The ground stations are numbered in alphabetical order, i.e. $gs = A, B, C, ...$. For example describes F_2^A the data content of the second frame received from ground station A. Each ground station is furthermore able to record the time of reception for each frame F_i^{gs} (denoted as timestamp $T(F_i^{gs})$). The recorded timestamp refers to the local computer clock available in each ground station.
The data synchronization problem is, to relate the obtained time of receptions $T(F_i^{gs})$ to an estimated time of transmission $\tilde{T}_(F_i^{gs}) = f_{sync}(T(F_i^{gs}))$, which can be used to identify duplicate frames. Duplicate frames were transmitted at the same time from the satellite, but were received at different times due to orbit position and system delays in the ground stations. The data content itself is not enough information to identify duplicates, due to transmission errors. The ordering of the

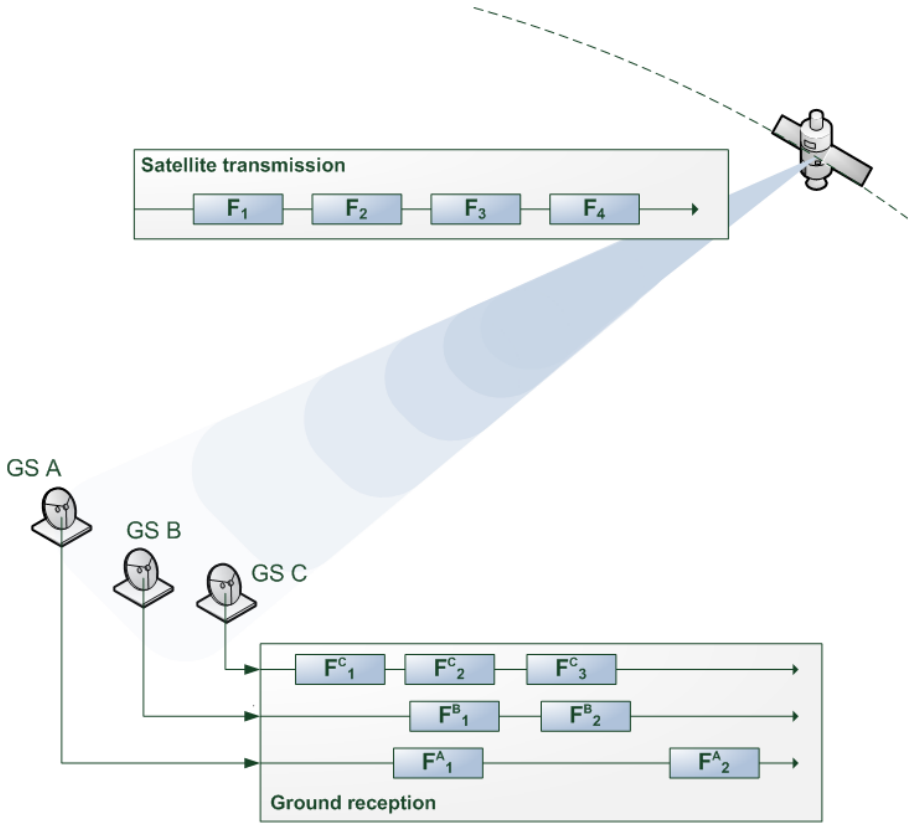


Figure 4.2: Data management problem in ground station networks

received frames might also not be consistent, for example could the second frame of ground station C (i.e. F_2^C) be the same as the first packet of ground station B (i.e. F_1^B), if their was a frame lost or not received before (compare figure 4.2). Therefore, it is necessary to derive from the time of reception $T(F_i^{gs})$ the estimated time of transmission $\tilde{T}(F_i^{gs})$, to determine how the received data frames correspond to the transmitted data frames. In summary, the objective is to derive an assignment f_{sync}:

$$f_{sync} : T(F_i^{gs}) \rightarrow \tilde{T}(F_i^{gs}) \tag{4.1}$$

To estimate the time of transmission, several delays on the receiving chain need to be compensated, a more detailed analysis is provided in section 4.2. It is in general not possible to determine the time of transmission exactly.
Additionally, as each ground station in the network has its own computer clock, an important prerequisites is to synchronize these clocks with each other, to compare the timestamps $T(F_i^{gs})$ on a global timescale. In the following, synchronizing the local computer clocks is referred as *time synchronization*, estimating the time of transmission to identify duplicate frames is referred as *data synchronization*.
With a suitable strategy to estimate the time of transmission, it is possible to identify duplicate frames and combine them in data sets S_k:

$$S_k = \left\{ F_i^{gs} \,\middle|\, f_{sync}\left(T(F_i^{gs})\right) \in [T_k^{min}, T_k^{max}] \right\} \tag{4.2}$$

The time of transmission $f_{sync}\left(T_i^{gs}\right) = \tilde{T}(F_i^{gs})$ can only be estimated with a certain accuracy. Therefore, an upper and lower bound (T_k^{min} and T_k^{max}) for a data set k has to be found, which determines identical frames respectively duplicates (see also figure 4.3). So, the main problem for grouping the received data frames in sets using equation 4.2 is the appropriate choice of T_k^{min} and T_k^{max}. If the time interval $[T_k^{min}, T_k^{max}]$ is to small, two identical frames might be considered as not identical, if the time interval $[T_k^{min}, T_k^{max}]$ is to large, different frames might be considered as identical. Therefore, the careful choice of $[T_k^{min}, T_k^{max}]$ is critical. As a reminder: The data content of the frames in a data set S_k are not necessarily identical, as the data can be corrupted by transmission errors. This circumstance is then used for data combination to recover from transmission errors.

To solve the data synchronization problem, an assignment f_{sync} has to be developed to construct the data sets S_k. These data sets are the input for the second subproblem, the data combination.

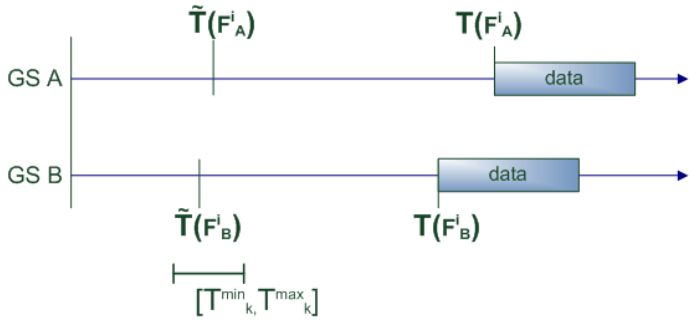


Figure 4.3: Synchronization of parallel received frames

Data combination problem

The second subproblem is the combination of the identified data sets S_k to single data frames. The idea is to compensate transmission errors by comparing the data content of duplicate frames. From the operators view the combined data frames can be treated like a single, virtual data stream received from the satellite, created from n data streams received from the n ground stations. So, the objective is to derive a function f_{comb}, with

$$f_{comb}(S_k) \rightarrow F'_k \tag{4.3}$$

The obtained data frames F'_k should in the ideal case match the frames transmitted from the satellite, or expressed mathematically:

$$F'_k = F_k, \forall k = 1 - l \tag{4.4}$$

where F_k , $k = 1$ to l describe the frames sent from the satellite. The challenge of combining the received data is the detection and correction of transmission errors. The key aspect for the data combination procedure is the high degree of redundancy introduced through the simultaneous reception of data frames. This redundant information can be used to obtain the original sent data frames. Beside the redundancy inherent due to parallel reception of data frames, another source of redundancy can be used: The frame structure is mainly defined by the communication protocol on the satellite link and provides additional information to reconstruct corrupted data frames.
Before addressing the developed algorithms, other state of the art approaches for data recovery and synchronization are discussed in sections 4.1.2, 4.1.3 and 4.1.4.

4.1.2 Data and network management in computer networks

As already discussed in the beginning of this chapter, data management is a broad research field and contains many more disciplines than mentioned in this section. The focus of this work are approaches respectively mechanisms for data synchronization and combination in distributed ground station systems to achieve error robustness. Nevertheless the general terms of data and network management are introduced here to provide a better overview of special problems and needs in space applications.

Under the context of data management are several disciplines grouped together, especially relevant for space applications is data quality management, database and content management as well as data security management. Montenegro discusses in [Montenegro, 2008] data management with special focus on orbit systems and describes their special requirements, like reliability, error robustness, etc. Tendencies to more autonomy, distributed systems and more sophisticated software can be observed in the last years. In ground systems typical data management functions comprise among others data compression, data quality monitoring and payload data handling [Whitworth, 2003]. Data management is also an important aspect with respect to efficient resource utilization. Especially in the context of small satellite missions is communication time a valuable resource, restricted to a few minutes each day. Therefore proper data management to increase throughput is a crucial task [Schilling, 2009c].

The term network management comprises operation, administration and maintenance of computer and telecommunication networks. The International Organization of Standardization (ISO) developed the FCAPS model as a framework for network management, which comprises Fault Management, Configuration Management, Accounting Management, Performance Management and Security Management (ISO/IEC 10040). A good overview on network management is given in [Leinwand and Fang, 1993], which contains a very general description of disciplines related to network management. A very popular implementation is the Simple Network Management Protocol SNMP, which provides services to operate and administrate computer networks remotely [Stallings, 1993]. New approaches extend the FCAPS concept also for Ad-hoc networks, which is especially interesting for ground station networks to cope with the high dynamics of LEO satellites. Such an Ad-hoc Network Management Protocol (ANMP) was proposed by [Chen et al., 1999].

Current implementations of academic ground station networks rely only rarely on these standards, they incorporate often individual solutions for achieving proper data management. The simple strategy followed by the GENSO network is the distribution of satellite data by a hybrid peer-to-peer network, which is supervised from an authentication server [Shirville and Klofas, 2007]. The data exchange takes only place between the client peer (Mission Control Center) and the server peer (GSS). The japanese GSN network uses a similar server client structure to exchange data in the network. The implemented software bases on Web Services technology for remote operations [Nakamura and Nakasuka, 2006] [UNISEC-GSN, 2006].
The next sections describe state-of-the-art approaches for data combination and synchronization, i.e. algorithms related to the problem description in section 4.1.1.

4.1.3 Data management for information recovery

Majority voting

Majority voting is a well known redundancy concept in space systems to ensure error robustness [Fortescue, 2003]. The algorithm itself is rather a simple decision rule for hardware devices, for example processors or microcontrollers. A good example is the main processor (CPU) of a spacecraft, which is often available in 3 identical versions. If for any reason, e.g. radiation damage, one of these processors is not working properly anymore, the correct result of a processor operation can be checked by comparing the calculated values from all 3 CPUs for that operation. The correct result of the operation is then considered as the result calculated by the majority of the CPUs (i.e. 2 out of 3 CPUs).

This procedure can be easily transferred to other systems, for example satellite

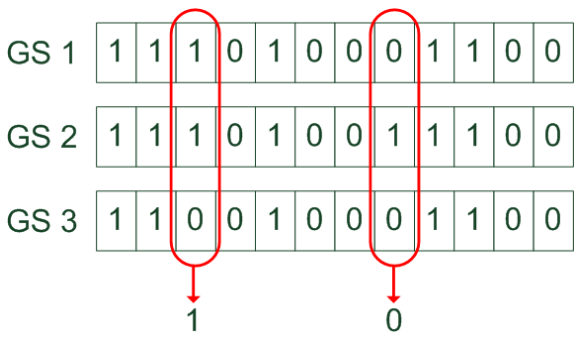


Figure 4.4: Majority voting for parallel received data frames

communication: When a satellite passes over a ground station, also the neighboring ground stations are able to receive the same signals from the satellite. A majority voting algorithm for such a ground station network can be implemented by just comparing the received data streams from the different ground stations bit by bit. An example is illustrated in figure 4.4, where the data streams of 3 ground stations are compared with each other to obtain an uncorrupted data frame. A majority voting algorithm is a fairly simple approach to achieve error robustness, a first concept in that direction was proposed from Stolarski [Stolarski and Winiecki, 2006] and Dabrowska [Dabrowska and Stolarski, 2007]. They proposed to compare the data streams from radio links bit by bit to reduce the Bit Error Rate (BER). Their approach focused on the comparison of the physical layer of radio links. In contrast, this work uses data link layer information to recover information gaps. Stolarski conducted hardware experiments on a stratospheric balloon [Stolarski, 2009] to demonstrate the occurrence of local transmission errors at different ground stations. The obtained error ratios show the applicability of majority voting to satellite communication channels.

Another system, which uses a different principle of majority voting, was proposed from Lai [Lai et al., 2009]. In contrast to the work discussed before, Lai uses sequential transmissions for bit comparison. The proposed approach is only applicable to communication links using the Automatic Repeat reQuest (ARQ) mechanism, which requests automatically a corrupted data frame again from the sender, until the data frame was received correctly or a certain number of trials were not successful. The basic idea of Lai is to keep the corrupted data frames stored in memory, until a specified threshold of corrupted packets was received. The system tries then to reconstruct the correct packet by majority voting bit by bit. As traditional error correction is divided in Forward Error Correction and ARQ schemes, the method of combining both approaches is also called Hybrid ARQ schemes.

The performance of the proposed system was evaluated with simulations, the algorithm exceeds the classical ARQ scheme in error correcting performance as well as throughput efficiency [Lai and Lu, 2010]. Nevertheless, additional memory is needed to keep corrupted data frames in memory, data processing time is negligible.

Forward Error Correction for satellite communication links

A field which is closely related to the data combination problem presented in section 4.1.1 is the discipline of error correction. The principle of error correction is to reduce the amount of transmission errors through proper usage of artificially added redundancy. This kind of error correction is called Forward Error Correction (FEC). FEC is a very common strategy in satellite communications to guarantee error robustness on the communication channel. The principle relies on adding redundancy to enable the correction of transmission errors on the receiver side. This is reasonable for wireless communication in general, i.e. also in satellite communication. An overview of different error correction algorithms can be found in [Klimant and Piotraschke, 2003], they can be distinguished in two main groups, block codes and convolutional codes. A popular block code family is Reed Solomon, which was used for example in the Voyager and Magellan satellite missions. Prominent mission examples utilizing convolutional codes are the SMART-1 and ROSETTA missions. Also the CCSDS recommends error correction codes on basis of of these techniques [CCSDS, 2003a][CCSDS, 2003b]. For a more extensive overview of error correction in satellite communications please refer to [Richharia, 1999].
Also in academic ground station networks are forward error correction capabilities desired, but are not supported from the primarily used communications protocol AX.25. Of course, it is in general possible to use other protocols, which support error correction, but the drawback is that the interoperability with a standard ground station equipment is then not given any more. Such a solution is used from in the AAUSAT-II project [Alminde et al., 2003], the integrated communication device MX909 uses a block code for telemetry encoding. Furthermore, interleaving is used to protect the transmission from burst errors. The AX.25 structure is then not longer identifiable with a standard TNC device. To overcome the problem of interoperability, other approaches are proposed: McGuire et al defined an extension for AX.25, which integrates an FEC mechanism without altering the AX.25 frame structure. The so called FX.25 protocol [McGuire et al., 2006] adds a preamble and postamble to each AX.25 frame, which is then still identifiable for a standard AX.25 device, but can be corrected from a receiver which implements the FX.25 protocol. First tests with the FX.25 protocol were implemented and performed with the AO-40 satellite [Karn, 2002]. A similar approach was presented in the scope of the UWE-1 project at the University of Würzburg. The observed PER in the AX.25

communication link was main driver for the development of a FEC extension for AX.25 [Zeiger et al., 2006]. The strategy is to neglect the FCS field of the TNC and adding redundancy in the AX.25 drivers at the physical and link layer (e.g. 6pack or kiss protocol) to stay consistent with the AX.25 specification.

4.1.4 Time synchronization

This last state-of-the-art section is dedicated to time synchronization. The presented approaches are totally independent from the data combination problem discussed before. Nevertheless, proper synchronization is a necessary prerequisite for a solution to the data synchronization problem (see section 4.2.2). Therefore, the most important contributions of computer clock synchronization are introduced:
Time synchronization in networks is a widely known problem and occurs in everyday life. Especially in the context of the Internet computer synchronization were such issues studied in detail. Current network clock synchronization methods are distinguished in linear systems [Lindsay and Kantak, 1980] and byzantine agreement [Lamport and Melliar-Smith, 1985] algorithms.
The most famous clock synchronization implementation is the Network Time Protocol (NTP), developed and maintained from David Mills. The NTP protocol exchanges timestamps between time servers and client to estimate the offset between them. A clock filter algorithm is used to identify clients with large delays producing large errors. The intersection and clustering algorithm [Mills, 1996] is then employed to identify appropriate reference clocks and to calculate time corrections to discipline the local clock. The achievable accuracy of the NTP protocol depends on many parameters (e.g. network delay, hardware, number of available time servers, etc.), but can be assumed to be a few milliseconds. For more information about the working principle of NTP please refer to [Mills, 2006]. Meanwhile four generations of the NTP protocol were issued: Version 1 was documented already in 1988 [Mills, 1988], only a few years later NTP version 2 [Mills, 1998] and version 3 [Mills, 1992] were published. The new versions introduced a broadcast mode as well as new features and algorithm revisions [Mills, 2003]. The latest generation, NTP version 4, contains a simplified protocol standard, the Simple Network Time Protocol (SNTP) [Mills et al., 2005]. Main difference of SNTP is, that it uses only one time server for synchronization, while the NTP standard uses several time servers to determine an accurate time (byzantine agreement). SNTP is compatible with

NTP and is available for almost all operating systems. A detailed description of the different milestones of the NTP development, from the first RFC documents until the latest NTP version, can be found in [Mills, 2003].
Several other protocols are also available for distributing time in a network, for example the Daytime protocol [Postel, 1983], or the Time [Postel, 1983] protocol, but these protocols are only used rarely, due to the high performance and wide acceptance of the NTP protocol. Further protocols for time recording and transmitting are listed in [Mills, 1991].
In recent years another technique became very popular for time synchronization, using GPS (respectively GNSS) satellites for clock synchronization. The basic principle of time and frequency dissemination uses a telecommunication system to distribute time from a precise clock, a good general overview on radio time and frequency dissemination is presented in [Blair, 1974], a more satellite specific survey is given in [Somayajulu, 1980]. The problem of these approaches is in general that they are subject to gross error [Mills, 1991]. Due to the recent advances in satellite navigation systems, many devices for computer synchronization are now available. An investigation on achievable accuracy and performance of such devices was performed in [Tu et al., 2000] [Lombardi et al., 2001] and [Jefferson et al., 1996]. A general problem for the applicability in academic ground station networks is, that special hardware (at least a GPS receiver) is required for this type of time synchronization, the accuracy of the time synchronization depends on the quality of the hardware. It can be assumed that there will not be a dedicated GPS device at each ground station in an academic network available.

4.2 Synchronization in academic ground station networks

This section presents an approach to solve the synchronization problem described in section 4.1.1. The objective is to synchronize different data streams on link layer level to identify parallel received frames, received from one satellite at several ground stations. The parallel received frames are then used to identify and resolve information gaps. As the content of the received data frames could be corrupted by transmission errors, only the time of reception can be used to relate the different

frames with each other. A very simple approach is to have timestamps on each received frame from the corresponding ground station and to use these timestamps to identify the duplicate packets. In the following the term *duplicate* is used for frames which were received in parallel. In figure 4.5 is the scenario depicted, where two ground station receive the same frame from a satellite. Ground station A receives the signal from the satellite at time $T_{A'}$ and records the corresponding time of reception at time T_A (i.e. there is a small system delay between the physical reception of the satellite signal and the recorded timestamp of the digital data frame). Ground station B stores the according timestamp T_B. It can be seen in the figure, that $T_A \neq T_B$, the reasons for that are discussed later in this section. It is now necessary to relate these times of reception to a signal sent from the satellite at time T_0.

One problem is, that the a signal from the satellite will not reach the individual ground stations at exactly the same time, there can be a difference of a few milliseconds. How is this delay induced? A deeper look at the receiving chain in figure 4.5 reveals two major influences: The time needed from the satellite to the ground station and the processing time of the ground station hard- and software: The time from the satellite to the ground station (i.e. signal propagation delay) only relates to the slant range and depends on the actual position of the satellite. For a LEO satellite it is a relatively short time interval between 3 and 13 milliseconds (depending if the satellite is at the horizon or at zenith, see also appendix A), it is a deterministic value and can be calculated from the satellite orbit. Unfortunately, the processing time of the ground station is not deterministic, it is rather a varying delay which is not known beforehand. Furthermore, the processing delay might be different on each individual ground station, because the processing of data frames can be realized on different layers (e.g. directly in hardware or on a high level operating system). Last point which needs to be considered is the transmission delay from the time of reception at the client stations (T_A, T_B) and the reception at the data management server ($T_A^{Internet}$, $T_B^{Internet}$). One might consider this delay as negligible, as the timestamps T_A and T_B contain already the required time of reception, but as it will be explained in more detail in section 4.2.1, the transmission delay over the network influences indirectly the time synchronization. This leads finally to the twofold, formalized synchronization problem:

Problem 1: The time of reception at the client stations (T_A, T_B,...) have to be used to identify the time T_0, when the signal was sent from the satellite. This

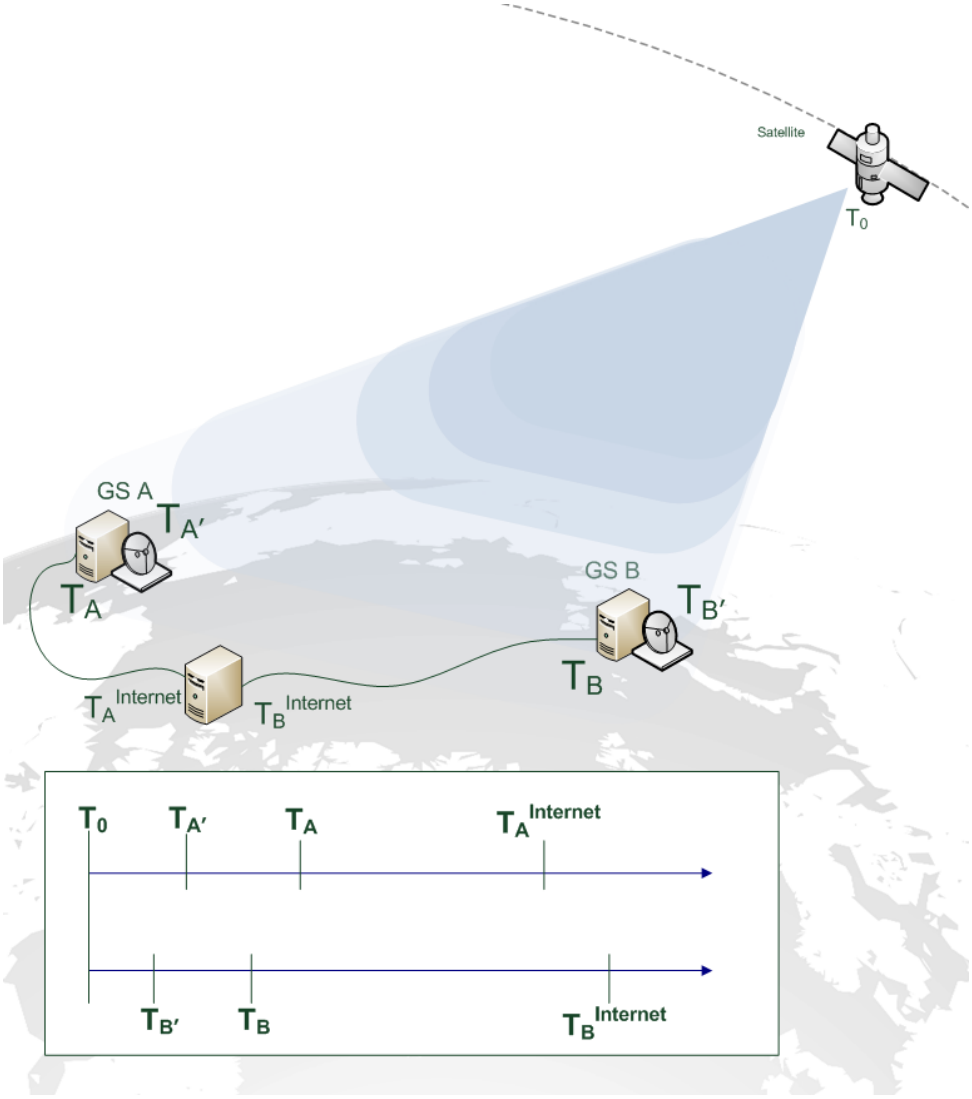


Figure 4.5: Synchronization problem in academic ground station networks

includes the determination of the propagation delay and the estimation of the processing time (respectively system delay) at each client stations. This problem is handled under the context of data synchronization in section 4.2.2 in detail.

