Technische Universität München
Department of Informatics

Bachelor's Thesis in Informatics

Nanolink: A data link layer protocol for the Cubesat MOVE 2

Nicolas M.E. Appel


Technische Universität München

Department of Informatics

Bachelor's Thesis in Informatics

Nanolink: A data link layer protocol for the Cubesat MOVE 2

Nanolink: Ein Schicht-2-Protokoll für den Cubesat MOVE 2

Author	Nicolas M.E. Appel
Supervisor	Univ.-Prof. Dr.-Ing. Georg Carle
Advisor	Dipl.-Inf. Stephan-A. Posselt
	Dipl.-Ing. Martin Langer
Date	March 16, 2015

Informatik VIII
Chair for Network Architectures and Services

Ich versichere, dass ich diese Bachelor's Thesis selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Garching b. München, March 16, 2015 Nicolas M.E. Appel

Acknowledgments

This thesis is dedicated to the memory of my father, Rainer Appel. He is missed