Problem 2: If the time of receptions need to be correlated with each other, an important prerequisite is the synchronicity of the computer clocks on the client ground stations. Several mechanisms to synchronize computer clocks were introduced already in section 4.1.4. The accuracy of the synchronization process is typically influenced by the latency δ, therefore the transmission delays ($T_A^{Internet}$, $T_B^{Internet}$, etc.) also need to be taken into account. This problem is handled under the context of time synchronization in section 4.2.1 in detail.

Before the developed approaches are presented, a last remark about synchronization in ground station networks is necessary: Of course, there exist various, well proven solutions for clock synchronization, nevertheless they do not fully solve the problem of time synchronization in this scenario. The main reason for that is, that the ground stations of the network belong to individual institutes, they will not adapt or implement software just by request from another research institute. So, it could be possible that a ground station does not rely on any synchronization technique and it is just not possible to control this issue from the outside. Additionally, there is always the problem of trusting a timestamp if no control over the system itself is available. From that point of view it was decided to develop a time synchronization method, tailored for the usage in academic ground station networks, which does not need any external source for the required time synchronization.

4.2.1 Time synchronization between ground stations

Prerequisites

Before addressing in more detail time synchronization, some definitions related to synchronizing clocks in computer networks need to be introduced: The definitions are mainly taken from [Mills, 1993] [Mills, 1995] and [Moon et al., 1998].

The time of an event is an abstraction, which determines the ordering of events in a given timescale [Mills, 1993]. Computer clocks are often used in technical systems to compare the time of different events. The term clock refers here in general to computer clocks. Different representations for clocks exist, a simple model from David Mills is the representation as an oscillator in combination with a counter,

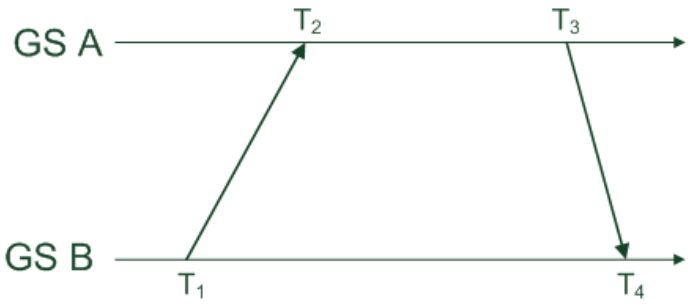


Figure 4.6: Synchronization procedure with timestamps

which records the number of oscillator cycles since initialization. The counter at any given time t is called its epoch or timestamp $T(t)$. Other researchers use a piecewise continuous function to describe a clock [Moon et al., 1998]. This continuous function is twice differentiable (except for a finite set of points) and can be used to determine the frequency and drift of a clock.

A standard method to synchronize two computer clocks exchanges data packets over an available network (typically the Internet). In figure 4.6 is such a synchronization attempt depicted, initialized from the client station GS B. The synchronization request contains the timestamp T_1, which was attached before transmitting the packet to the server (GS A). The time of reception on the server side is recorded in T_2 and a synchronization acknowledgement is prepared with its time of transmission T_3 added. The synchronization attempt is finalized when the client receives the acknowledgement and obtains T_4. The four timestamps (T_1,T_2,T_3,T_4) are now used to calculate the required parameters for synchronization.
The most important parameters of this scenario are the latency δ and the offset Θ. The latency is indicated in figure 4.7 as the end-to-end delay and is given through the timestamps by

$$\delta = (T_4 - T_1) - (T_3 - T_2) \tag{4.5}$$

The offset describes the absolute difference between two clocks. Typically the true offset Θ is distinguished from the estimated offset θ. The accuracy of the estimated offset depends on the size of the latency δ, which introduces a systematic delay when asymmetric transmission path are present. For more information about this issue please refer to [Mills, 1995]. The estimated offset can be calculated from

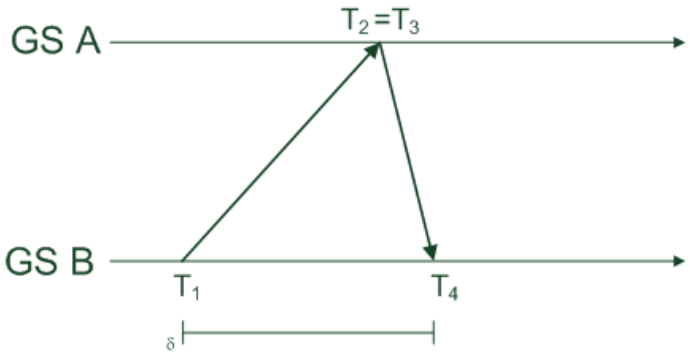


Figure 4.7: Synchronization procedure

a synchronization attempt by

$$\theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \tag{4.6}$$

Deeper analysis of the time synchronization problem takes also into account other clock characteristics and try for example to remove the skew (difference in oscillator frequency) [Moon et al., 1998].

It is well known [Mills, 1993], that the true offset Θ_t between two clocks for a synchronization attempt at epoch t lies between

$$\theta_t - \frac{\delta_t}{2} \leq \Theta_t \leq \theta_t + \frac{\delta_t}{2} \tag{4.7}$$

In the context of NTP the two endpoints of a synchronization attempt are called client and server, where the servers run at a very accurate time (e.g. attached to an atomic clock), the clients try to synchronize their own clocks with these servers (the so called NTP server). The role allocation is a bit different for the entities in academic ground station networks: The client stations are ground stations contributing received data frames to the network. Each client station has an own computer clock which is used to obtain timestamps for received data frames. On the other side there is a central server, which is responsible for collecting received data frames in the network and for processing the retrieved data. The central server can be connected to a ground station, but can be in principle any computer connected to the ground station network. The central server as well has its own clock. In the further text, the terms *client station* and *central server* are used to distinguish these roles. In the scenario assumed in this work, the clocks of the central server and the client systems are not synchronized by a protocol like NTP or SNTP. The introduced data

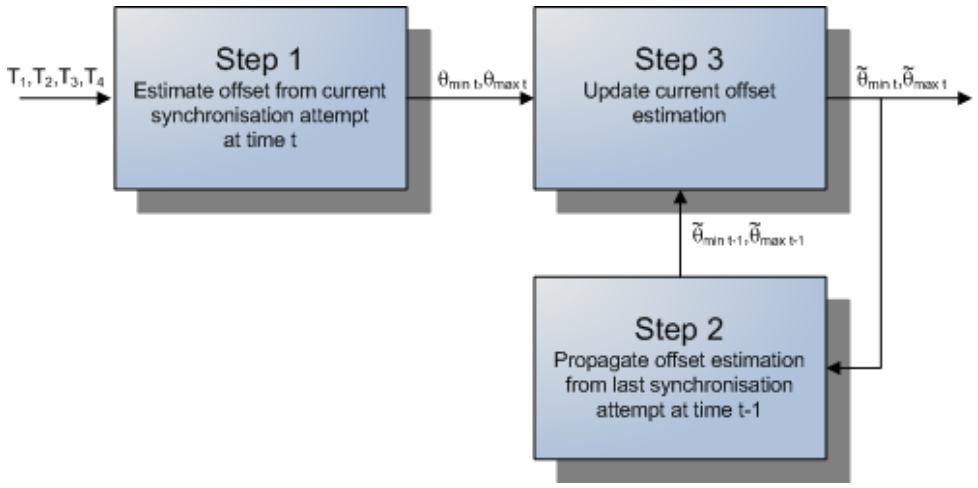


Figure 4.8: Offset estimation process

management system tries to estimate the offset θ between the different clocks in the ground network with the algorithm presented in the following section. This is necessary to correlate the timestamps of the received frames with each other. Thus, the first objective is to estimate the offset of each client station to be able to synchronize the received data frames from the satellite.

Offset estimation algorithm

This section presents an approach to determine the offset between a client station clock and the central server clock. The procedure described in the following is applied to one single station, in the complete network the whole procedure has to be performed for each client station. The algorithm works in three different steps (see figure 4.8) and starts in *Step 1* with the derivation of the estimated minimum offset θ_{min_t} and estimated maximum offset θ_{max_t} from to the last synchronization attempt at time t.

Step 1: Offset estimation from synchronization attempt at time t:

The estimated offset is derived from the synchronization attempt at time t, which delivers the four timestamps T_1,T_2,T_3 and T_4 as described before. To obtain the minimum offset θ_{min_t} and the maximum offset θ_{max_t} the following equations are

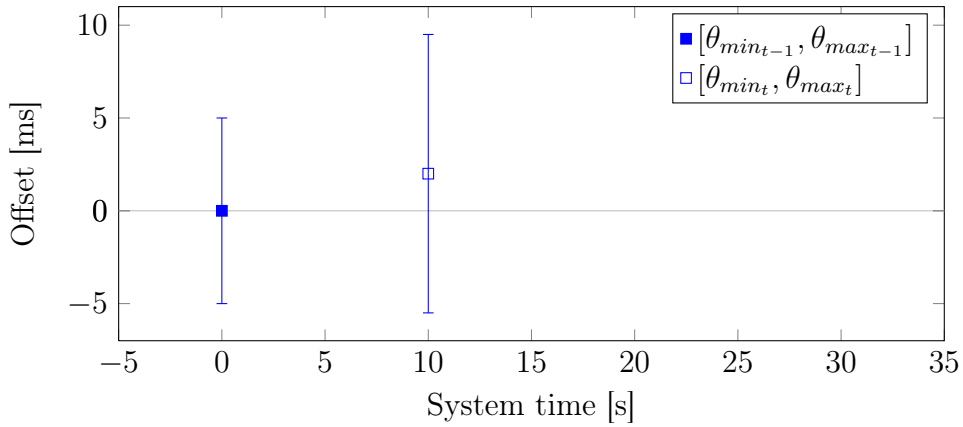


Figure 4.9: Offset estimation based on current sample

used:

$$\theta_{min_t} \equiv \theta_t - \frac{\delta_t}{2} = T_{3_t} - T_{4_t} \tag{4.8}$$

$$\theta_{max_t} \equiv \theta_t + \frac{\delta_t}{2} = T_{2_t} - T_{1_t} \tag{4.9}$$

It is obvious from equations 4.8 and 4.9 that

$$|\theta_{max_t} - \theta_{min_t}| = \delta_t \tag{4.10}$$

This estimation is only based on the last synchronization attempt at time t (see figure 4.9). As the accuracy of the estimation depends mainly on δ_t, the algorithm tries in the next step to correct this estimation with the information obtained from the previous synchronization attempt. To use the information from the previous synchronization attempt, the last offset estimation at time $t-1$ is propagated and the current sample at time t accordingly corrected. The result of the correction is a time interval $\left[\tilde{\theta}_{min_t}, \tilde{\theta}_{max_t}\right]$, which is closer to the true offset Θ between the client station and central server. In the following denotes "˜" the corrected offset. The propagation of the previous estimated offset $\left[\tilde{\theta}_{min_{t-1}}, \tilde{\theta}_{max_{t-1}}\right]$ is performed in *Step 2*:

Step 2: Propagation of the last estimated and corrected offset at time $t-1$:

To update the actual offset estimation, the information from the previous offset estimation is used. It has to be considered, that the local clock might have drifted

since last synchronization at time $t-1$, therefore the last estimated and corrected offset $[\tilde{\theta}_{min_{t-1}}, \tilde{\theta}_{max_{t-1}}]$ is propagated to

$$\tilde{\theta}'_{min_{t-1}} = \tilde{\theta}_{min_{t-1}} + m_{min_{t-1}} \left(T_{4_t} - T_{4_{t-1}}\right) \tag{4.11}$$

$$\tilde{\theta}'_{max_{t-1}} = \tilde{\theta}_{max_{t-1}} + m_{max_{t-1}} \left(T_{4_t} - T_{4_{t-1}}\right) \tag{4.12}$$

where $m_{min_{t-1}}$ and $m_{max_{t-1}}$ describe the minimum and maximum approximation of the drift. The approximation of the drift is calculated with

$$m_{min_t} = min\left(m_{avg_t}, \frac{m_{min_{t-1}} + m'_{min_t}}{2}\right) \tag{4.13}$$

$$m_{max_t} = min\left(m_{avg_t}, \frac{m_{max_{t-1}} + m'_{max_t}}{2}\right) \tag{4.14}$$

with a given average drift of

$$m_{avg_t} = \frac{m_{min_{t-1}} + m_{max_{t-1}}}{2} \tag{4.15}$$

and the helping variables m'_{max_t} and m'_{min_t}:

$$m'_{max_t} = min\left(\frac{\tilde{\theta}_{max_t} - \tilde{\theta}_{max_{t-1}}}{T_{4_t} - T_{4_{t-1}}}, (m_{avg_t} + m_{max_0})\right) \tag{4.16}$$

$$m'_{min_t} = max\left(\frac{\tilde{\theta}_{min_t} - \tilde{\theta}_{min_{t-1}}}{T_{4_t} - T_{4_{t-1}}}, (m_{avg_t} + m_{min_0})\right) \tag{4.17}$$

The average drift of the clock in equation 4.15 is used as a lower limit for the approximated maximum drift m_{max_t} to avoid negative values. Similar is the value in equation 4.16 used as an upper limit for the drift, depending on the initialization value m_{max_0}. For the approximated minimum drift in equation 4.13 is the interpretation analogue.

For initialization the following values are used: $\tilde{\theta}_{min_0} = \theta_{min_0}$ and $\tilde{\theta}_{max_0} = \theta_{max_0}$, i.e. the first estimated and corrected offset relies only on the first synchronization attempt at time $t = 0$. The minimum and maximum drift is initialized with $m_{max_0} = 5 \cdot 10^{-5}$, respectively $m_{min_0} = -5 \cdot 10^{-5}$, which is equal to a drift of 5 milliseconds within a time interval of 10 seconds. This is a worst case assumption, which is not a restriction for any computer clock [Murdoch, 2006].

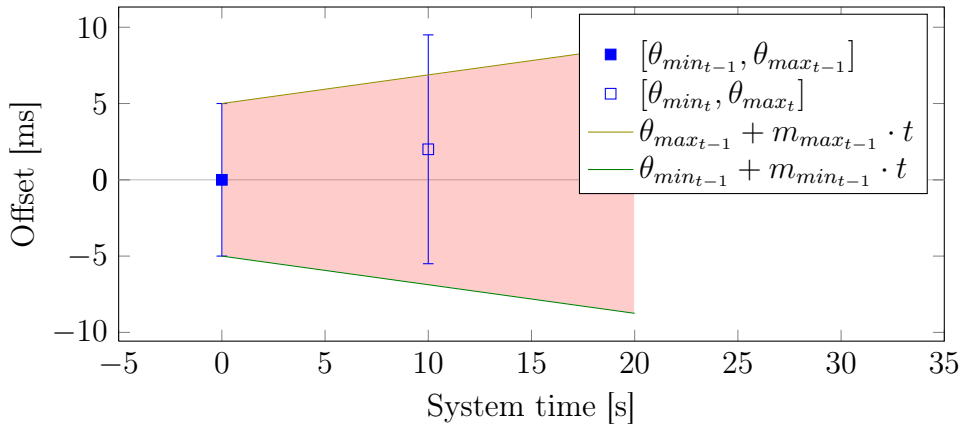


Figure 4.10: Offset propagation from last sample

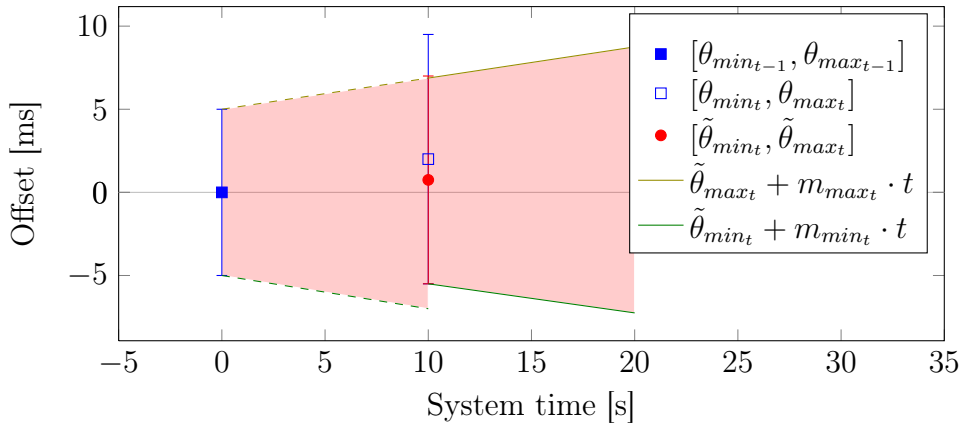


Figure 4.11: Estimated and corrected offset at time t

With the propagated offset (see figure 4.10) from the last synchronization, the algorithm corrects in *Step 3* the estimated offset:

Step 3: Correction of the estimated offset:

The last step of the algorithm corrects the estimated minimum offset $\tilde{\theta}_{min_t}$ and maximum offset $\tilde{\theta}_{max_t}$ to

$$\tilde{\theta}_{min_t} = min\left(\theta_{min_t}, \tilde{\theta}'_{min_{t-1}}\right) \tag{4.18}$$

$$\tilde{\theta}_{max_t} = max\left(\theta_{max_t}, \tilde{\theta}'_{max_{t-1}}\right) \tag{4.19}$$

The algorithm compares the estimated offset (which might be relatively large due to high latency δ), with the propagated offset from the last estimation, which considers only the drift of the clock (see figure 4.11).

Remark: If the estimated offset is disjunct with the last estimated and corrected offset estimation, i.e. $[\theta_{min_t}, \theta_{max_t}] \cap \left[\tilde{\theta}_{min_t}, \tilde{\theta}_{max_t}\right] = \emptyset$, it can be assumed that the computer clock was affected by a time correction from another source (for example a time synchronization protocol like NTP). In this case, the value for the estimated offset $[\theta_{min_t}, \theta_{max_t}]$ will not be corrected and the approximated drift will be initialized to $m_{min_t} = m_{min_{t-1}} + m_{min_0}$ respectively $m_{max_t} = m_{max_{t-1}} + m_{max_0}$. For a more detailed investigation of the synchronization algorithm please refer to [Schmidt et al., 2010].

The result of this section is an estimated and corrected offset $\left[\tilde{\theta}_{min_t}, \tilde{\theta}_{max_t}\right]$ for one client station, i.e. an estimated time difference between the central server clock and a client ground station clock. The proposed offset estimation algorithm is not as susceptible to delay variations as the SNTP estimation procedure, because information from the previous estimations is included. The offset has to be estimated for each client station separately in the network. The estimated offset is a very critical parameter, as it is used from the central server to estimate the time of reception of each data frame at each client station.

Estimated time of reception

In the next step, the estimated offset for a ground station A is used to determine the estimated time of reception. The estimated offset describes only the "difference" between two clocks, now this knowledge is used to derive the time of reception in a global timescale. If a data frame F_i is received at ground station A at epoch $T(F_i^A)$, an estimated time of reception can be obtained from

$$T_{min}(F_i^A) = T(F_i^A) + \tilde{\theta}_{min_t} + m_{min}\left(T(F_i^A) - T_{4_t}\right) \tag{4.20}$$

$$T_{max}(F_i^A) = T(F_i^A) + \tilde{\theta}_{max_t} + m_{max}\left(T(F_i^A) - T_{4_t}\right) \tag{4.21}$$

In summary, this section presented an algorithm for time synchronization in ground station networks. The outcome of this section is on the one hand an estimated offset θ for each client station in the network, which is used to obtain a

synchronized global timescale in the ground station network. On the other hand, the synchronized timescale is used to derive the time of reception $T(F_i^{gs})$ of each data frame on that global timescale. More specifically, an estimated time of reception in terms of a time interval $\left[T_{min}(F_i^A), T_{max}(F_i^A)\right]$ is calculated. Referring to the formalized problem in section 4.1.1, this solves the demand for proper time synchronization (see also **Problem 2** on page 106). The next section discusses how that approach is used to synchronize the data frames respectively how the time of reception can be used to estimate the time of transmission.

4.2.2 Data synchronization on frame level

When the clocks of the individual ground stations are synchronized, i.e. the offset with respect to the central server is estimated for each client station, the data frames need to be synchronized. That means, identical frames should be identified only based on their time of reception. As explained on page 103, the time of reception is biased by a system delay, which has to be compensated:

System delay compensation

The system delay is estimated for each frame. Hence, in the first step the average time of reception $\overline{T}^A(i)$ of a frame F_i^A is calculated. This is done with the equations for the estimated time of reception derived in section 4.2.1.

$$\overline{T}^A(i) = \frac{T_{min}(F_i^A) + T_{max}(F_i^A)}{2} \tag{4.22}$$

Without the loss of generality it is assumed that ground station A was not the first station which received frame F_i. Another ground station submitted this data frame to the central server at $\overline{T}^0(i)$. Now the difference in time of reception between ground station A and the first station, which submitted the identical frame, is calculated as

$$\Delta^A(i) = \overline{T}^A(i) - \overline{T}^0(i) \tag{4.23}$$

The $\Delta^A(i)$ value is a first estimation of the system delay at ground station A, consisting of the time needed to receive, process and forward the received frame F_i (offset and propagation delay are already included in $\overline{T}^A(i)$). But of course, this system delay changes over time due to changes in slant range and processing load. Therefore, the variable S_{delay}^A is introduced to propagate the system delay over a

period of time. A moving average filter is used to adapt the system delay S^A_{delay} for ground station A:

$$S^A_{delay}(k) = \frac{\alpha(k) \cdot S^A_{delay}(k-1) + \Delta^A(k)}{\alpha(k) + 1} \tag{4.24}$$

with

$$\alpha(k) = \left|\Delta^A(k) - \Delta^A(k-1)\right| + 1 \tag{4.25}$$

k represents the number (or counter) of received frames at ground station A, i.e. after each new frame the system delay S^A_{delay} is updated. The weight α prevents the system delay to be influenced from large peaks, which can appear for single frames. For the case that A was itself the first ground station which submitted frame i, the term $\Delta^A(i)$ reduces to 0 and decreases the system delay S^A_{delay} by factor $\frac{\Delta^A(k-1)+1}{\Delta^A(k-1)+2}$. One restriction has be mentioned: In equation 4.23 is the difference of two identical frames, received at different ground stations, determined. This implies that it is known that these two frames are identical, which is only the case if the data content was not corrupted. If there is a bit error in one of these frames, it is not possible to identify them as identical. Therefore, the system delay variable S^A_{delay} is only updated, when two correct identical frames are received in the network. It is assumed that at least a few frames are received correctly during the contact window, the system delay S^A_{delay} will then be adapted by the moving average filter.
The result of this section is the variable $S^A_{delay}(k)$, which represents the system delay of a ground station A. This delay includes the transmission time from the satellite to Earth as well as the processing time at the ground station, which might be influenced strongly by several factors, like hardware devices or operating system components. Using this approach, the proposed system is now able to compensate the system delay, i.e. to estimate if frames were received nearly at the same time (unbiased by data processing and forwarding). Finally, in the last step of the data synchronization approach, the data frames are grouped in different data sets.

Data frame synchronization

In the last step, the algorithm identifies identical data frames only based on the time of reception and estimated delays. If a data frame F_j is received at ground

station B, the average time of reception $\overline{T}^B(j)$ of that frame is given by equation 4.22. Additionally, for ground station B is a system delay $S^B_{delay}(j-1)$ available (see equation 4.24). From these parameters the time interval $T_{search}(j)$ is determined, which is used to identify duplicate frames:

$$T_{search}(j) = \left[\left(\overline{T}^B(j) - S^B_{delay}(j-1) - D_{LEO}\right), \left(\overline{T}^B(j) - S^B_{delay}(j-1) + D_{LEO}\right)\right] \tag{4.26}$$

where D_{LEO} describes the maximum transmission delay when the satellite is at the horizon (maximum slant range) and is for a LEO orbit approximately 15 milliseconds (see appendix A). This time interval $T_{search}(j)$ is used to search for other data frames, which were received in parallel from another ground stations. If this is the case, i.e. the average time of reception of a second data frame l received at ground station C ($\overline{T}^C(l)$) lies within $T_{search}(j)$, it is assumed that these frames are identical. The algorithm groups these in the same data set S, respectively if

$$\overline{T}_B(j) - S^B_{delay}(j-1) - D_{LEO} \leq \overline{T}_C(l) \wedge \tag{4.27}$$

$$\overline{T}_C(l) \geq \overline{T}_B(j) - S^B_{delay}(j-1) + D_{LEO}$$

is fulfilled, both data frames are assigned to the same data set S, which contains then

$$S = \left\{F^B_j, F^C_l\right\} \tag{4.28}$$

The two frames in data set S are assumed to be duplicate frames received from two different ground stations in parallel. To search for more duplicate respectively identical frames, a common average time of reception of data set S is calculated with

$$\overline{T}_S = \frac{\sum^{gs}\left[\overline{T}_{gs} - S^{gs}_{delay}\right]}{m} \tag{4.29}$$

where m is the number of frames in data set S and $\overline{T}_{gs}$ is the average time of reception for all frames in that data set S.
For the case that several frames lie within $T_{search}(j)$, the closest frame to the center of that interval is used for initial search. After that, the average time of reception

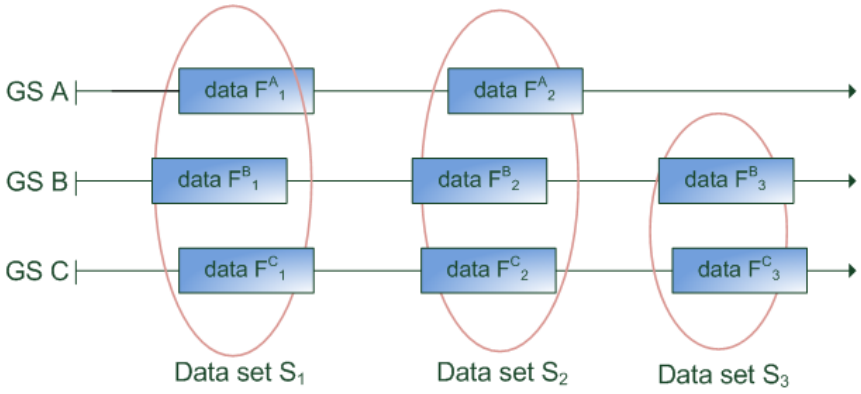


Figure 4.12: Identifying of data sets containing duplicate frames

$\overline{T}_S$ is calculated with equation 4.29, the algorithm searches iteratively for the closest data frame until no more frames lie within T_{search}. From equation 4.29 it is obvious, that the search interval $T_{search}(j)$ is decreased steadily with each new data frame assigned to the data set.

The result of the proposed data synchronization algorithm are data sets S_1 to S_p, each containing at least one data frame. All data frames comprised in a single data set were identified as "identical", here identical means that these frames were assigned to the same time of transmission, but might differ in data content due to transmission errors (c.f. figure 4.12). These data sets are used as input for the data combination algorithm introduced in section 4.3. Referring to the formalized problem in section 4.1.1, this solves the issue of data synchronization (see also **Problem 1** on page 104).

Finally, a short summary of the complete synchronization approach is presented: In section 4.2.1, a time synchronization algorithm was developed. Its main benefit is the ability to synchronize ground station computers without the need of external time sources (like NTP). This algorithm guarantees synchronicity of the computer clocks, which is required to compare the timestamps of the received frames. A data synchronization algorithm was presented in section 4.2.2, it is divided in two approaches to solve the subproblems of *system delay compensation* and *data frame synchronization*. *System delay compensation* delivers a solution to the problem statement of equation 4.1, i.e. it derives an estimated time of transmission $\tilde{T}(F_i^{gs})$ from the time of reception $T(F_i^{gs})$. The *data frame synchronization* problem is related to the identification of duplicate frames from the estimated time of transmission

$\tilde{T}(F_i^{gs})$ to group them in data sets (see equation 4.2 in the problem description). The proposed approach is the basis for the data combination approach described in the next section. Performance analysis of the time and data synchronization approaches are handled in the experiment section 4.5.

4.3 Data combination in academic ground station networks

The term data combination is used here in a very abstract way and was introduced in section 4.1.1 to clarify the problem scope. The underlying scenario assumes a single satellite, which sends data frames to multiple ground stations connected to a network. These stations receive the data frames in parallel and forward them to the central server, which uses the time of reception to identify duplicate frames to group them in data sets. If transmission errors have affected the data frames, the objective of the data combination algorithm is to retrieve the original data frame from the available data frames. It is avoided to use for that approach the term "data fusion" as it is often used in the context of sensor data fusion. Sometimes the term also appears in database modeling. The work of Stolarski [Stolarski, 2009] uses for a similar approach the term data comparison. Throughout this work the process of combining data frames received in parallel from a satellite is referred as data combination.
In the following, three different approaches for data combination are presented:

- Ground station majority voting, GSNV (section 4.3.1)
- Brute force correction, BFC (section 4.3.2)
- Single bit error correction in AX.25, SBEC (section 4.3.3)

4.3.1 Ground station majority voting approach

To combine the information of the obtained data sets (each set S_k contains duplicate data frames), a simple strategy is pursued to reconstruct the original data content: Each ground station is subject to local transmission errors, it is possible to use duplicate, i.e. in parallel received, data frames to detect and correct local errors. Local errors are only observed at single ground stations and have different error sources,

e.g. hardware performance, different slant range etc. Experiments by Stolarski showed already the occurrence of such local bit errors in a radio transmission from a weather balloon (see section 4.1.3). The working principle behind the proposed Ground Station Majority Voting (GSMV) algorithm is relatively simple: Compare the received data frames bit by bit to detect flipped bit errors. In contrast to the work of Stolarski [Stolarski, 2009], the novel approach in this work is the usage of information contained in the data link layer, not only the bit stream itself. Hence, the proposed GSMV algorithm compares the data frames on a higher protocol layer to reconstruct information gaps more efficiently. The input for the GSMV algorithm are the data sets S_k identified from the data synchronization algorithm described in section 4.2.2. These sets contain only duplicate, but maybe corrupted data frames. If a data set S_k has n duplicate frames assigned, the bit value of bit i is calculated by

$$Bit(i) = \begin{cases} 0 & \text{if } d(i) < 0.5 \\ 1 & \text{if } d(i) \geq 0.5 \end{cases} \tag{4.30}$$

where $d(i)$ counts the sum of the bits divided by the number of frames:

$$d(i) = \frac{\sum_{j=1}^{n} \mathrm{Bit}_j(i)}{n} \tag{4.31}$$

$\mathrm{Bit}_j(i)$ describes the i-th bit of frame j. It should be mentioned, that for the case that an even number of ground stations collect data from the same satellite, it is possible that no unique majority for a bit value exists (this corresponds to the case of $d(i) = 0.5$). As a result, the algorithm assumes for that case with a probability of 0.5 the wrong bit value. This issue is discussed in more detail in the performance analysis of the algorithm in section 4.4.1. If an odd number of data frames were assigned to a data set, $d(i) \neq 0$ per definition.

Key aspect of this algorithm is now, that the FCS field of the AX.25 protocol can be used to check if the obtained data frame is correct. This is only possible as the GSMV algorithm scans data frames on the link layer instead of pure bit streams. The FCS field is calculated in a standard AX.25 link from a TNC device before transmission and is automatically attached to each frame leaving the sender. The FCS calculation procedure of AX.25 uses the CRC-16-CCITT standard to obtain a value of size 2 byte. When a frame is obtained at the receiver side, it is possible to detect transmission errors by calculating the FCS bytes. Hence, for the complete

frame (including the attached FCS field from the sender), the FCS procedure is performed to obtain the FCS value at the receiver side (denoted as FCS_R). The CRC-16-CCITT algorithm assumes the frame to be error free when equation 4.32 is fulfilled.

$$FCS_R \,\&\, 0xFFFF = 0x0F47 \tag{4.32}$$

This property is used from the GSMV algorithm to test if the data combination procedure (equation 4.30 and 4.31) was successful, i.e. whether the bits of the data set S_k were combined to a single correct data frame. If the combined frame does not fulfill equation 4.32, the frame is marked as corrupted.
The presented GSMV approach is a solution to the problem description specified in equation 4.3. A deeper analysis of the performance of this algorithm is presented in section 4.4.1.

Applicability and implementation

The GSMV algorithm takes advantage of the observation that transmission errors affect the parallel received data frames particularly only locally. The occurrence of local bit errors has several reasons: The first reason is, that the measured strength of the radio signal at the ground station (signal to noise ratio, SNR) varies over time. The SNR depends on the distance between ground station and satellite, which is different for geographically distributed ground stations. Furthermore, the signal is affected by different atmospheric disturbances. A second reason for local transmission errors (and probably more critical factor) is the inhomogeneous hardware used in the participating stations. Especially in the context of academic ground station networks, the hardware shows quite different performance characteristics, due to variety of antennas, modems and radio equipment used. Additionally pointing inaccuracies, cable losses and other noise sources influence the reception of the signal at each station. Therefore, a distinct bit error rate can be expected for each of the individual stations.
Also the satellite projects UWE-1 and UWE-2 from Würzburg confirm this observation, as the rate of successfully decoded beacons varied strongly between the different ground stations. The variance of the measured bit error rate from UWE-1 at the ground station in Würzburg was relatively high [Schmidt and Zeiger, 2006] [Schmidt, 2006] and indicates the strong influence of error sources. Furthermore,

experiments conducted in cooperation between University of Würzburg and University of Tokyo [Oda and Nishikawa, 2007] showed that the set of received data frames in overlapping regions differ significantly.
One of the major influences for local errors (as discussed above) is the performance characteristics of the ground station components. The performance of a software modem (for example [Sailer, 2000]) depends strongly on the soundcard used, which is not the most suitable device for software demodulation of a satellite signal. Nevertheless, software modems are used more and more in academic ground stations as they provide a high degree of flexibility [Rodriguez-Osorio et al., 2008]. Using low performance COTS components is a very uncommon strategy in classic space technology engineering, where typically only the highest standard components are used for reliability reasons. Also in ground systems primarily robust and qualified (but also expensive) equipment is employed. The approach presented in this work tries to combine the limited capabilities of many low-cost stations to increase the performance. This is an important aspect, as the presented approach tries to optimize the operation of a satellite by using low performance entities, the increase in performance is achieved by combining their performance in an intelligent way. Of course it would be possible to increase the performance even more by using high-end components, but it is assumed that only low cost components are available and utilized.

An important issue, which has to be addressed for the implementation of the GSMV algorithm, is the discarding function of a standard TNC device, which is in many academic ground stations integral part of the receiving chain: A TNC discards by default data frames, when transmission errors are detected. This is realized through a simple FCS procedure: If the calculated 2 byte checksum is not valid, the complete frame will be discarded (i.e. not forwarded to the connected computer). Of course, it is then not possible to correct bit errors, because no access to the corrupted data exists. This problem can be solved in different ways: One solution is the usage of a TNC which is able to forward also corrupted frames. A second solution is the replacement of the TNC with a software modem, which provides more access to the incoming data frames. For more details related to this issue refer to [Zeiger et al., 2006] and [Schmidt and Schilling, 2010b].

4.3.2 Brute force method for data recovery

A second mechanism to correct errors in a received data frame, uses the redundancy contained in the communication protocol AX.25. The proposed Brute Force Correction (BFC) algorithm takes advantage of the fact that each AX.25 frame has a FCS attached, which enables the receiver to detect errors in the data frame. The working principle behind the BFC algorithm is relatively simple: The bits of the data frames are altered systematically until the correct FCS is obtained. For example, if a single bit error appears in a data frame of size n bytes, one can obtain the correct data frame by inverting all of the $8 \cdot n$ bits of the frame after each other and check if it leads to the originally received FCS field. If more errors are in the frame (unfortunately it is not known before how many errors are in the frame), the BFC algorithm has to try in the worst case all 2^{8n} possible bit combinations to get the correct FCS. This implies two major limitations for the BFC algorithm: First, as the number of possible combinations increases exponentially, the method is time consuming and not appropriate for large data sets. For AX.25 the maximum frame size of 512 byte fortunately restricts already the search space. The second, more critical limiting factor is the uniqueness of the obtained FCS field. As only 2^{16} FCS patterns (2 bytes) exist, it is not possible to allot 2^{8n} different error patterns in a bijective assignment. Therefore, it is possible to create the same FCS bits from two different data frames and there is theoretically no possibility to identify the original frame (except parts of the frame are fixed, like the protocol header). Hence, the applicability of the BFC algorithm strongly depends on the number of errors in the data frame.

To overcome the problem of exponential growth of correction possibilities, a strategy is applied to limit the search space of the BFC algorithm. This is achieved by restricting the bit positions, which are altered to obtain the correct FCS. It is assumed, that bits are correct, when all frames in the same data set are equal (all bits 0 or all bits 1), only the bit positions with diverging values are considered for the brute force correction (see figure 4.13). Thus, the runtime can be reduced significantly.

In conclusion, the BFC mechanism enables the reconstruction of data gaps, especially when a frame is only affected by a small number of bit errors. A detailed analysis of the correction capability is presented in section 4.5. Main issue is, that the number of bit errors must not exceed a certain threshold. Due to the limited

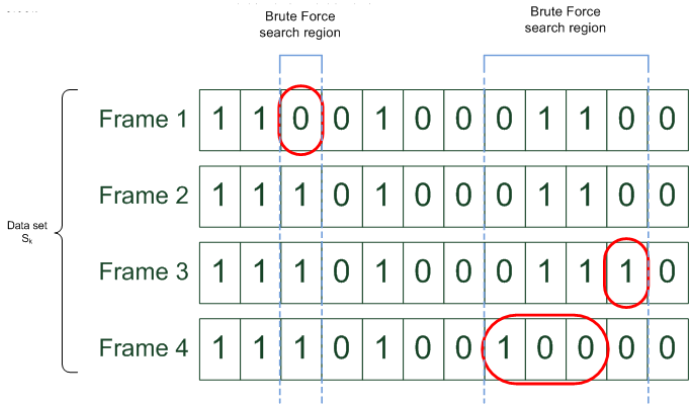


Figure 4.13: Reduced search space for BFC algorithm

frame size of the AX.25 specification and the strategy to reduce the search space, it is possible to correct errors within reasonable time (see analysis in 4.5.1).
One of the main advantages of the BFC algorithm is, that it can be used independently from the the GSMV algorithm. The proposed approach uses a two step approach: In a first step, the GSMV algorithm of section 4.3.1 is utilized to obtain a data frame from several parallel received frames. This reduces already the number of bit errors for the combined data frame. If the FCS field indicates, that bit errors are still in the frame, the BFC algorithm is used in the second step to search in the defined search regions for a possible solution. In the experiments in section 4.5.1 an upper threshold was derived, which restricts the runtime of the BFC algorithm by a maximum number of bits to be corrected.

4.3.3 Single bit error correction in AX.25

Forward error correction through frame checking sequence processing

The proposed GSMV data combination algorithm takes advantage of the redundancy induced through parallel data reception in the ground network. In the previous section the BFC algorithm was introduced, which uses the Frame Checking Sequence (FCS) bytes of the communication protocol to correct transmission errors. In this section is another approach presented, which uses the characteristics in the FCS calculation to identify transmission errors.
AX.25, the primarily used communication protocol in current academic ground station networks, is conform to the HDLC standard and implements the FCS procedure

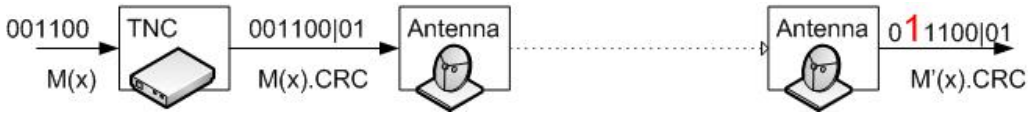


Figure 4.14: Transmission error in radio link

like described in section 4.3.1. Cyclic Redundancy Check (CRC) algorithms are often used for error detection. In the case of a transmission error, the corrupted data frame is requested again (ARQ mechanism). Nevertheless, the new algorithm presented in this work is able to correct errors by proper usage of the redundancy contained in the CRC bits. This is only possible if the generator polynomial was chosen appropriately. The error correction capability is of course limited and the correction of a bit error is only possible under certain conditions. The CRC-16-CCITT standard uses the generator polynomial $GP(X) = x^{16} + x^{12} + x^5 + 1$ and it will be shown in the following, how it can be used for single bit error correction. The algorithm is only elaborated in detail for the AX.25 protocol, for more information about error correction with CRC algorithms refer to [Shukla and Bergmann, 2004]. The basic principle of the proposed algorithm is an error correction based on a look-up table, which contains characteristic bit error patterns. These patterns can be generated from the generator polynomial offline and will be used to correct occurring single bit transmission errors online with the generated look-up table.

The working principle of a CRC algorithm bases on polynomial division: A given message $M(X)$, which is transmitted from a sender to a receiver, can be represented in polynomial notation (for example 101100 is expressed as $M(X) = 1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0$) and divided by the generator polynomial $GP(X)$. As already mentioned before, the CRC-16-CCITT standard uses the generator polynomial $GP(X) = x^{16} + x^{12} + x^5 + x^0$. The remainder $M(X)\ mod\ GP(X)$ is appended to the message as checksum, the so called CRC bits (or alternatively in AX.25 called FCS). The remainder is smaller than the the generator polynomial, if the polynomial degree of $GP(X)$ is 16, the checksum polynomial has a maximum degree of 15 (i.e. 2 bytes in binary representation). The standard procedure in a AX.25 supporting TNC is the division of the outgoing message $M(X)$ of length l bits by $GP(X)$. The calculated checksum bits (CRC) are attached to the message before transmission (see figure 4.14). The transmitted message has then a length of $l + 16$ bits.

It can be shown now, that a single bit error in the data frame leads to a characteristic error pattern in the CRC field. It is possible with the error pattern to locate the position of the corrupted bit and retrieve the original data frame. The error pattern can be calculated beforehand and stored in a look-up table. Such a procedure is often realized in hardware [Pan et al., 2007].
When a data frame of length l bit is sent, it is possible to calculate l different error patterns with only a single bit error. To show that the error patterns are unique, the transmitted message $M(X).CRC$ is considered (the "." operator describes a simple concatenation). The CRC bits are calculated by

$$CRC = M(X)\ mod\ GP(X) \tag{4.33}$$

First, the case of a single bit error in the message field $M(x)$ is investigated. The receiving ground station obtains then $M'(X).CRC$ (see figure 4.14). The received message is just a combination of the original message $M(X)$ with an error sequence $E_i(X)$, which is 0 everywhere except the i-th bit (i.e. x^i in polynomial notation). So, at the receiver side the message is

$$M'(X) = M(X) + E_i(X) \tag{4.34}$$

and one can calculate for the received message $M'(X)$ the checksum CRC_{Error} with equation 4.33. This can be transferred to

$$CRC_{Error} = [(M(X)\ mod\ GP(x)) + (E_i(X)\ mod\ GP(X))] \tag{4.35}$$

So, one obtains

$$CRC_{Error} = CRC + E_i(x)\ mod\ GP(X) \tag{4.36}$$

and can be modified to

$$CRC \oplus CRC_{Error} = E_i(x)\ mod\ GP(X) \tag{4.37}$$

It is easy to see, that this equation can be used to determine the position of a single error, as CRC is the transmitted checksum and CRC_{Error} is the calculated one (so both are known from the receiver). By comparison with all possible error patterns $E_i(X)\ mod\ GP(X)$, the position of the error is obtained.
The second case, which has to be handled, is a single bit error in the CRC field.

Due to the structure of the generator CRC-16-CCITT polynomial, the calculated CRC field changes at minimum 3 bits when a single bit error in the message field occurs. If there is only a single bit failure in the CRC field, this indicates that no transmission error occurred in the message field.

Error correction capabilities of CRC-16-CCITT

The error correction with a CRC respectively FCS field is only possible under certain circumstances: All error patterns are required to be unique. As the number of available single bit error patterns is defined by the length l of the frame (in the case of AX.25 $l = 4096$, defined by the maximum frame length of 512 byte) and the possibility to define with a checksum of 2 byte a number of $2^{16} = 65536$ distinct patterns, it is possible to assign the patterns bijective. Therefore, single bit forward error correction is possible when the AX.25 protocol is used. Due to the characteristics of the CRC-16-CCITT generator polynomial all these patterns are unique.
To create the look-up table, the following characteristic error patterns are calculated with $i = 1...l$:

$$Pattern_i = \{M(X) \cdot x^i\} \; mod \; GP(X) \tag{4.38}$$

All calculated error patterns $Pattern_i$ are stored in a table at the central server, if only a single bit is flipped in a data frame, the proposed algorithm can reliably obtain the original data frame. The strength of this approach is again, that it can be used in combination with the GSMV algorithm. The implemented system (see figure 4.15) tries in the first step to reduce bit errors with the GSMV algorithm. After combination of the different data frames, the system checks if the frame still contains bit errors. This would be the case if $CRC \oplus CRC_{Error} \neq 0$. Lets assume the result of $CRC \oplus CRC_{Error} = c$, the systems tries then to obtain a solution with the BFC algorithm. If it is not possible to derive a unique solution through BFC, the look-up table of the single bit error correction is used to search for a possible solution, i.e. the algorithm checks if an entry $Pattern_i = c$ exists. If this is the case, a single bit error might have affected the data and forward error correction is possible by altering the bit at position i. This algorithm is named Single Bit Error Correction (SBEC). Nevertheless, the solution of the SBEC algorithm has to be marked as "possible solution", as this approach works only reliable when only a single bit error occurs. A deeper analysis of the performance of the SBEC algorithm

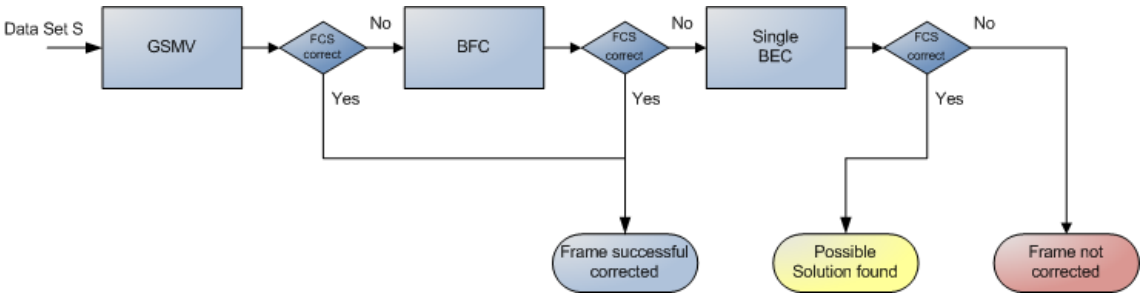


Figure 4.15: Combined error correction system

is presented in section 4.5.1.

The presented approaches for data combination and correction were tailored to the special needs of academic ground station networks. The presented GSMV algorithm is a solution to the problem description of section 4.1.1. The introduced BFC and SBEC algorithms further increase the performance of the GSMV algorithm by eliminating remaining transmission errors. A detailed analysis of the performance of the GSMV algorithm is presented in the next section. A performance evaluation of the complete implemented system (time and data synchronization, data combination) is presented in section 4.5.

4.4 Performance analysis

An estimation of the achievable performance of the proposed data management approach is presented in this section. Especially the performance of the GSMV data combination algorithm can be estimated with statistical means, which is explained in the following.

4.4.1 Performance of the data combination algorithm

To evaluate the performance of system, the performance criterion has to be defined in the first place. In this work, the performance is measured as the decrease of the Packet Error Rate (PER) after application of the GSMV algorithm. The decrease in PER is a reasonable criterion, as the communication link between satellite and ground station is affected stronger by transmission errors than in terrestrial communication. Therefore, the PER is significant higher in wireless communication links.

Moreover, a direct relation between SNR and the Bit Error Rate (BER) exists. To calculate the PER of a communication link, one method is to use the probability p, which describes the probability that a single bit is corrupted ($0 < p < 1$). The probability that a complete data frame with the length of l bits is received incorrect (i.e. the frame contains at least one bit error) is

$$PER(p, l) = \left(1 - (1 - p)^l\right) \tag{4.39}$$

The BER p of a satellite link is influenced from many parameters, for example transmit power, distance, noise, antenna gain, frequency, etc. Typical desired values for satellites range between 10^{-4} to 10^{-7} [Turner, 2008]. Especially in the research field of small satellites, the observed BER can be even worse, as the limited power and restricted attitude control capabilities have an influence on the link quality [Schmidt et al., 2007]. The algorithm described in section 4.3.1 reduces the bit error rate by conducting the correct bit value by majority voting. Nevertheless, a bit can still be corrupted if the majority of the involved ground stations received the bit incorrect. The probability that a number of m ground stations received a bit not correct can be calculated for a total number of n ground stations as

$$BER(n, m, p) = \binom{n}{m} p^m (1 - p)^{n-m} \tag{4.40}$$

A bit is only determined incorrect, when more then half of the ground stations received the bit incorrect. If one considers an odd number n of ground stations in the network, this would be the case for $m = \frac{n+1}{2}$. To obtain the BER value, one has to add the probability from $m = \frac{n+1}{2}$ up to $m = n$, as for all these cases the bit is calculated incorrect. Hence, the probability for an incorrect bit using the GSMV algorithm is

$$BER(n, p) = \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{\frac{n+1}{2} + i} p^{\frac{n+1}{2}+i} (1 - p)^{\frac{n-1}{2}-i} \tag{4.41}$$

It is now possible to calculate the PER for a data frame with the result from equation 4.41. Again, the case with an odd number of ground stations in the network is considered first. The PER is then determined as

$$PER_{odd}(n,p,l) = \left(1 - \sum_{i=0}^{\frac{n-1}{2}} \binom{n}{\frac{n+1}{2}+i} p^{\frac{n+1}{2}+i} (1-p)^{\frac{n-1}{2}-i}\right)^{l} \tag{4.42}$$

If the number of ground stations in the network is an even number, the PER can be calculated very similar, but a special case exists if exactly half of the received bits are corrupted (i.e. $m = \frac{n}{2}$):

$$PER_{even}(n,p,l) = \left(1 - \left[\binom{n}{\frac{n}{2}} p^{\frac{n}{2}} (1-p)^{\frac{n}{2}} \frac{1}{2} + \sum_{i=1}^{\frac{n}{2}} \binom{n}{\frac{n+1}{2}+i} p^{\frac{n}{2}+i} (1-p)^{\frac{n}{2}-i}\right]\right)^{l} \tag{4.43}$$

In this case of $m = \frac{n}{2}$, the same amount of votes for both solutions exist and no majority can be determined. This uncertainty is expressed in equation 4.43 by the factor $\frac{1}{2}$.
The results from equation 4.42 and 4.43 are combined in

$$PER_{GSMV}(n,p,l) = \begin{cases} PER_{odd}(n,p,l) & \text{if } n \bmod 2 = 0 \\ PER_{even(n,p,l)} & \text{else} \end{cases} \tag{4.44}$$

Using equation 4.44, it is possible to calculate the decrease of the PER for a ground station network of n stations, assuming an average BER of value p and a predefined frame length of l bits. It is obvious from equation 4.39, that a larger frame length l increases the PER. In figure 4.16 and 4.17 the estimated performance for different sizes of ground station networks, ranging from $n = 3$ to 9, is shown. The graph $n = 1$ depicts the original PER expected, if only one single ground station is available to receive the data frames. In this case, the PER increases already dramatically if the BER grows larger than 10^{-4}. If the GSMV algorithm is used, the packet error rate is still nearly 0 for a BER of 10^{-2}, even if only 5 ground stations contribute to the network. The performance can be increased even more by using 7 or more ground stations. The difference between figure 4.16 and 4.17 is only the used data frame length (256 respectively 512 byte), which represent typical values of data frames in AX.25 connections (the maximum frame size is restricted to 512 byte).
The graphs show, that it is possible to decrease the PER by data comparison in a

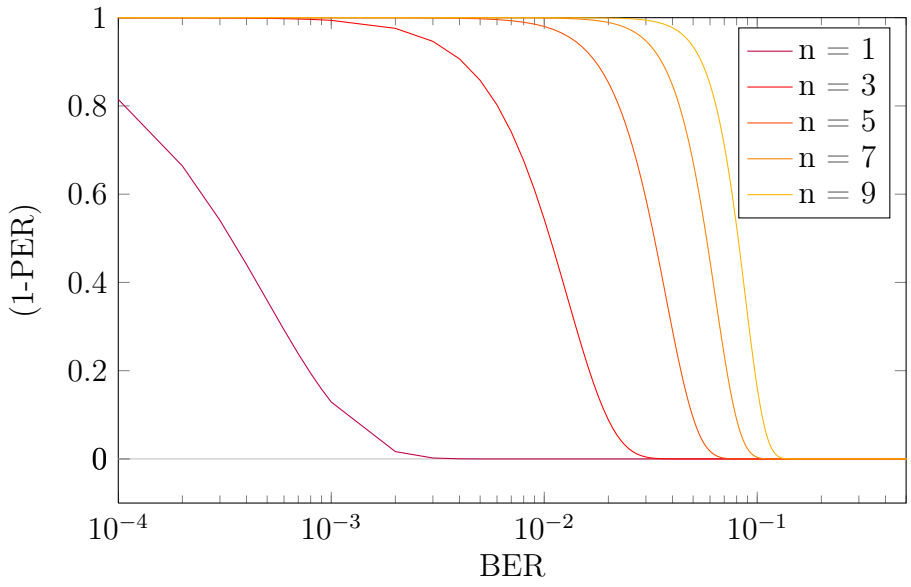


Figure 4.16: PER for a frame length of 256 byte for different sizes n of ground networks

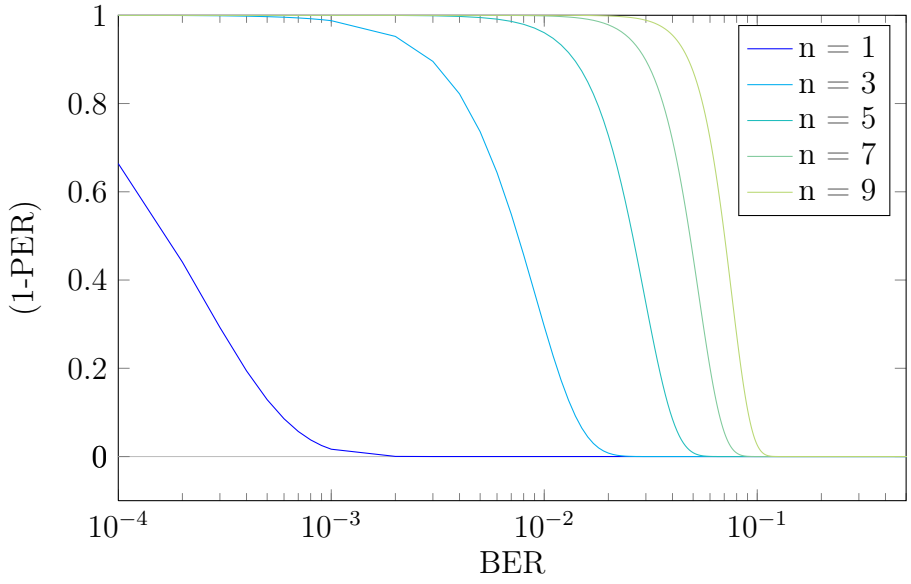


Figure 4.17: PER for a frame length of 512 byte for different sizes n of ground networks

ground station network dramatically. The originally experienced PER on a single ground station drops slowly over a large BER range, i.e. the quality of the link is already strongly affected at low BER values of 10^{-4}. The GSMV algorithm overcomes this problem by keeping the PER relatively constant until a certain BER threshold is reached, after that point the PER drops very fast to 0 within a relatively short BER range. Thus, the link quality is only affected when large BER values arise. The spacing between the graphs of larger values of n indicate furthermore, that it is not possible to sustain BER values larger than 10^{-1}. Nevertheless, the estimated performance of the GSMV algorithm is very promising, the PER is constant also in relatively small ground networks, even for BER values close to 10^{-2}. Of course, this statistical performance evaluation should be rather seen as the performance limit of the GSMV approach, as beside the BER itself also the bit error patterns are important for the performance of the algorithm.

Improvement through single bit error correction (SBEC)

As it is possible to correct with the SBEC algorithm a single error in the retrieved data frames, the performance increase can be calculated relatively simple. The probability of a single bit error in a data frame is

$$PER_{single}(n,p,l) = \binom{l}{1} BER(n,p)(1-BER(n,p))^{l-1} \tag{4.45}$$

The parameter p denotes again the BER of a single ground station, which is needed to calculate the BER of the complete ground station network. The PER of the GSMV algorithm in combination with SBEC is

$$PER_{SBEC}(n,p,l) = PER_{GSMV}(n,p,l) - PER_{single}(n,p,l) \tag{4.46}$$

The result from equation 4.46 is shown in figure 4.18 and 4.19. The dashed graphs describe the PER performance of the GSMV algorithm, the solid graphs of corresponding color the additional improvement by applying the SBEC algorithm.

Compared to the PER observed for a single ground station, the quality in terms of PER of the reconstructed data frames can be improved by more than a order of magnitude, also if only a small amount of ground stations is involved in the network.

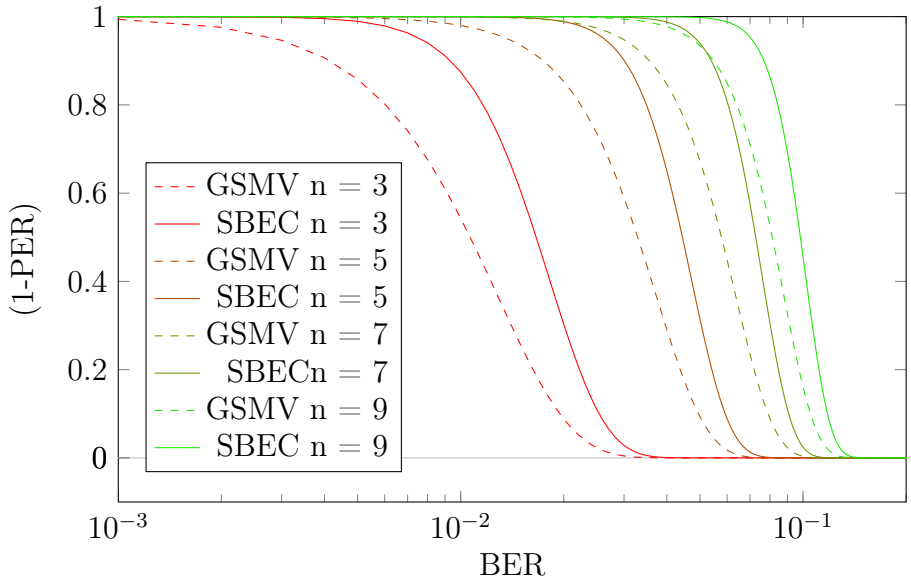


Figure 4.18: PER after applying GSMV and SBEC for a frame length of 256 byte

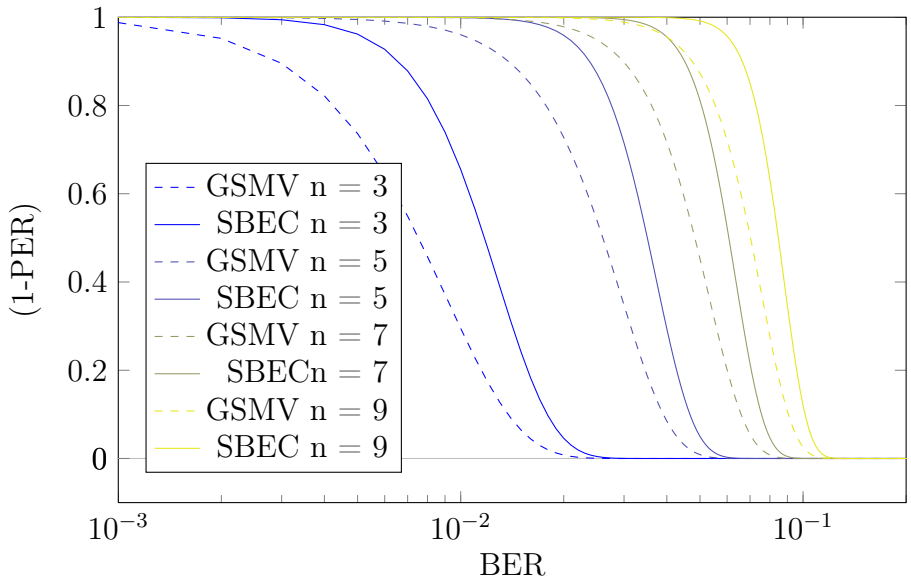


Figure 4.19: PER after applying GSMV and SBEC for a frame length of 512 byte

Furthermore, the algorithm can be implemented with simple means. This promising result advertises the potential for current ground station networks.

4.5 Hardware tests and results

The experiments presented in the following were conducted to show the applicability of the developed data management approaches and to determine performance limits. The proposed data management system handles time synchronization and data synchronization for ground station networks. In the first part of this section, a simulated ground station network on a distributed computer system was used to evaluate the different approaches. A simulated ground station network is more suitable for investigating the performance limits of the implemented system. Determining performance limits in experiments with real satellite links is very difficult, because the communication time is too restricted and valuable to perform statistical expressive experiments with a scientific satellite in orbit. Nevertheless, some aspects of the proposed data management system had to be analyzed with a real radio link to induce realistic delays and transmission errors. Thus, in the second part of this section were experiments performed, which contain real radio links to emulate the parallel reception of data frames. Hence, the approaches were investigated in a different order than presented in the previous sections, due to different experiment setups. The appertaining results are presented in section 4.5.2. The performed hardware-in-the-loop experiments rather demonstrate the applicability and benefits of the proposed data management system in a real ground station network than investigating performance limits.

4.5.1 Ground station network simulation

Experiment setup

To simulate the ground network on a computer network, the implemented software was installed on several PCs at the computer science institute in Würzburg. These computers are connected through a LAN to exchange information. Additionally, further computers outside the institute, connected through the Internet, were used to emulate the typical traffic occurring in IP connections.
A single computer in the network acts as the central server, it is responsible for

collecting, synchronizing and combining data frames. All other computers act as client stations, they receive frames from a virtual satellite and forward the obtained data to the central server. To simulate the satellite, the client stations are sending virtually received AX.25 frames to the central server. Virtually received frame denotes frames, which were requested from the central server to simulate the parallel reception of the frames at the client stations (see figure 4.20).

Moreover, each client station adds a configurable amount of random errors to the

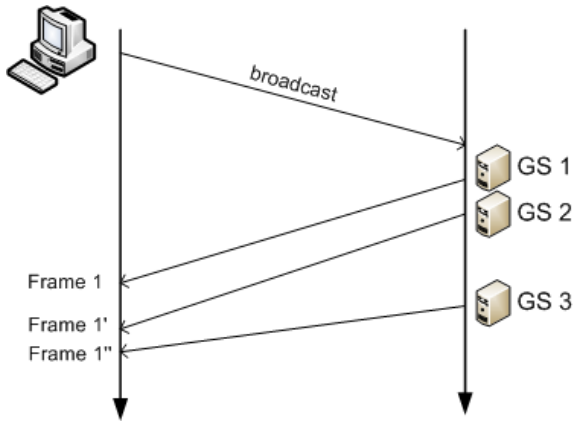


Figure 4.20: Experiment setup

data frames, simulating transmission errors. The central server synchronizes and combines the received data frames with the developed algorithms. Individual experiments for time and data synchronization as well as for data combination are described in the next sections.

Time synchronization evaluation

The aim of this experiment is to compare the offset estimation algorithm of section 4.2.1 with other state of the art synchronization procedures. The offset estimation is necessary to compare the timestamps of the client stations with each other on a global timescale. For comparison, the SNTP protocol [Mills et al., 2005] is used to determine the reference offset between the central server and a client station.

In figure 4.21, the synchronization procedure between the central server and a client station in the same LAN is depicted. The blue intervals represent the minimum and maximum values from the SNTP offset estimation (see section 4.2.1). The red surface depicts the measurements from the proposed synchronization algorithm, which propagates the offset with estimated drift and obtains therefore continuous

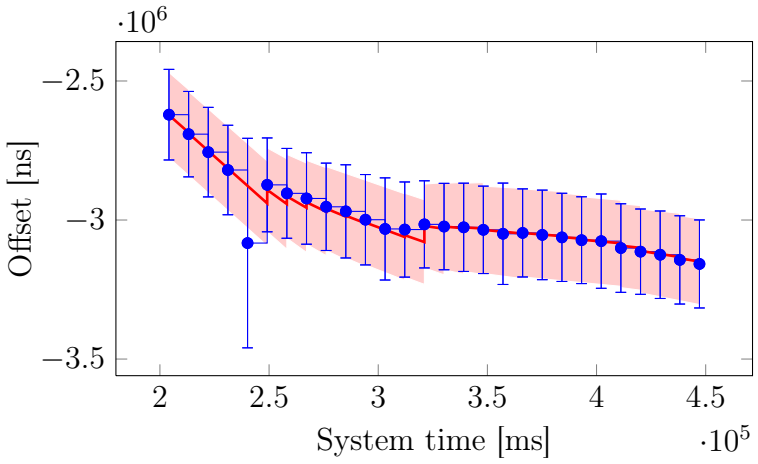


Figure 4.21: Time synchronization performance over LAN

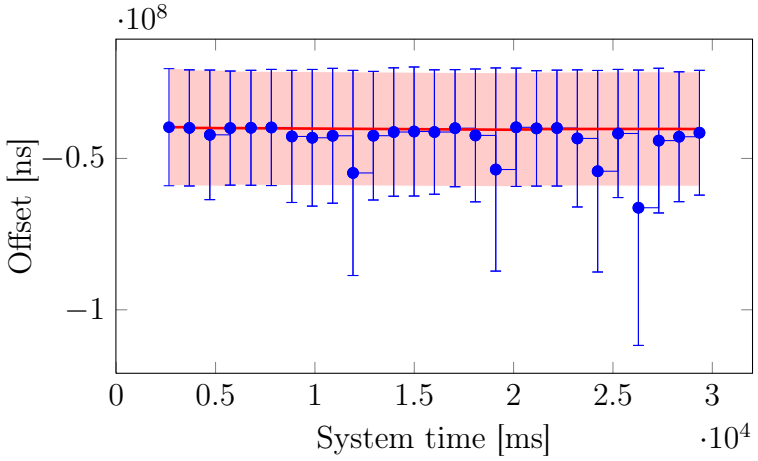


Figure 4.22: Time synchronization performance over Internet

values. The red line depicts the mean values between estimated minimum and maximum offset. From the figure it is easy to see, that the estimated offset from SNTP (blue) is equal or larger than the values obtained with the proposed algorithm (red). The explanation are delay peaks in the network, which increase the estimated time interval of SNTP significantly. The presented algorithm compensates these peaks with the approximated drifts of the computer clocks and is able to keep the estimated offset relatively stable. The same experiment was performed with an Internet connection between the central server and a client station, the result is depicted in figure 4.22. The main difference of this experiment is, that the delay peaks over the Internet are larger than in a local environment (LAN). Hence, SNTP (blue) delivers large varying estimations for the offset.

The main effect of the presented time synchronization algorithm is filtering delay peaks to deliver a better estimation of the offset. If for any reason (e.g. congestion on the Internet link), the data frame arrives delayed at the central server, the offset estimation gets worse with SNTP. By approximation of the clock drift, the presented approach is able to obtain a less variant estimation of the offset. This is an essential point, as the estimation of the offset is used to search for duplicate data frames from other ground stations. Therefore, compensation of the delay peaks is needed to guarantee appropriate search intervals for the data synchronization procedure. Keeping the search intervals at appropriate size is especially important, if the data frame rate of the satellite is high and the corresponding times of reception at the client stations are close together. If the search interval of the data synchronization procedure is too large, data frames will be assigned to the wrong data sets. To avoid this false identification of duplicate frames, the delay peaks can be filtered with the presented data synchronization approach.

The results of these experiments show, that the performance of the presented offset estimation algorithm is comparable with other state of the art techniques for time synchronization (at least in the required order of magnitude of a few milliseconds). The accuracy of the proposed synchronization procedure might not satisfy the high demands of hard realtime requirements, but in the scope of academic ground station networks the obtained offset fully satisfies the desired synchronization of data frames on link layer. To express the accuracy of the presented offset estimation in absolute values is not possible, because the magnitude depends on the observed network latency between the ground stations.

The most important advantage has not been mentioned yet: Using the presented synchronization algorithm does not need external sources for time determination. A heterogeneous network, like current academic ground station networks, face always the problem of restricted system access from the outside. If a NTP server would be required for the proposed system, the additional effort to guarantee a proper NTP synchronization on all participating ground stations is tremendous. In contrast, the proposed system only relies on the local computer clock and needs no other time resources for proper operation. This is an important feature, which will increase the acceptance of that system at other research institutes operating a ground station.

Data combination evaluation

In this section, the data combination capabilities of the proposed system are investigated. The intention of the experiments is twofold: First, the equations for the network performance in section 4.4.1 are to be verified with empirical data. Second, the performance limits of the proposed system for high bit error rates are to be identified.
To identify these performance limits, it is necessary to add transmission errors artificially to observe the system under high bit error conditions. Adding artificial transmission errors has benefits and drawbacks: The benefit is, that the amount of transmission errors can be increased to a desired rate. Furthermore, the statistical evaluation is easier, as it is known which errors were added artificially. The drawback is, that the error patterns of a real radio link to a satellite can not be simulated accurate enough. Many parameters influence the behavior of the radio link, which are partially not known or can not be simulated accurately enough. The experiments in this section were performed with a simulated satellite link with artificial added errors. Supplementary experiments were conducted with a real radio link integrated (c.f. in section 4.5.2). The transmission errors in these experiments are closer to a real mission scenario, however the rate of transmission errors in a real radio link can not be controlled directly.

Error correction capability of GSMV

This experiment was simulated with the setup described on page 134. Each client

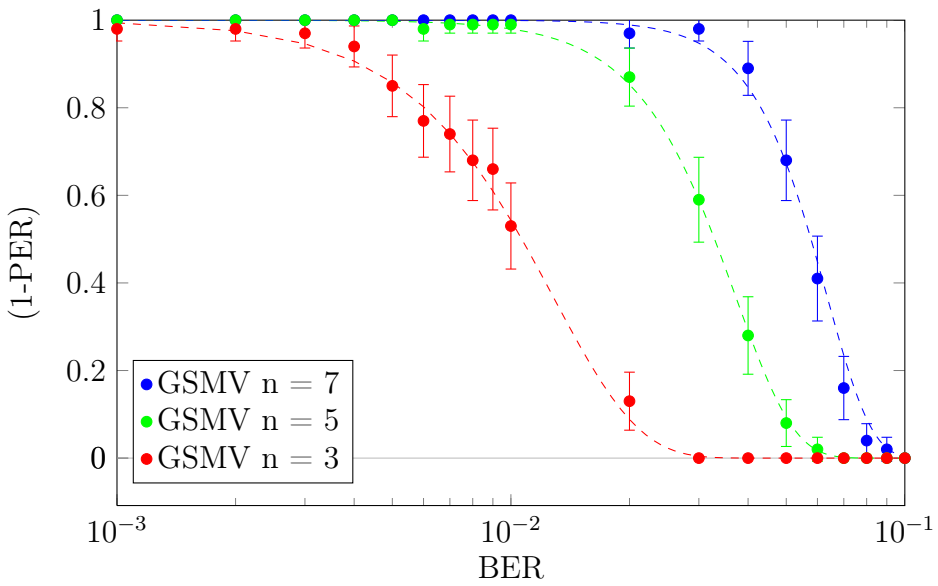


Figure 4.23: Determination of PER in simulated network with n ground stations for a frame size of 256 byte

station adds to the data frames (containing random data payload) artificially transmission errors. A predefined bit error rate of value BER was defined, i.e. each bit of the frame was flipped with a certain probability. The central server collects the corrupted data frames and tries to correct them with the GSMV algorithm. This procedure was performed for each bit error rate m times to determine an average packet error rate PER. The results were obtained with $m = 100$. To calculate the corresponding confidence intervals α was set to 5 % .

The graphs in figure 4.23 and 4.24 show the bit error rate for different frame sizes for the GSMV algorithm. The obtained values are consistent with the equations derived in section 4.4.1 (dashed graphs). Even for large ground station networks, the PER drops fast when the bit error rate approaches 0.5, due to the concept of the GSMV algorithm, which assumes that the majority of the client stations received the bit correct.

The result of this experiment is, that even a small number of ground stations in the network decreases the packet error rate dramatically, when the GSMV approach is applied. The experience from the UWE-1 mission showed [Schmidt and Zeiger, 2006], that the packet error rate varies significantly, even within a single contact window. Hence, it is useful to combine the data streams from different ground stations to overcome high packet error rates. Especially if COTS hardware components are

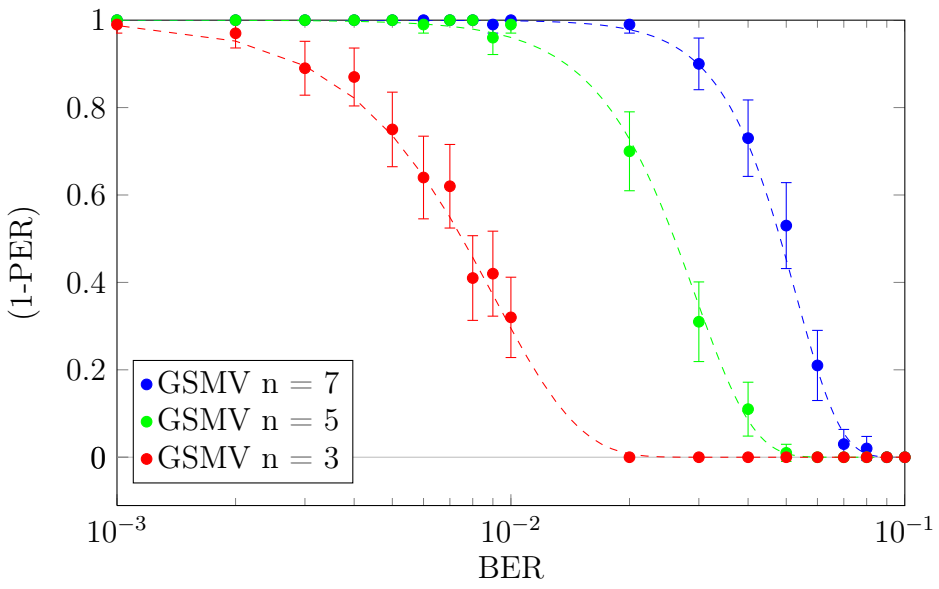


Figure 4.24: Determination of PER in simulated network with n ground stations for a frame size of 512 byte

used, the proposed procedure helps to improve the communication significantly.

Correction rate of SBEC and BF

Similar to the experiments in the previous section, the correction capability was tested for the single bit error correction (SBEC) and Brute Force Correction (BFC) algorithm. SBEC decreases in principle the bit error rate further, but is effected by the "wrong corrections" phenomenon. When more then a single bit error affects the frame, the algorithm might detect a wrong bit position (i.e. a correct bit). The explanation is, that the error patterns are not unique when the number of bit errors is greater than 1. So, only a fraction of the corrected data frames can be corrected reliably.

Of course, the same holds for the BFC algorithm, but here it is possible to check if the obtained solution is unique or if a second solution (i.e. wrong correction) exists. It is trivial that the system can only compensate transmission errors up to a certain bound, so the BFC algorithm is not able to correct a large amount of bit errors due to the restricted error detection capability of the CRC-16-CCITT standard. If too many bit errors affect the data, the BFC algorithm might obtain two different solutions with exactly the same FCS field. Thus it is not always possible to identify

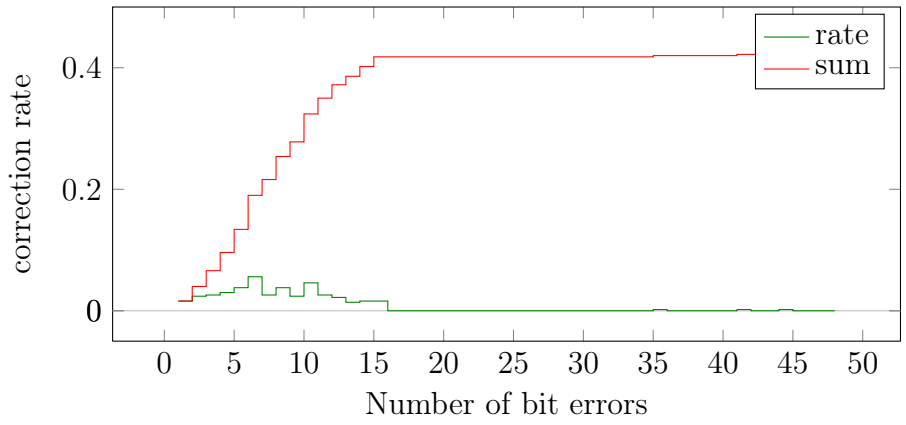


Figure 4.25: Achievable correction rate of the BFC algorithm

which is the correct solution. The following experiment shows which fraction of data frames can be obtained correctly with the BFC algorithm.
A data frame was artificially corrupted with a predefined number of bit errors. The BFC algorithm was used to obtain the original data frame. The fraction of data frames, which the system was able to correct, is expressed in terms of the correction rate.

The graph in figure 4.25 shows on the x-axis the number of bit errors in the data frame, the y-axis describes the average fraction of correct data frames obtained from the BFC algorithm. The green graph depicts the absolute fraction of correctable frames for a given number of bit errors. The red graph contains the correction rate of the summed corrected frames. The sum of the correction rate does not further increase when more than 20 bit errors corrupted the data frame. In these cases the BFC algorithm obtains typically more than one solution for the frame, i.e. a correction is not possible. In combination with the results from the runtime experiments (see next section), it is recommended to restrict the search space of the BFC algorithm to a threshold of 16 bits, because it can be expected that the correction rate increases only marginally and the runtime of the procedure is still in reasonable dimension.
In summary, the overall performance of the BFC algorithm is better than the SBEC with respect to PER improvement. This is obvious, as the BFC algorithm can correct more than single bit errors. Both algorithms are affected by the problem of "wrong corrections", which is especially critical for SBEC, because there is no way to identify the wrong correction. The BFC algorithm faces the same problem,

but is able to go through all possible solutions and identify if the found solution is unique. Hence, the BFC algorithm outperforms the SBEC algorithm in general. Unfortunately, the BFC procedure is much more time consuming, due to the large number of error patterns to be evaluated. Therefore, the runtime was investigated in the next section. For general purpose, the BFC fits better the requirements of academic ground station networks, the SBEC algorithm is only recommended for post processing when BFC is not applicable (for example when only a single frame is available in a data set).

Runtime comparison

The runtime is a very critical parameter for the applicability of the proposed data management system. Performing the GSMV algorithm does not involve calculation intensive steps, but the BFC algorithm is expected to take more processing time due to the exponential number of error patterns involved in the algorithm. Thus, the runtime of the data combination process was investigated in detail.
The runtime of the pure GSMV algorithm is compared to a combination of GSMV and BFC (BF16 and BF32). The main difference between BF16 and BF32 is the number of bit errors the system tries to correct, i.e. BF16 tries to obtain the uncorrupted frame by altering maximum 16 bits, on the other hand BF32 changes up to 32 bit to search for the uncorrupted data frame (the number of error pattern grows exponentially with the number of bits). The SBEC algorithm of section 4.3.3 was not evaluated in this experiment, as the algorithm uses a lookup table for determining the result, which yields negligible runtimes.
In figure 4.26 the total number of bit errors in a combined data set (x-axis) is plotted against the average runtime of the algorithm (y-axis). Bit errors in a combined data set denotes the number of errors in a frame after application of the GSMV algorithm. The plotted runtime is the execution time until the correct data frame was obtained. For the case that no correction was possible, another frame with the same error characteristics was put into the data set and the algorithm was invoked again, until a successful correction was possible. In this way a worst case scenario was created. The reason for the correlation of the increasing runtime and the increasing number of bit errors is, that more data frames were needed to obtain the correct data frame from the data set. More interesting is the runtime difference between the

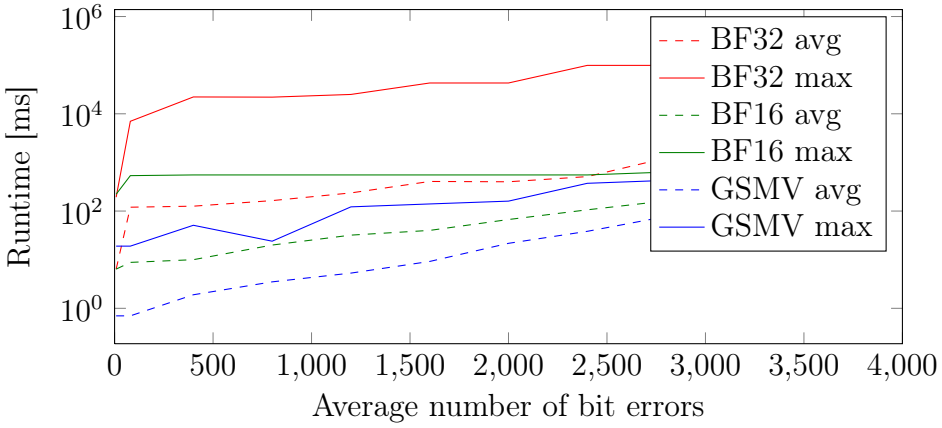


Figure 4.26: Runtime comparison of data combination algorithm

GSMV and the BFC algorithms. This difference is almost constant and is mainly influenced from the execution time of the BFC algorithm. The explanation is, that the BFC algorithm has to evaluate all possible error pattern combinations for each iteration, to ensure if an obtained solution is unique. Therefore, the difference in runtime between the GSMV and the BFC algorithm seems almost constant.
The result from this experiment indicates, that a search space restriction of the BFC algorithm to a maximum number of bit errors is reasonable. The maximum measured values of BF32 exceed 10^4 milliseconds and are problematic for online processing. The maximum values of the BF16 algorithm are unproblematic, due to the parametrization of the AX.25 protocol. Protocol specific timers of the AX.25 protocol are preset to relatively large values, for example the acknowledgment timer is per default 3000 ms [Beech et al., 1997]. Thus, the observed runtime of the BF16 algorithm is small enough to apply the correction procedure to AX.25 connections without modifications. It is recommended with respect to runtime to restrict the search space of the BFC algorithm to 16 bit.

4.5.2 Local ground station network with radio link

The experiments conducted in the previous section were intended to identify limitations of the proposed approaches, the experiments in this section validate the implemented system on a small ground station network with hardware-in-the-loop. In this way, it is possible to observe more realistic transmission errors on the com-

munication link, otherwise transmission errors of a radio link are complicated to simulate.
Additionally to the transmission errors, the characteristic system delays of ground station hardware can not be simulated properly. Hence, the following experiments were performed on typical COTS hardware components used in a state-of-the-art ground station.

Experiment setup

A small local ground station network, consisting of three receiving stations, was established. Each of the receiving stations is connected to the local network. A dedicated server runs the implemented central server software, which collects the data from the receiving client stations. A sending station was used instead of a real satellite, otherwise the experiments would have been too restricted by the visibility of the satellite.
The sending station consists of the same COTS components used in the UWE-1 and UWE-2 satellites for communication. The PR430 transceiver device can be used for communication in 2m and 70cm band and supports the 6pack protocol [Zeiger et al., 2006]. A transmission rate of 1200 bps was used throughout the experiments. The receiving stations utilize an ICOM 910 receiver connected to a desktop computer through the soundcard. For demodulation of the received signals, the soundmodem software from Thomas Sailer [Sailer, 2000] (installed on a linux operating system) is utilized. The implemented client station software retrieves the data frames from the AX.25 network device of the linux kernel and forwards the data, including recorded timestamps, to the central server. The complete experiment setup is shown in figure 4.27 and 4.28.

Data Synchronization Test

The time synchronization between central server and client station was tested already in section 4.5.1 and is also not directly related to the behavior of the radio link, as only the clocks on the local computers are affected from the time synchronization procedure. Therefore, the experiments of this section are related to the delay compensation algorithm and the data frame synchronization.

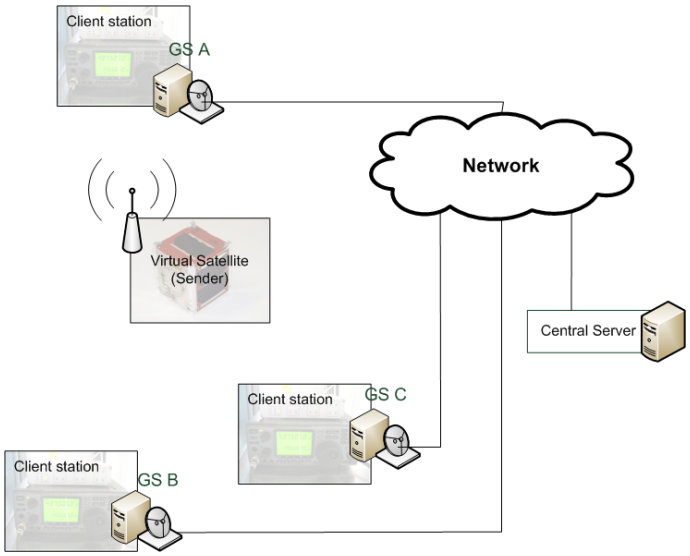


Figure 4.27: Local GS network experiment setup

Figure 4.28: Hardware setup with 3 receiving stations using the ICOM 910 device. Transmitting station with PR430 transceiver

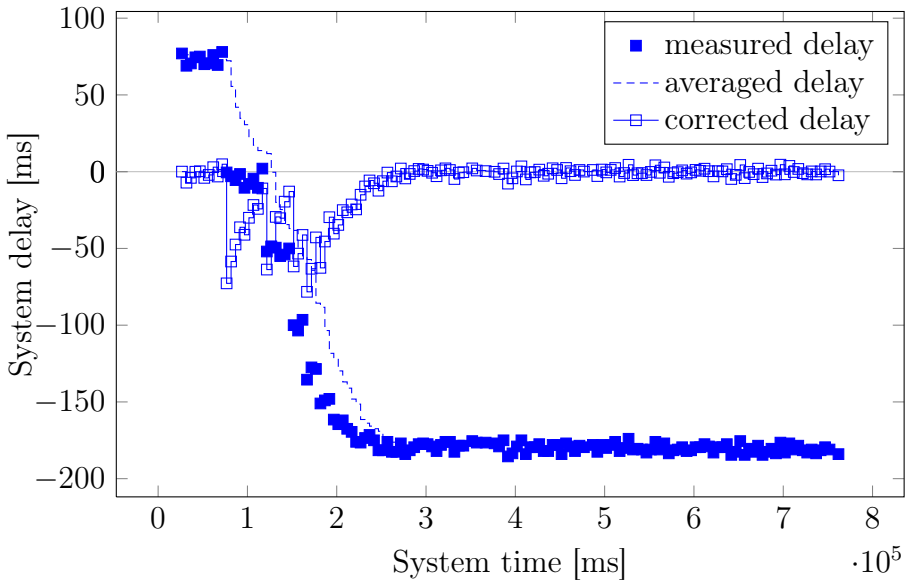


Figure 4.29: System delay correction for a client stations

System delay compensation

To synchronize the data frames, received from the different ground stations, it is necessary to compensate the system delay at each client station (see algorithm in section 4.2.2). The system delay is mainly influenced by the computing speed of the client station and the used radio hardware (transceiver, TNC). Thus, it is necessary to determine the system delay for each individual station to correct the timestamps of the client station properly. In this experiment the system delay S^{gs}_{delay} (see equation 4.24) was investigated for the scenario shown in figure 4.27.

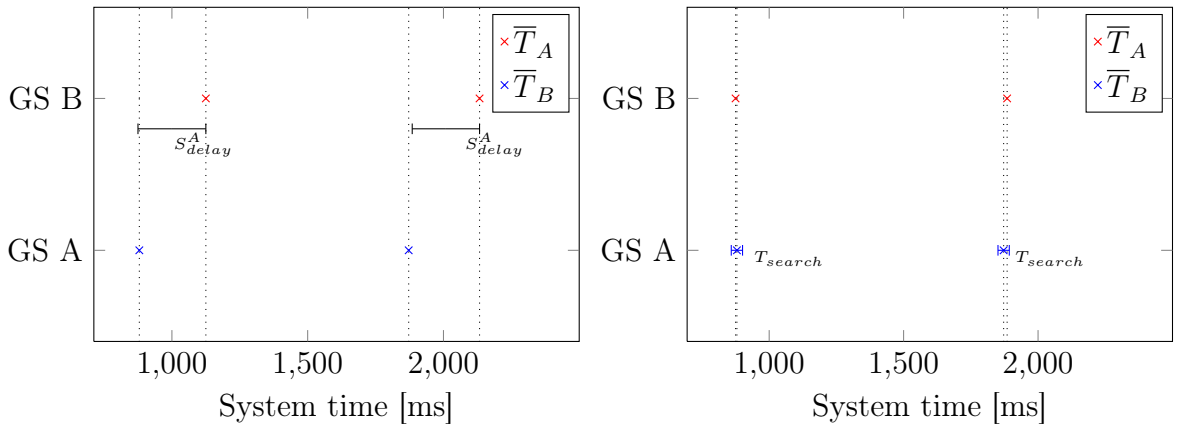
Figure 4.29 shows the system delay in milliseconds for client station A. The measured delay values (blue) describe the time difference between the frame received first from an other client station in comparison to client station A, i.e. the system delay is here always a relative value (the exact definition of the system delay and its calculation are given in equations 4.23 and 4.24). The graph shows, that the system delay stabilizes at a value of about -200 ms, which means client station A submitted the received data frame 200 ms later than the first client station in the network. This experiment was performed within a LAN, therefore it is known that the observed time difference between parallel frames is not caused by the Internet protocol. Client station B observed a stabilized system delay of 380 ms, client station C submitted always the frame first. The client stations in this experiment setup are very

inhomogeneous with respect to processor speed and available memory, which is the main reason for the different system delays. In a real ground network it is expected to have even more inhomogeneous ground station setups. The dotted line in figure 4.29 contains the averaged values calculated from equation 4.24, the corrected values are related to the corrected timestamps, which are used to identify duplicate frames. The objective of the proposed algorithm is to compensate the system delay near to values around 0 ms, only in this case it is possible to identify which data frames belong together. The algorithm was able to correct in all hardware-in-the-loop experiments the system delay to a satisfying value smaller than 5 ms after the initialization phase.

Data frame synchronization

When the time between the client stations and the central server is synchronized appropriately and the system delay at the client stations is compensated, it is now possible to synchronize in the last step the data frames itself. The approach presented in section 4.2.2 is used to identify which data frames belong to each other, i.e. are duplicates. The corrected time of receptions are used to create the search interval T_{search} (c.f. equation 4.26). Only if the offset estimation and the system delay compensation work accurate enough, it is possible to find duplicate data frames within that time interval.
The time of reception of two exemplary frames received from client stations A and C are shown in figure 4.30(a), where the clocks of the client stations were synchronized already, but the system delay was not yet compensated. In figure 4.30(b) also the system delay was corrected. The time of receptions are now close to each other. When the corrected timestamps lie within T_{search}, the frames are assigned to the same data set S_k. This would be the case for both frames in figure 4.30, as their time differences are 5 ms and 13 ms. This example demonstrates the order of magnitude of the delay and the search interval.

For a more extensive analysis, 1000 data frames were transmitted from the sender to the client stations. To evaluate the performance of the data frame synchronization approach, the time difference between parallel frames was determined after applying the synchronization and delay compensation. The frequency distribution of the



(a) System delay determination

(b) Corrected system delay

Figure 4.30: Data frame synchronization

determined time difference is depicted in figure 4.31 for all possible client station combinations ($A \leftrightarrow B$, $A \leftrightarrow C$, $B \leftrightarrow C$). On the x-axis, the absolute time difference between frames is given in milliseconds, the y-axis shows the obtained frequency for a total value of 1000 sent frames.
The average absolute time differences are $1.268ms$, $7.879ms$ and $6.519ms$, i.e the average synchronization accuracy was less then 10 milliseconds. The worst result was observed for the client station A to C, here the amount of the absolute values smaller than 21 milliseconds was 5.9%, which means in the worst case 94.1% of the frames are assigned to the correct data set. The implemented system validates the correct assignment of parallel frames through an ID, which is contained in the data field of the frames. The 5.9% percent of the frames were the absolut difference was larger than 21 milliseconds were mainly contributed in the initialization phase of the experiment, where the system delay was not yet stabilized (compare the initialization phase in figure 4.29). Neglecting the first 50 data frames, which were received during the initialization phase, a much smaller fraction of the frames exceeded the search interval T_{search}. Cutting the intialization phase yields a correct frame assignment rate of 97.5% in the worst case.

An important point, which has to be mentioned is, that the signal propagation delay was not taken into account here, as the sender and the client stations were extremely near located to each other. In a real world scenario, a maximum signal propagation delay of 21 milliseconds could occur. Therefore it is proposed to extend the range value of the search interval T_{search} from 21 ms to 40 ms. This would

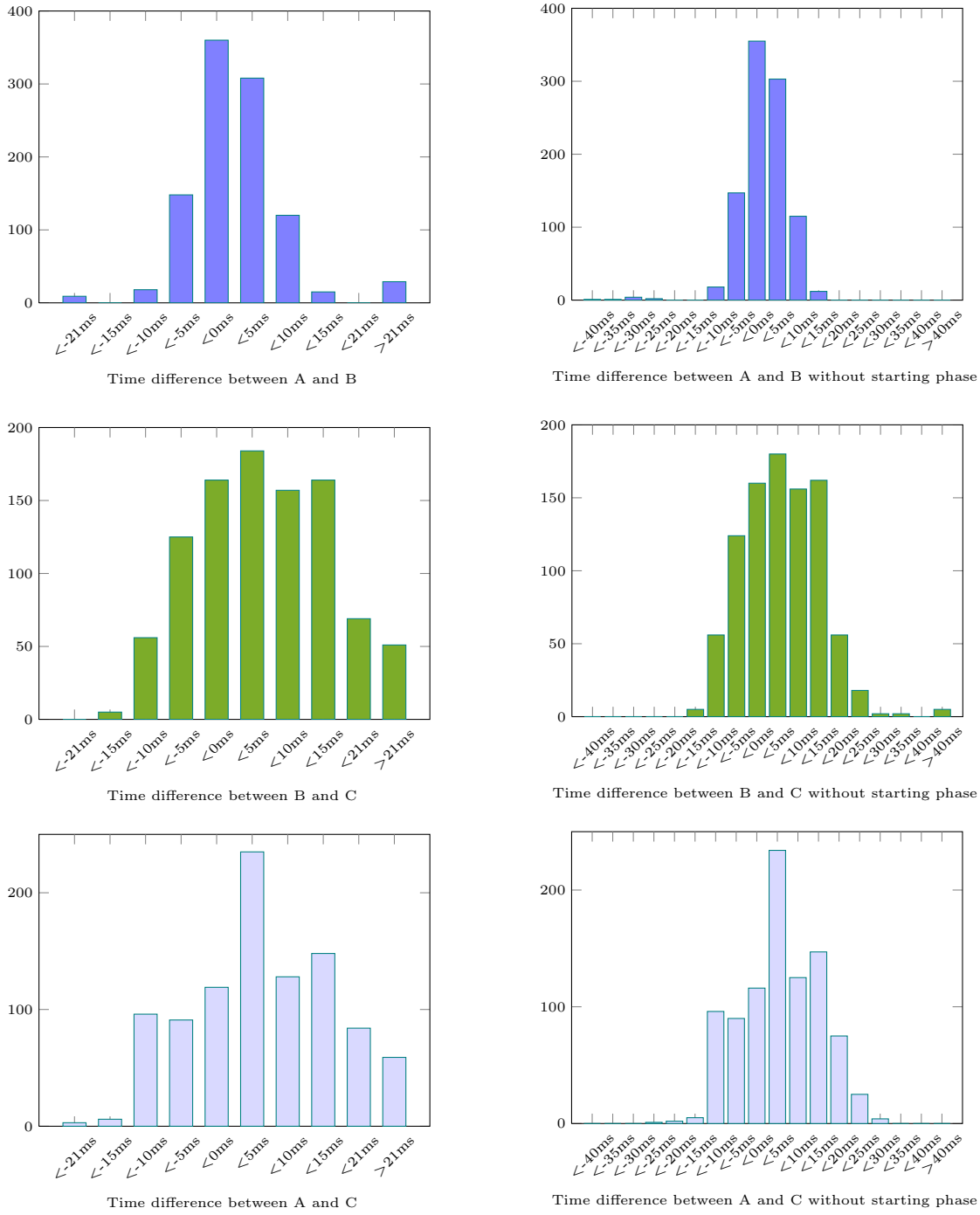


Figure 4.31: Accuracy of data frame synchronization

further increase the rate of correct assignment in this experiment to 99.5%. The range value is a suitable parameter to optimize the performance in future work.
In summary the experiment showed, that the system is capable to synchronize data frames only based on their time of reception. Tuning the search interval T_{search} and the system delay compensation, it is possible to assign 99% of the parallel received frames to the correct data set.

4.6 Conclusion and future work

This chapter introduced a new operation concept for academic ground station networks. The concept originated from the idea to combine a large number of low cost communication nodes, which act together in an efficient way to improve the quality of a single communication link. The implemented and evaluated system uses the redundant communication links available in academic ground networks to detect and correct transmission errors. Even if the research in the field of small satellites focuses on the development of more powerful communication devices, there is still a strong need to better utilize the existing resources. Furthermore, the infrastructure is already existing and can be employed to overcome the communication bottleneck of current small satellite technology.
The developed algorithms for data synchronization and data combination were introduced and analyzed extensively. The data synchronization procedure is divided into the time synchronization and frame synchronization steps. Within this work were offset estimation and system delay compensation algorithms developed, to ensure proper synchronization inside the ground station network. The GSMV, BFC and SBEC strategies were introduced for the data combination problem to detect and correct transmission errors. A data management system adopting all those algorithms was implemented, especially tailored to the communication demands and applicability in small satellite projects. Extensive experiments were performed to evaluate the proposed approach: In simulations the system specific limitations were identified, in hardware-in-the-loop experiments the performance in a real world scenario was validated.
The main contribution of the data synchronization approach is the detection of parallel received data frames, based only on the times of reception. Thus, it is possible to identify from several data streams, received in parallel from a single satellite,

which frames belong together. The system works fully autonomously and processes the data frames online, which makes it applicable to scenarios with ground networks for parallel signal reception and a dedicated single station for uplink. The data synchronization algorithm is theoretically independent of the communication protocol, the implementation was validated with the AX.25 protocol.
Under the context of data combination, a system for transmission error detection and correction was presented. The data synchronization of the proposed approach uses the redundancy contained in parallel received data frames to recover data gaps. The algorithms were derived and analyzed, the implemented system demonstrated how the packet error rate can be decreased in academic ground station networks. Especially for the error-prone communication links of pico-satellites, the introduced concept promises great potential.

Future work in this direction will focus on the optimization of the data synchronization process: The initialization phase of the system delay compensation should be shortened to increase the success rate of the frame synchronization procedure. Additionally, the time synchronization algorithm will be approached in a hybrid solution: When the central server, which is not necessary a ground station, is located far away from the client stations, the delay variations due to distance can influence the time synchronization significantly. In that case the central server might rather synchronize its own clock with a NTP server and the client stations synchronize their clocks themselves with a second NTP time server. This violates the policy of being independent from external sources, but might be the most effective solution when the ground station network is highly distributed.

Chapter 5

Conclusion

The field of small satellite formations and constellations attracted growing attention, based on recent advances in small satellite engineering. An overview on current developments and achievements was presented in chapter 2, properties and requirements of such distributed space system were derived to outline the challenges of this new strategy in space research. The utilization of distributed space systems allows the realization of innovative applications and will enable improved temporal and spatial resolution in observation scenarios. On the other side, this new paradigm imposes a variety of research challenges. This contribution proposes new networking concepts for space missions, using networks of ground stations. The developed approaches combine ground station resources in a coordinated way to achieve more robust and efficient communication links.

Within this thesis, two specific topics were elaborated to improve the performance in distributed space missions: On the one hand scheduling and on the other hand data management, both in the scope of low cost ground station networks. Appropriate scheduling of contact windows in a distributed ground system is a necessary process to avoid low utilization of ground stations. Additionally is an efficient resource utilization only possible if proper scheduling algorithms are applied. Unfortunately, for the special requirements of small satellite missions is no appropriate scheduling system available. This work entered new ground with the development of a tailored scheduling approach introduced in chapter 3, which is capable of satisfying the special needs of small satellite missions. The theoretical basis for the novel concept of redundant scheduling was elaborated in detail, an approach to handle its peculiarities was developed within this thesis. Additionally to the presented algorithm

was a scheduling system implemented, its performance was tested extensively with real world scheduling problems. It was shown, that the implemented scheduling approach fully applies to the needs in current ground station network structures, it creates schedules in reasonable time and is very flexible. Furthermore, interfaces for remote operation are integrated, which facilitate a future integration in available network infrastructure, like GENSO or GSN. Applying the scheduling system increases the contact time with the individual satellites.

In the scope of data management, a system was developed which autonomously synchronizes data frames in ground station networks and uses this information to detect and correct transmission errors. The realization requires the solution of several subproblems: First, the ground stations in the network need to be synchronized with each other to have a common time base. A time synchronization algorithm dedicated to academic ground station networks is presented in section 4.2.1. The second step is the synchronization of the received data frames, a sophisticated algorithm taking into account the orbit geometry and compensating the system delay of each ground station is presented in section 4.2.2. Finally, the system identifies transmission errors in parallel received data frames and applies different strategies for error recovery, which are analyzed in detail in section 4.3. The system was validated with hardware in the loop experiments, demonstrating the benefits of the developed approach, i.e. increasing the robustness and efficiency of small satellite communication links by decreasing the bit error rate.

The developed approaches presented in this work can be applied to single satellites as well as multi satellite systems. The first satellite mission integrating the obtained results in the operation phase will be UWE-3, the third satellite from the University of Würzburg, which is currently developed. But especially future missions, consisting of more than one satellite will benefit from the presented approaches. In particular the scheduling system unfolds its potential when applied to distributed space systems. Upcoming missions like QB50, consisting of a large number of satellites in a single mission, even need advanced coordination strategies to fully adept its advantages.

Looking into the future of satellite applications and space exploration is difficult, but an ongoing miniaturization trend in aerospace engineering is foreseeable. A number of multi satellite systems were launched in the last years, many projects dedicated to distributed space systems are on the way. The presented approaches are only a first

step in the direction of coordination strategies for such missions. To take advantage of the distributed nature and to finally realize intelligent behavior of networks in space, it is necessary to improve techniques and strategies to increase robustness, efficiency and scalability. Many interesting ideas have been brought up already, for example using distributed ground station systems for accurate orbit determination. Other promising fields for further investigation are high precision formation control and fully self configuring communication in satellite networks.

Appendix A

Propagation delay in Low Earth Orbits (LEO)

To synchronize the data frames on geographically distributed ground stations from a single satellite, one has to consider the varying propagation delay affecting the transmission (see figure A.1). The magnitude of the propagation delay depends mainly on the slant range (distance between ground station and satellite). According to Wertz [Wertz and Larson, 1999] the minimum distance can be calculated by

$$D_{min} = R_E \frac{sin(\lambda_{min})}{sind(\eta_{min})} \tag{A.1}$$

the maximum distance is analogue defined as

$$D_{max} = R_E \frac{sin(\lambda_{max})}{sind(\eta_{max})} \tag{A.2}$$

where λ_{min}, λ_{max} describe the minimum and maximum Earth central angle and η_{min}, η_{max} the corresponding nadir angles. The maximum angles are calculated with

$$\lambda_{max} = 90^\circ - \epsilon_{min} - \eta_{max} \tag{A.3}$$

$$sin(\eta_{max}) = sin\rho \cdot cos\epsilon_{min} \tag{A.4}$$

where the Earth angular radius ρ is the only parameter which depends on the orbit height h

$$sin\rho = \frac{R_E}{R_E + h} \tag{A.5}$$

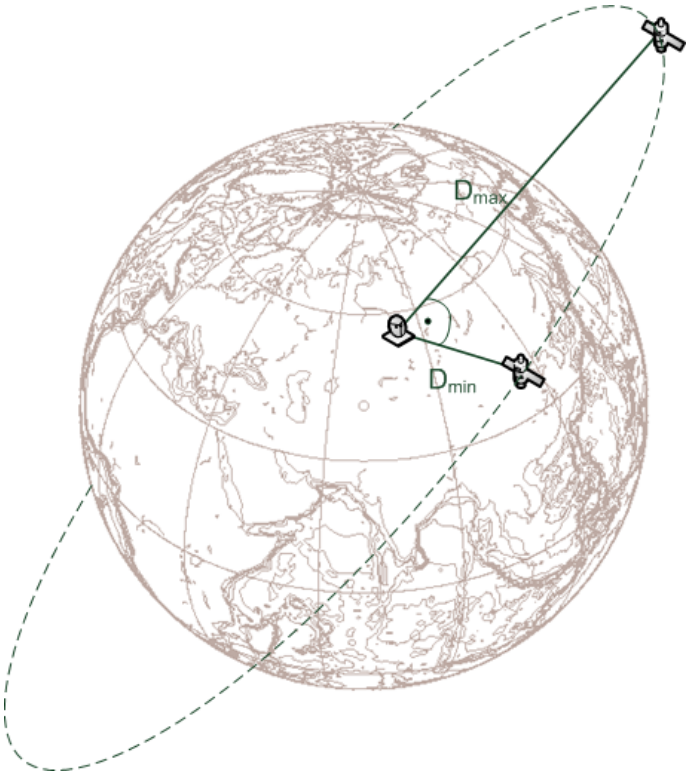


Figure A.1: Minimum and maximum slant range

for the worst case scenario ϵ_{min} is taken as 0, i.e. the satellite has the maximum distance when directly crossing the horizon. The average radius of Earth R_E is 6378 km. How to calculate the minimum distance D_{min} using the parameters λ_{min}, η_{min} is described in detail in [Wertz and Larson, 1999], in our case one can simplify the calculation as the minimum possible distance occurs when the satellite is exactly in zenith of the ground station, i.e. $D_{min} = h$.
Assuming an orbit height between 200 and 1000 km for a LEO orbit [Brown, 1992] and a propagation velocity of $300000\frac{km}{s}$, this results in a maximum propagation delay of 13 ms or a minimum propagation delay 3 ms, respectively. The proposed data management approach in chapter 4 accounts for that by adding a margin of 20 ms to the interval calculated in equation 4.26.

Appendix B

OSI layer model for satellite communication

The OSI reference model (sometimes also called ISO/OSI model) is used to model the communication in computer networks in a layered structure, each layer is responsible for a dedicated service or task. The model contains 7 distinguished layers, not each layer is necessarily implemented in a communication link between two nodes. The layers of the ISO/OSI model are typically governed from communication protocols, providing the required functionalities of the corresponding layer. A protocol can also take over the responsibilities of several or only partial layers, but its communication is restricted to the protocols of the adjacent layers. Extensive documentation about the OSI reference model is available as the X.200 series of recommendations from the ITU-T webpage [1].

The layers of the ISO/OSI model and their corresponding functions are described from bottom up, an extensive description can be found in [Tanenbaum, 2003]:

- Physical layer: Is responsible for transmission of raw bit data on the communication channel. The physical layer deals with mechanical and electrical properties of communication devices.

- Data link layer: This layer guarantees that transmission errors are detected and corrected if possible. Network nodes can exchange information reliably through the data link layer.

[1] http://www.itu.int/rec/T-REC-X/en

- Network layer: Main task of the network layer is to bring the data from the source to the destination, i.e. performs routing in the network.
- Transport Layer: It splits data from the session layer in smaller parts for the network layer. Furthermore, the transport layer provides reliable data transfer to upper layers.
- Session Layer: Brings users on different machines together in sessions. Higher level services like access to restricted resources are provided from the session layer.
- Presentation layer: The presentation layer is the only layer which is related to the syntax and semantics of the transmitted data. For example are different coding standards converted for the application layer.
- Application Layer: Is the closest layer to the end user and provides access to the network.

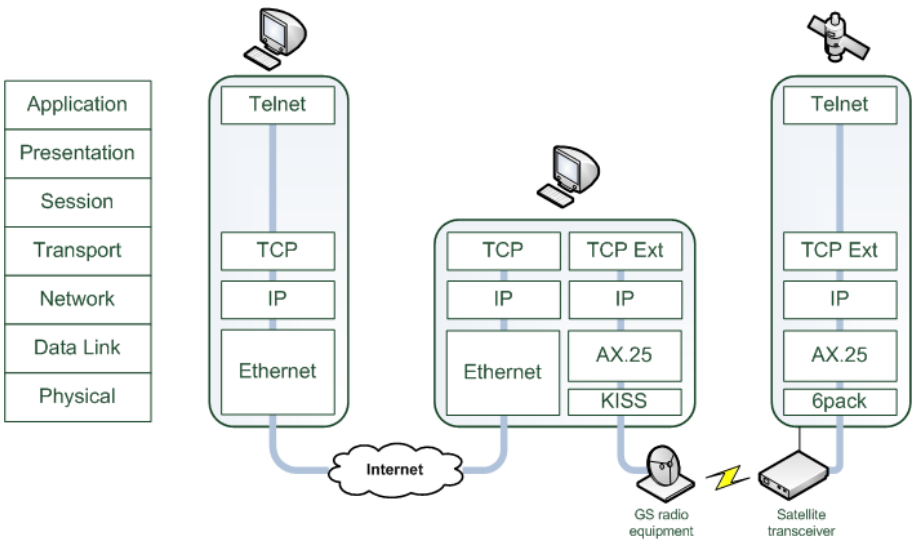


Figure B.1: OSI reference layers in typical small satellite mission

In current small satellite projects are only the first two layers occupied from the AX.25 protocol, which provides the services of the physical and data link layer, supported from hardware specific protocols like KISS or 6pack (see figure B.1). A more sophisticated protocol structure was realized in the UWE-1 satellite, where an

IP protocol suite was stacked on top of AX.25 (for a more detailed description of KISS, 6pack and AX.25 in combination with IP refer to [Schmidt et al., 2007] and [Zeiger et al., 2006]). In that way it is possible to realize an end-to-end communication to satellite over an academic ground station network.
From the distinction in layers originates also the distinction of data packets and data frames. Both carry digital data moving through a network, by definition exists a packet on layer 3 (network layer) of the OSI model, a frame corresponds to data on layer 2 (data link layer) Thus, data exchanged with the AX.25 protocol are denoted as frames, data send over IP are denoted as packets. On the other side is often the term AX.25 packet used, as the AX.25 standard is a transmission mode from the field of packet radio [Beech et al., 1997]. This leads sometimes to confusion, throughout this work the distinction in packet and frames is consistently made.

Appendix C

Satellite orbit data for evaluation of CUSS

The orbit data of the following satellites was used to evaluate the scheduling system of chapter 3. All TLE data sets were obtained from the United States Space Surveillance Network [1].

AAU CUBESAT	1 27846U 03031G 11048.99489955 .00000067 00000-0 50874-4 0 9447 2 27846 098.7002 060.1735 0009186 156.1226 204.0393 14.21082946396099
AAUSAT CUBESAT 2	1 32788U 08021F 11048.63905960 .00000696 00000-0 93504-4 0 9010 2 32788 097.8624 114.8156 0015050 203.4186 156.6295 14.82308357151863
AMSAT ECHO	1 28375U 04025K 11047.47785605 +.00000034 +00000-0 +22586-4 0 08983 2 28375 098.0843 032.3546 0084944 075.7919 285.2698 14.40724001348681
APRIZESAT 1	1 28372U 04025G 11047.55399643 .00000094 00000-0 36124-4 0 8677 2 28372 097.9861 055.6328 0047343 355.0842 004.9876 14.48249515350691
BEESAT	1 35933U 09051C 11048.73803045 .00007830 00000-0 19109-2 0 9026 2 35933 098.3220 149.8764 0004142 213.9289 146.1341 14.52893295 74417
CANX-1	1 27847U 03031H 11052.53293795 +.00000099 +00000-0 +65859-4 0 09622 2 27847 098.6987 063.6383 0009735 147.4830 212.6940 14.21060904396545
CANX-2	1 32790U 08021H 11048.71498687 .00000113 00000-0 20995-4 0 9079 2 32790 097.8629 114.6846 0014979 202.8371 157.2155 14.81794699151828
CANX-6	1 32784U 08021B 11047.65875462 +.00000007 +00000-0 +76484-5 0 09228 2 32784 097.8658 113.2422 0015326 201.6261 158.4312 14.81342051151649

[1] http://www.space-track.org/

CAPE 1	1 31130U 07012P 11047.73124128 +.00000256 +00000-0 +69447-4 0 01077
	2 31130 097.9144 078.6530 0102027 194.7618 165.0591 14.52290982203151
COMPASS 1	1 32787U 08021E 11048.72868898 .00000491 00000-0 68268-4 0 9121
	2 32787 097.8620 114.7154 0015091 202.6807 157.3740 14.82143203151861
CP3	1 31129U 07012N 11049.10626444 .00000107 00000-0 35075-4 0 8001
	2 31129 097.9119 080.1432 0102018 190.2196 169.6964 14.52376371203557
CP4	1 31132U 07012Q 11048.62168840 .00000194 00000-0 52434-4 0 1177
	2 31132 097.9094 086.4917 0085518 170.9161 189.3603 14.55423948203816
CSTB 1	1 31122U 07012F 11049.07043084 .00000558 00000-0 13192-3 0 1450
	2 31122 097.9124 087.2739 0085729 169.5855 190.7132 14.55535839204017
CUBESAT	1 27848U 03031J 11052.53860630 +.00000115 +00000-0 +73897-4 0 09979
XI-IV	2 27848 098.7146 062.0654 0009674 172.5228 187.6093 14.20555601396441
CUBESAT	1 28895U 05043F 11047.65615684 +.00000252 +00000-0 +61818-4 0 08040
XI-V	2 28895 097.9870 288.5849 0016880 236.8685 123.0926 14.60008153282653
CUTE-1	1 27844U 03031E 11047.54239525 -.00000676 +00000-0 -29307-3 0 00480
	2 27844 098.7083 057.9922 0009710 182.9252 177.1880 14.20766987395786
CUTE1.7	1 32785U 08021C 11049.10630057 .00000504 00000-0 70410-4 0 9178
+APD II	2 32785 097.8642 115.2119 0014445 200.4056 159.6568 14.81830235151909
DELFI C3	1 32789U 08021G 11049.15225101 .00001669 00000-0 21352-3 0 9181
	2 32789 097.8708 116.1180 0015009 199.9922 160.0715 14.82605664151946
DTUSAT	1 27842U 03031C 11048.71328243 .00000060 00000-0 47490-4 0 472
	2 27842 098.7004 059.9067 0009527 155.8617 204.2996 14.21085251396059
ITUPSAT 1	1 35935U 09051E 11048.72839354 .00001938 00000-0 48744-3 0 4924
	2 35935 098.3342 149.9509 0009093 152.2340 207.9286 14.52298958 74379
LATINSAT	1 27612U 02058H 11052.59738937 +.00000112 +00000-0 +36299-4 0 04108
A	2 27612 064.5569 253.5289 0046267 319.4573 040.3079 14.76124968440162
LATINSAT	1 27606U 02058B 11047.33644883 +.00000205 +00000-0 +56004-4 0 03774
B	2 27606 064.5604 003.2826 0065171 357.5775 002.5003 14.69498779437714
LIBERTAD	1 31128U 07012M 11048.64426039 -.00000046 00000-0 00000+0 0 1502
1	2 31128 097.9120 079.4611 0102297 192.1605 167.7157 14.52265586203494
MAST	1 31126U 07012K 11047.58295247 +.00000158 +00000-0 +45917-4 0 00870
	2 31126 097.9085 081.5554 0094265 185.5863 174.4285 14.53673993203560
NCUBE-2	1 28897U 05043H 11047.22570521 +.00000213 +00000-0 +53398-4 0 03737
	2 28897 097.9820 288.0575 0016186 239.8688 120.0912 14.60201225274890

QUAKESAT	1 27845U 03031F 11048.70025337 .00000038 00000-0 38054-4 0 440 2 27845 098.7162 058.7191 0008616 197.8438 162.2417 14.20432451395830
SAUDICOM SAT 1	1 28369U 04025D 11048.62098061 .00000169 00000-0 53310-4 0 8761 2 28369 097.9650 066.1545 0033700 327.7477 032.1679 14.50739664351454
SAUDICOM SAT 2	1 28370U 04025E 11048.99540177 .00000108 00000-0 41565-4 0 8866 2 28370 098.0150 049.1306 0059691 015.3896 344.9097 14.45833092350328
SAUDICOM SAT 3	1 31125U 07012J 11047.65827067 +.00000412 +00000-0 +88749-4 0 01827 2 31125 097.9180 102.5006 0046116 125.6647 234.8854 14.62697397204836
SAUDICOM SAT 4	1 31127U 07012L 11052.69934971 .00000382 00000-0 89803-4 0 1752 2 31127 097.9062 096.4111 0070985 139.7687 220.8779 14.58096112204920
SEEDS	1 32791U 08021J 11048.72229905 .00000212 00000-0 33412-4 0 9029 2 32791 097.8641 114.6830 0015499 201.4495 158.6071 14.81883220151816
SWISSCUBE	1 35932U 09051B 11048.72689784 .00005244 00000-0 12986-2 0 8332 2 35932 098.3294 149.6460 0008973 152.0133 208.1384 14.52339170 74373
UNISAT	1 26547U 00057C 11047.91887838 +.00000393 +00000-0 +66429-4 0 07995 2 26547 064.5567 018.0320 0019850 166.6686 193.4953 14.82724635561902
UNISAT 2	1 27608U 02058D 11048.79157460 .00000113 00000-0 37439-4 0 4003 2 27608 064.5598 287.2013 0049333 331.1823 028.6565 14.74594474439421
UNISAT 3	1 28373U 04025H 11049.00621794 .00000139 00000-0 52030-4 0 8679 2 28373 098.0489 042.0539 0071896 042.2348 318.4353 14.43452638350252
UWE-1	1 28892U 05043C 11049.14687096 .00000219 00000-0 54916-4 0 8593 2 28892 097.9801 289.1399 0016385 232.9851 126.9852 14.59987695282984
UWE-2	1 35934U 09051D 11048.72719583 .00003315 00000-0 81574-3 0 7049 2 35934 098.3225 149.6995 0006896 170.1002 189.9896 14.52889913 74414

Bibliography

[Akan, 2002] Akan, O. (2002). Performance of tcp protocols in deep space communication networks. *IEEE Communications Letter*, 6:478–481.

[Alfriend et al., 2010] Alfriend, R., Vadali, S., Gurfil, P., How, J., and Breger, L. (2010). *Spacecraft Formation Flying.* Elsevier.

[Alminde et al., 2003] Alminde, L., Bisgaard, M., Vinther, D., Viscor, T., and Ostergard, K. (2003). Educational value and lessons learned from the aau-cubesat project. In *Recent Advances in Space Technology (RAST 2003), Istanbul.*

[Barbulescu et al., 2002] Barbulescu, L., A. Howe, A., Watson, J., and Whitley, D. (2002). Satellite range scheduling: A comparison of genetic, heuristic and local search. In *Seventh International Conference on Parallel Problem Solving from Nature 2002*, pages 611–620.

[Barbulescu et al., 2004a] Barbulescu, L., How, A., Whitley, D., and Roberts, M. (2004a). Trading places: How to schedule more in a multi-resource oversubscribed scheduling problem. In *International Conference on Automated Planning and Scheduling 2004*, pages 227–234.

[Barbulescu et al., 2007a] Barbulescu, L., Howe, A., Whitley, D., and Roberts, M. (2007a). Understanding algorithm performance on an oversubscribed scheduling application. In *Conference on Artificial Intelligence 2007 AAAI*, pages 577–615.

[Barbulescu et al., 2007b] Barbulescu, L., Kramer, L., and Smith, S. (2007b). Benchmark problems for oversubscribed scheduling. In *The 17th International Conference on Automated Planning & Scheduling.*

[Barbulescu et al., 2004b] Barbulescu, L., Watson, J., Whitley, D., and Howe, A. (2004b). Scheduling space-ground communication for the air force satellite control network. *Journal of Scheduling*, 7(1):7–34.

[Barbulescu et al., 2004c] Barbulescu, L., Whitley, D., and How, A. (2004c). Leap before you look: An effective strategy in an oversubscribed scheduling problem. In *International Workshop on Scheduling a Scheduling Competition AAAI 2004*, pages 143–148.

[Barza et al., 2006] Barza, R., Aoki, Y., and Schilling, K. (2006). Cubesat uwe-1 - technology tests and in orbit results. In *57th International Astronautical Congress Valencia*, number IAC-06-B5.3.07.

[Becker and Smith, 2000] Becker, M. and Smith, S. (2000). Mixed-initiative resource management: The amc barrel allocator. In *5th Int. Conf. on AI Planning and Scheduling*, pages 31–41.

[Beech et al., 1997] Beech, W., Nielsen, D., and Taylor, J. (1997). Ax.25 link access protocol for amateur packet radio, version 2.2.

[Bensana et al., 1999] Bensana, E., Lemaitre, M., and G., V. (1999). Earth observation satellite management. *Constraints*, 4(3):293–299.

[Bensana et al., 1996] Bensana, E., Verfaillie, G., Agnese, J., Bataille, N., and Blumstein, D. (1996). Exact and inexact methods for the daily managment of an earth observing satellite. In *SpaceOps*, pages 507–514.

[Bester et al., 2003] Bester, M., Lewis, M., Quinn, T., and Rauch-Leiba, J. (2003). Automation of operations and ground systems at u.c. berkeley. In *5th International Symposium on Reducing the Cost of Spacecraft Ground System Operations, Pasadena*.

[Blair, 1974] Blair, B. (1974). *Time and Frequency Theory and Fundamentals*, chapter Time and frequency dissemination: an overview of principles and techniques, pages 233–313. National Bureau of Standards Monograph.

[Blott and N., 1996] Blott, R. and N., W. (1996). The space technology research vehicles: Strv-1a, b, c and d. In *10th annual AIAA/USU Small Satellite Conference, Logan*.

[Bresina, 1996] Bresina, J. (1996). Heuristic-biased stochastic sampling. In *Thirteenth National Conference on Artificial Intelligence*, pages 1260–1266.

[Bresina et al., 1995] Bresina, J., Drummond, M., and Swanson, K. (1995). Expected solution quality. In *IJCAI*, pages 1583–1591.

[Bresina et al., 1996] Bresina, J., Edgington, W., Swanson, K., and Drummond, M. (1996). Operational closed-loop observation scheduling and execution. In *AAAI-96 Workshop on Theories of Planning, Action, and Control*, pages 29–33.

[Bresina et al., 1997] Bresina, J., Morris, R., and Edgington, W. (1997). Optimizing observation scheduling objectives. In *International Workshop for Planning and Scheduling in Space 2007.*

[Brown, 1992] Brown, C. (1992). *Spacecraft Mission Design.* AIAA Education Series.

[Burleigh et al., 2002] Burleigh, S., Cerf, V., Durst, R., Fall, K., Hooke, A., Scott, K., and Weiss, H. (2002). The interplanetary internet: A communications infrastructure for mars exploration. In *Acta Astronautica*, number 53, pages 4–10.

[Burrowbridge, 1999] Burrowbridge, S. (1999). Optimal allocation of satellite network resource. Master's thesis, Virginia Polytechnic Institute and State University.

[Busch and Schilling, 2010] Busch, S. and Schilling, K. (2010). A modular and flexible design for a picosatellite bus. In *1st International SPACE World Conference.*

[Cakaj et al., 2007] Cakaj, S., Keim, W., and Malaric, K. (2007). Communications duration with low earth orbiting satellites. In *IASTED, 4th International Conference on Antennas, Radar and Wave Propagation, Montreal*, pages 85–88.

[CCSDS, 2003a] CCSDS (2003a). *TM Space Data Link Protocol. Blue Book. Issue 1.* Number 132.0-B-1. CCSDS Secretariat.

[CCSDS, 2003b] CCSDS (2003b). *TM Synchronization and Channel Coding. Blue Book. Issue 1.* Number 131.0-B-1. CCSDS Secretariat.

[Chen et al., 1999] Chen, W., Jain, N., and Singh, S. (1999). Anmp: Ad hoc network network management protocol. In *IEEE Journal on Selected Areas in Communications*, volume 17, pages 1506–1531.

[Cormen et al., 2009] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press.

[Curtis et al., 2003] Curtis, A., Rilee, M. L., Clark, P., and Marr, G. (2003). Use of swarm intelligence in spacecraft constellations for the resource exploration of the asteroid belt. In *Third International Workshop on Satellite Constellation and Formation Flying, Pisa*, pages 24–26.

[Cutler, 2003] Cutler, J. (2003). Ground station virtualization. In *The Fifth International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena*.

[Cutler and Bhasin, 2002] Cutler, J. and Bhasin, K. (2002). Applying the lessons of internet services to space systems. In *Proceedings of the IEEE Aerospace Conference, Montana*.

[Cutler et al., 2002] Cutler, J., Lindner, P., and Fox, A. (2002). A federated ground station network. In *SpaceOps 2002, Houston*.

[Dabrowska and Stolarski, 2007] Dabrowska, K. and Stolarski, M. (2007). Ground segment of distributed ground station system. In *EUROCON 2007, Warsaw*.

[Damiani et al., 2006] Damiani, S., Dreihahn, H., Noll, J., Niezette, M., and Calzolari, G. P. (2006). Automated allocation of esa ground station network services. In *International Workshop on Planning and Scheduling for Space,*.

[Damiani et al., 2007] Damiani, S., Dreihahn, H., Noll, J., Niezette, M., and Calzolari, G. P. (2007). A planning and scheduling system to allocate esa ground station network services. In *The International Conference on Automated Planning and Scheduling, Providence, Rhode Island*.

[Damiani et al., 2004] Damiani, S., Verfaillie, G., and Charmeau, M. (2004). Autonomous management of an earth watching satellite. In *5th IFAC Symposium on intelligent autonomous vehicles, Lisbon*.

[Damiani et al., 2005a] Damiani, S., Verfaillie, G., and Charmeau, M. (2005a). An anytime planing approach for the management of an earth watching satellite. In *International Conference on Autonomous Agents*.

[Damiani et al., 2005b] Damiani, S., Verfaillie, G., and Charmeau, M. (2005b). A continuous anytime planning module for an autonomous earth watching satellite. In *The International Conference on Automated Planning and Scheduling, Providence, Monterey*.

[Damiani et al., 2005c] Damiani, S., Verfaillie, G., and Charmeau, M. (2005c). An earth watching satellite constellation: How to manage a team of watching agents with limited communications. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 455 – 462.

[Drummond et al., 1995] Drummond, M., Bresina, J., Edgington, W., Swanson, K., Henry, G., and Drascher, E. (1995). Flexible scheduling of automatic telescopes over the internet. In *Henry & Eaton (eds). Robotic Telescopes: Current Capabilities, Present Developments, and Future Prospects for Automated Astronomy*.

[Drummond et al., 1994] Drummond, M., Bresina, J., and Swanson, K. (1994). Just-in-case scheduling. In *Twelfth National Conference on Artificial Intelligence*, pages 1098–1104.

[Durst et al., 1996] Durst, R. C., Miller, G. J., and Travis, E. J. (1996). Tcp extensions for space communications. In *ACM MobiComm*, pages 15–26.

[Fisher et al., 1999] Fisher, F., Mutz, D., Estlin, T., Paal, L., Law, E., Golshan, N., and Chien, S. (1999). The past, present, and future of ground station automation within the dsn. In *Aerospace Conference*, pages 315–324.

[Fortescue, 2003] Fortescue, P. (2003). *Spacecraft Systems Engineering*. Wiley.

[Funase et al., 2006] Funase, R., Nakamura, Y., Nagai, M., Sako, N., Eishima, T., Enokuchi, A., Miyamura, N., Hatsutori, Y., Yoo, I. Y., Komatsu, M., and Nakasuka, S. (2006). Technology demonstration results on university of tokyo's pico-satellite xi-v. In *25th International Symposium on Space Technology and Science*, number 2006-f-07, pages 767–773.

[Funase et al., 2005] Funase, R., Takei, E., Nakamura, Y., Nagai, M., Enokuchi, A., Yuliang, C., Nakada, K., Nojiri, Y., Sasaki, F., Funane, T., Eishima, T., and Nakasuka, S. (2005). Technology demonstration on university of tokyo's picosatellite 'xi-v' and its effective operations result using ground station network. In *IAC*, number IAC-05-B5.3/B5.5.02.

[Gil Biraud et al., 2009] Gil Biraud, M., Page, H., and Kurahara, N. (2009). Global educational network for satellite operations. In *IAC*.

[Globus et al., 2002] Globus, A., Crawford, J., Lohn, J., and Morris, R. (2002). Scheduling earth observing fleets using evolutionary algorithms: Problem description and approach. In *IWPSS*.

[Globus et al., 2003] Globus, A., Crawford, J., Lohn, J., and Pryor, A. (2003). Scheduling earth observing satellites with evolutionary algorithms. In *Space Mission Challanges for Information Technology*.

[Globus et al., 2004] Globus, A., Crawford, J., Lohn, J., and Pryor, A. (2004). A comparison of techniques for scheduling earth observing satellites. In *16th Conference on the Innovative Applications of Artificial Intelligence*, pages 836–843.

[Gooley, 1993] Gooley, T. (1993). Automating the satellite range scheduling process. Master's thesis, Graduate School of Engineering of Air Force Institute of Technology.

[Graham et al., 1979] Graham, R., Lawler, E., Lenstra, J., and Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326.

[Guinn, 2002] Guinn, J. (2002). Autonomous navigation flight demonstration results for the new millenium program eo-1 mission. In *AIAA/AAS Astrodynamics Specialist Conference*.

[Haeusler and Wiedemann, 2008] Haeusler, M. and Wiedemann, K. (2008). *Handbuch der Raumfahrttechnik*, chapter Bodenstationsnetzwerk, pages 464–483. Number 6.3. Hanser.

[Herbst et al., 2005] Herbst, B., Zeiger, F., Schmidt, M., and K., S. (2005). Uwe-1: A pico-satellite to test telecommunication protocols. In *International Astronautical Congress*.

[Hoffmann and Theis, 2009] Hoffmann, A. and Theis, G. (2009). Improving performance and interoperability of the estrack planning system. In *International Workshop of Planning and Scheduling in Space*.

[Hogie et al., 2005] Hogie, K., Criscuolo, E., and Parise, R. (2005). Using standard internet protocols and applications in space. *Computer Networks*, 47(5):603–650.

[Hoots and Roehrich, 1998] Hoots, F. and Roehrich, R. (1998). Spacetrack report no. 3: Models for propagation of norad element sets. Technical report, NORAD.

[Howe et al., 2000] Howe, A., Whitley, D., Barbulescu, L., and Watson, L. (2000). A study of air force satellite access scheduling. In *World Automation Kongress*.

[Hsiao et al., 2000] Hsiao, F.-B., Liu, H.-P., and Chen, C.-C. (2000). The development of a low-cost amateur microsatellite ground station for space engineering education. *Global Journal of Engineering Education*, 4:83–88.

[Israel et al., 2004] Israel, D., Parise, R., and Hogie, K. (2004). Demonstration of internet technologies for space communication. Technical report, NASA GSFC.

[Jackson et al., 2001] Jackson, C., Smithies, C., and Sweeting, M. (2001). Ip demonstration in orbit via uosat 12 minisatellite. In *Proceedings of the 52nd International Astronautical Congress*.

[Janicik and Wolff, 2003] Janicik, J. and Wolff, J. (2003). The chipsat spacecraft design - significant science on a low budget. In *SPIE*.

[Jason et al., 2000] Jason, S., Ward, J., Curiel, A., Phipps, A., Gomes, L., and Sweeting, M. (2000). Low cost planetary exploration: Surrey lunar minisatellite and interplanetary platform missions. In *Small Satellite Conference*, pages 669–680.

[Jefferson et al., 1996] Jefferson, D., Lichten, S., and Young, L. (1996). A test of precision gps clock synchronization. In *IEEE International Frequency Control Symposium*, pages 1206–1210.

[Karabeyoglu et al., 2005] Karabeyoglu, A., Falconer, T., Cantwell, B., and Stevens, J. (2005). Design of an orbital hybrid rocket vehicle launched from canberra air platform. In *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, number AIAA-2005-4096.

[Karn, 2002] Karn, P. (2002). Proposal for a fec-coded ao-40 telemetry link. In *AMSAT Annual Meeting*, pages 60–73.

[Kayal, 2000] Kayal, H. (2000). An experimental low-cost ground station for the small satellite project bird. *Acta Astronautica*, 46:213–220.

[Kayal et al., 2008] Kayal, H., Baumann, F., and Brieß, K. (2008). Beesat - a pico satellite of tu berlin for the in-orbit verification of miniaturised wheels. In *59th International Astronautical Congress*, number IAC-08-B4.4.B10.

[Klimant and Piotraschke, 2003] Klimant, K. and Piotraschke, R. Schönfeld, D. (2003). *Informations- und Kodierungstheorie*. Teubner B.G. GmbH.

[Klofas et al., 2009] Klofas, B., Anderson, J., and Leveque, K. (2009). A survey of cubesat communication systems. *The AMSAT Jounal*, November/December:23–30.

[Kramer et al., 2007a] Kramer, L., Barbulescu, L., and Smith, S. (2007a). Analyzing basic representation choices in oversubscribed scheduling problems. In *MISTA*.

[Kramer et al., 2007b] Kramer, L., Barbulescu, L., and Smith, S. (2007b). Understanding performance tradeoffs in algorithms for solving oversubscribed scheduling. In *Conference on Artificial Intelligence IAAA*, pages 1019–1024.

[Kramer and Smith, 2003] Kramer, L. and Smith, S. (2003). Maximizing flexibility: A retraction heuristic for oversubscribed schedule problems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1218–1223.

[Kramer and Smith, 2004a] Kramer, L. and Smith, S. (2004a). Task swapping for schedule improvement: A broader analysis. In *ICAPS*, pages 235–243.

[Kramer and Smith, 2004b] Kramer, L. and Smith, S. (2004b). Task swapping: Making space in schedules for space. In *Fourth International Workshop on Planning and Scheduling for Space (IWPSS '04)*.

[Kramer and Smith, 2005] Kramer, L. and Smith, S. (2005). The amc scheduling problem: A description for reproducibility. Technical Report CMU-RI-TR-05-75, Robotics Institute, Carnegie Mellon University.

[Lai and Lu, 2010] Lai, H.-K. and Lu, E.-H. (2010). Hybrid coding gain and hybrid arq based on conception of majority voting. *International Journal of Computer Networks and Communications*, 2(1):98–105.

[Lai et al., 2009] Lai, H.-K., Ma, C.-C., and Lu, E.-H. (2009). Error control scheme of hybrid arq based on majority voting bit by bit. In *3rd International Conference and Workshops on Advances in Information Security and Assurance*, pages 563–569.

[Lamport and Melliar-Smith, 1985] Lamport, L. and Melliar-Smith, P. (1985). Synchronizi.ng clocks in the presence of faults. *Journal of the Ascciation for Computing Machinery*, 32 No.1:52–78.

[Leinwand and Fang, 1993] Leinwand, A. and Fang, K. (1993). *Network management: a practical perspective*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Leitner, 2004] Leitner, J. (2004). Formation flying - the future of remote sensing from space. In *2nd International Symposium on Formation Flying - Missions & Technologies*, pages 621–626.

[Leung, 2004] Leung, J. (2004). *Handbook of Scheduling*. Chapman & Hall/CRC.

[Lin et al., 2005] Lin, W.-C., Liao, D.-Y., Liu, C.-Y., and Lee, Y.-Y. (2005). Daily imaging scheduling of an earth observation satellite. *IEEE Transactions on Systems, Man, and Cypernetics*, 35(2):213–223.

[Lindsay and Kantak, 1980] Lindsay, W. and Kantak, A. (1980). Network synchronization of random signals. *EEE Transactions on Communications,*, COM-28:1260–1266.

[Lombardi et al., 2001] Lombardi, M., Nelson, L., Novick, A., and Zhang, V. (2001). Time and frequency measurements using the global positioning system. *International Journal of Metrology*, Jul-Sep:pp. 2633.

[Maldari and Bobrinskiy, 2008] Maldari, P. and Bobrinskiy, N. (2008). Cost efficient evolution of the esa network in the space era. *Space Operations Communicator*, 5:10–18.

[Marszalek et al., 2011] Marszalek, M., Kurz, O., Drentschew, M., Schmidt, M., and Schilling, K. (2011). Intersatellite links and relative navigation: Pre-conditions for formation flights with pico- and nanosatellites. In *IFAC World Congress*.

[McGuire et al., 2006] McGuire, J., Galysh, I., Doherty, K., Heidt, H., and Neimi, D. (2006). Forward error correction extension to ax.25 link protocol for amateur packet radio. Technical report, Stensat Group LLC.

[Melville, 2009] Melville, N. (2009). Genso project - concept, progress and demonstration. In *AMSAT-UK Colloquium*.

[Menana and Demassey, 2009] Menana, J. and Demassey, S. (2009). Sequencing and counting with the `multicost-regular` constraint. In *6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 5547, pages 178–192.

[Michalewicz and Fogel, 2004] Michalewicz, Z. and Fogel, D. (2004). *How to Solve It: Modern Heuristics*. Springer-Verlag.

[Mills, 1988] Mills, D. (1988). Network time protocol (version 1) specification and implementation. Network Working Group Report.

[Mills, 1991] Mills, D. (1991). Internet time synchronization: the network time protocol. *IEEE Trans. Communications*, COM-39, 10:1482–1493.

[Mills, 1992] Mills, D. (1992). Network time protocol (version 3) specification, implementation and analysis. Network Working Group Report.

[Mills, 1993] Mills, D. (1993). Precision synchronization of computer network clocks. Technical Report Report 93-11-1, Electrical Engineering Department.

[Mills, 1995] Mills, D. (1995). Improved algorithms for synchronizing computer network clocks. *IEEE/ACM Trans. Netw.*, 3(3):245–254.

[Mills, 1996] Mills, D. (1996). The network computer as precision timekeeper. In *Precision Time and Time Interval (PTTI) Applications and Planning Meeting.*

[Mills, 1998] Mills, D. (1998). Network time protocol (version 2) specification and implementation. Network Working Group Report.

[Mills, 2003] Mills, D. (2003). A brief history of ntp time: confessions of an internet timekeeper. *ACM Computer Communications Review*, 33:9–22.

[Mills, 2006] Mills, D. (2006). *Computer Network Time Synchronization: the Network Time Protocol.* CRC Press.

[Mills et al., 2005] Mills, D., Plonka, D., and Montgomery, J. (2005). Simple network time protocol (sntp) version 4 for ipv4, ipv6 and osi. Network Working Group Report.

[Minelli et al., 2008] Minelli, G., Kitts, C., Ronzano, K., Beasley, C., Rasay, R., Mas, I., Williams, P., Mahacek, P., Shepard, J., Acain, J., Hines, J., Agasid, E., Friedericks, C., Piccini, M., Parra, M., Timucin, L., Henschke, M., Luzzi, E., Mai, N., McIntyre, M., Ricks, R., Squires, D., Storment, C., Tucker, J., Yost, B. andDefouw, G., and Ricco, A. (2008). Extended life flight results from the genesat-1 biological microsatellite mission. In *22nd Annual AIAA/USU Conference on Small Satellites*, number SSC-08-II-4.

[Montenbruck et al., 2006] Montenbruck, O., Gill, E., and Markgraf, M. (2006). Phoenix-xns - a miniature real-time navigation system for leo satellites. In *3rd ESA Workshop on Satellite Navigation User Equipment Technologies.*

[Montenegro, 2008] Montenegro, S. (2008). *Handbuch der Raumfahrttechnik*, chapter Datenmanagement, pages 356–387. Hanser.

[Moon et al., 1998] Moon, S., Skelly, P., and Towsley, D. (1998). Estimation and removal of clock skew from network delay measurements. Technical report, University of Massachusetts.

[Murdoch, 2006] Murdoch, S. (2006). Hot or not: revealing hidden services by their clock skew. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 27–36.

[Muylaert, 2009] Muylaert, J. (2009). An international network of 50 double cubesats for multi-point, in-situ, long-duration measurements in the lower thermosphere (90 - 320 km) and for re-entry research. QB50 Workshop.

[Nakamura and Nakasuka, 2006] Nakamura, Y. and Nakasuka, S. (2006). Ground station networks to improve operations efficiency of small satellites and its operation scheduling method. In *IAC*.

[Nguyen, 2007] Nguyen, M. (2007). University cubesat projects sector overview. Online Report.

[Nugent et al., 2008] Nugent, R., Munakata, R., Chin, A., Coelho, R., and Puig-Suari (2008). The cubesat: The picosatellite standard for research and education. In *AIAA SPACE Conference*, number AIAA-2008-7736.

[Oda and Nishikawa, 2007] Oda, J. and Nishikawa, K. (2007). Results of experiment 2 on cubesat operation in irv kiruna. Technical report, University of Tokyo.

[Oda et al., 2008] Oda, Y., Komatsu, M., Kurahara, N., Nakamura, Y., and Sakamoto, Y. (2008). Improvement in university satellite operation using ground station network. In *26th International Symposium on Space Technology and Science*, number TU31-TU34.

[Page et al., 2010] Page, H., Biraud, M., Beavis, P., and Aguado, F. (2010). Genso: A report on the early operational phase. In *International Astronautical Congress*, number IAC-10.B4.3.8.

[Page et al., 2008] Page, H., Preindl, B., and Nikolaidis, V. (2008). Genso: The global educational network for satellite operations. In *IAC*, number B4.3.

[Pan et al., 2007] Pan, Y., Ge, N., and Dong, Z. (2007). Crc look-up table optimization for single-bit error correction. *Tsinghua Science and Technology*, 12:620–623.

[Pemberton and Galiber, 2000] Pemberton, J. and Galiber, F. (2000). A constrained-based approach to satellite scheduling. In *DIMACS workshop on on Constraint programming and large scale discrete optimization*, pages 101–114.

[Pinciroli et al., 2008] Pinciroli, C., Birattari, M., Tuci, E., Dorigo, M., Zapatero, M., Vinko, T., and Izzo, D. (2008). Self-organizing and scalable shape formation for a swarm of pico satellites. In *NASA/ESA Conference on Adaptive Hardware and Systems*, pages 57–61.

[Postel, 1983] Postel, J. (1983). Daytime protocol. DARPA Network Working Group Report.

[Preindl et al., 2010] Preindl, B., Seidl, M., Mehnen, L., Krinninger, S., Stuglik, S., and Machnicki, D. (2010). A performance comparison of different satellite range scheduling algorithms for global ground station networks. In *International Astronautical Congress*, number IAC-10.B4.7.1.

[Puig-Suari et al., 2001a] Puig-Suari, J., Turner, C., and Ahlgren, W. (2001a). Development of the standard cubesat deployer and a cubesat class picosatellite. In *Aerospace Conference*, pages 1/347–1/353.

[Puig-Suari et al., 2001b] Puig-Suari, J., Turner, C., and Twiggs, R. (2001b). Cubesat: The development and launch support infrastructure for eighteen different satellite customers one one launch. In *SSC*.

[Radice et al., 2010] Radice, G., Yang, T., and Zhang, W. (2010). Robust algorithms for formation flying reconfiguration. In *International Workshop on Satellite Constellations and Formation Flying*.

[Ravandoor et al., 2010] Ravandoor, K., Drentschew, M., Schmidt, M., and Schilling, K. (2010). Orbit and drift analysis for swarms of picosatellites. In *1st International SPACE World Conference*.

[Raymond et al., 2004] Raymond, C., Bristow, J., and Schoeberl, M. (2004). Needs for an intelligent distributed spacecraft infrastructure. In *Nasa Earth Science Technology Workshop*.

[Richharia, 1999] Richharia, R. (1999). *Satellite Communication Systems: Design Principles (Telecommunications)*. McGraw-Hill Inc.

[Rodriguez-Osorio et al., 2008] Rodriguez-Osorio, R. M., Diaz-Miguel Coca, S., and Vedal, F. (2008). Educational ground station based on software defined radio. In *IAC*, number IAC-08-E1.1.12.

[Ruan et al., 2005] Ruan, Q., Tan, Y., He, R., and Chen, Y. (2005). Simulation-based scheduling for photo-reconnaissance satellite. In *Winter Simulation Conference*, pages 2585–2589.

[Sailer, 2000] Sailer, T. (2000). Soundmodem on modern operating systems. Technical report, BayCom.

[Sandau, 2009] Sandau, R. (2009). Distributed satellite systems for earth observation and surveillance. In *RTO Lecture Series*, volume RTO-EN-SCI-209.

[Schalck, 1993] Schalck, M. (1993). Automating satellite range scheduling. Master's thesis, Graduate School of Engineering of the Air Force Institute of Technology.

[Scharf et al., 2003] Scharf, D., Hadaegh, F., and Ploen, S. (2003). A survey of spacecraft formation flying guidance. In *Amer. Contr. Conf.*, pages 2976–2985.

[Scharf et al., 2004] Scharf, D., Hadaegh, F., and Ploen, S. (2004). A survey of spacecraft formation flying guidance and control (part ii): Control. In *American Control Conf.*

[Schilling, 2009a] Schilling, K. (2009a). Earth observation by distributed networks of small satellites. In *ICICI-BME.*

[Schilling, 2009b] Schilling, K. (2009b). Mission analysis for low-earth-observation missions with spacecraft formations. NATO Lecture Series SCI-209. Small Satellite Formations For Distributed Suveillance: System Design and Optimal Control Considerations.

[Schilling, 2009c] Schilling, K. (2009c). Networked distributed pico-satellite systems for earth observation an telecommunication applications. In *IFAC Workshop Aerospace Guidance, Navigation and Flight Control Systems, Samara.*

[Schilling and Brieß, 2008] Schilling, K. and Brieß, K. (2008). Analyse der anwendungsfelder und des nutzungspotentials von pico- und nanosatelliten. Endbericht, FKZ: 50RU0701.

[Schilling et al., 2006] Schilling, K., Schmidt, M., and Barza, R. (2006). In orbit experiences from the picosatellite uwe-1. In *Small Satellite Systems and Services.*

[Schmidt, 2006] Schmidt, M. (2006). Entwurf und durchfuehrung von experimenten zur charakterisierung der kommunikationsverbindung von uwe-1 mit der anpassung und optimierung der benoetigten protokolle. In *Deutscher Luft- und Raumfahrtkongress, Braunschweig.*

[Schmidt et al., 2010] Schmidt, M., Bolvansky, J., and Schilling, K. (2010). Satellite operation improvement through efficient data combination in ground station networks. In *IFAC Symposium on Automatic Control in Aerospace, Nara.*

[Schmidt et al., 2009] Schmidt, M., Ravandoor, K., Kurz, O., Busch, S., and Schilling, K. (2009). Attitude determination for the nano-satellite uwe-2. In *Space Technolog*, number 28, pages 67–74.

[Schmidt et al., 2008] Schmidt, M., Rybysc, M., and Schilling, K. (2008). A scheduling algorithm for ground station networks related to small satellites. In *SpaceOps, Heidelberg.*

[Schmidt and Schiling, 2009] Schmidt, M. and Schiling, K. (2009). Small satellites for educational purposes. In *60th International Astronautical Congress*, pages 81–87.

[Schmidt and Schilling, 2008a] Schmidt, M. and Schilling, K. (2008a). An extensible on-board data handling software platform for pico satellites. *Acta Astronautica*, 63, Issues 11-12:1299–1304.

[Schmidt and Schilling, 2008b] Schmidt, M. and Schilling, K. (2008b). Internet-based ground station networks for pico-satellites. In *SpaceOps, Heidelberg.*

[Schmidt and Schilling, 2010a] Schmidt, M. and Schilling, K. (2010a). Formation flying techniques for pico-satellites. In *6th International Workshop on Satellite Constellation and Formation Flying, Taipeh.*

[Schmidt and Schilling, 2010b] Schmidt, M. and Schilling, K. (2010b). Ground station majority voting for communication improvement in ground station networks. In *SpaceOps, Huntsville.*

[Schmidt et al., 2007] Schmidt, M., Shankar, R., and Schiling, K. (2007). The picosatellite uwe-1 and ip based telecommunication experiments. In *Automatic Control in Aerospace IFAC, Toulouse.*

[Schmidt and Zeiger, 2006] Schmidt, M. and Zeiger, F. (2006). Design and implementation of in orbit experiments for the pico satellite uwe-1. In *International Astronautical Congress*, number IAC-06-E2.1.07. 2006.

[Schnurr et al., 2002] Schnurr, R., Wesdock, J., Criscuolo, E., Hogie, K., and Parise, R. (2002). Hdlc link framing for future space missions. In *SpaceOps 2002 Conference*, pages 1–11.

[Scholz et al., 2009] Scholz, A., Miau, J.-J., Dachwald, B., Plescher, E., Ley, W., and Juang, J.-C. (2009). Flight results of the compass-1 picosatellite mission. In *60th International Astronautical Congress*, number IAC-09-B4.3.10, pages 1289–1298.

[Serrano et al., 2009] Serrano, M. A., Garcia Matatotos, M. A., and Engdahl, M. (2009). Achieving the ers-2 - envisat intersatellite inter-satellite interfermetry tandem constellation. In *21st International Symposium on Space Flight Dynamics*.

[Shirville and Klofas, 2007] Shirville, G. and Klofas, B. (2007). Genso: A global ground station network. In *AMSAT Symposium*.

[Shukla and Bergmann, 2004] Shukla, S. and Bergmann, N. (2004). Single bit error correction implementation in crc-16 on fpga. In *In International Conference on Field Programmable Technolog*, pages 319–322.

[Smith et al., 2000] Smith, D., Frank, J., and Jonsson, A. (2000). Bridging the gap between planning and scheduling. *The Knowledge Engineering Review*, 15:47–83.

[Somayajulu, 1980] Somayajulu, Y. (1980). Time and frequency dissemination using satellite transmissions. In *Indian natn. Sci. Acad.*, pages 268–274.

[Stallings, 1993] Stallings, W. (1993). *SNMP, SNMPv2, and CMIP. The practical guide to network management standards*. Addison-Wesley.

[Steele, 2004] Steele, B. (2004). Internet protocol (ip) in space. Technical report, University of Maryland University College.

[Stolarski, 2009] Stolarski, M. (2009). Distributed ground station system experimental theory confirmation. In *The European Ground System Architecture Workshop (ESAW)*, pages 1–12.

[Stolarski and Winiecki, 2006] Stolarski, M. and Winiecki, W. (2006). Building distributed ground station system with radio amateurs. In *Space Technology Workshop (MIKON 2006)*, pages 49–54.

[Tanenbaum, 2003] Tanenbaum, A. S. (2003). *Computer Networks*. Pearson, Upper Saddle River, NJ, 4. edition.

[Tu et al., 2000] Tu, K., Pen, H., and Liao, C. (2000). Clock synchronization using gps,glonass carrier phase. In *32nd Annual Precise Time and Time Interval Meeting*, pages 171–179.

[Tuli et al., 2006] Tuli, T., Orr, N., and Zee, R. (2006). Low cost ground station design for nanosatellite missions. In *AMSAT-NA Space Symposium*.

[Turner, 2008] Turner, P. (2008). *Handbuch der Raumfahrttechnik*, chapter Kommunikation, pages 372–387. Hanser.

[Twiggs, 2002] Twiggs, R. (2002). The next generation of innovative space engineers: University students are now getting a taste of space experience building, launching and operating their own space experiments with low-casts picosatellites. In *5th ESA International Conference on Spacecraft Guidance, Navigation and Control Systems*, pages 409–422.

[UNISEC-GSN, 2006] UNISEC-GSN (2006). *GS Management Service Manual*. University of Tokyo.

[Unsal, 1993] Unsal, C. (1993). Self-organization in large populations of mobile robots. Master's thesis, Virginia Polytechnic Institute and State University.

[Vallado, 2008] Vallado, D. (2008). Sgp4 orbit determination. In *AIAA/AAS Astrodynamics Specialist Conference*, number AIAA-2008-6670.

[Wang et al., 2007] Wang, J., Jing, N., and Li, J. (2007). A multi-objective imaging scheduling approach for earth observing satellites. In *Genetic And Evolutionary Computation Conference*, pages 2211–2218.

[Wang and Hadaegh, 1996] Wang, P. and Hadaegh, F. (1996). Coordination and control of multiple microspacecraft moving in formation. *Journal of Astronautical Science*, 44(3):315–355.

[Wang and Hadaegh, 2000] Wang, R. and Hadaegh, G. (2000). Self-organizing control of multiple free-flying miniature spacecraft in formation. In *AIAA Guidance, Navigation and Control Conference*, number AIAA-2000-4437.

[Wertz and Larson, 1999] Wertz, J. and Larson, W. (1999). *Space Mission Analysis and Design*. Microcosm Press.

[Whitworth, 2003] Whitworth, G. (2003). *Space Mission Analysis and Design*, chapter Ground System Design and Sizing, pages 621–644. Microcosm Press and Kluwer Academic Publishers.

[Wittmann and Hanowski, 2008] Wittmann, K. and Hanowski, N. (2008). *Handbuch der Raumfahrttechnik*, chapter Raumfahrtmissionen, pages 41–67. Number 1. Hanser.

[Wolfe and Sorensen, 2000] Wolfe, W. and Sorensen, S. (2000). Three scheduling algorithms applied to the earth observing systems domain. *Management Science*, 46:148–166.

[Xiang and Jorgensen, 2005] Xiang, W. and Jorgensen, J. L. (2005). Formation flying: A subject being fast unfolding in space. In *5th IAA Symposium*, number IAA-B5-0309P.

[Zeiger et al., 2008] Zeiger, F., Kraemer, N., and Schiling, K. (2008). Parameter tuning of routing protocols to improve the performance of mobile robot teleopation via wireless ad-hoc networks. In *5th International Conference on Informatics, Automation and Robotics (ICINCO)*, pages 53–60.

[Zeiger et al., 2009a] Zeiger, F., Krämer, N., Sauer, M., and Schilling, K. (2009a). *Mobile Robot Teleoperation via Wireless Multihop Networks - Parameter Tuning of Protocols and Real World Application Scenarios*, pages 139–152. Springer.

[Zeiger et al., 2006] Zeiger, F., Schmidt, M., and Schilling, K. (2006). A flexible extension for pico-satellite communication based on orbit operation results of uwe-1. In *International Astronautical Congress*, number IAC-06-B5.2.05.

[Zeiger et al., 2009b] Zeiger, F., Schmidt, M., and Schilling, K. (2009b). Remote experiments with mobile robot hardware via internet at limited link capacity. *IEEE Transactions on Industrial Electronics*, 56(12):4798–4805.

[Zufferey et al., 2008] Zufferey, N., Amstutz, P., and Giaccari, P. (2008). Graph colouring approaches for a satellite range scheduling problem. *Journal of Scheduling*, 11:263–277.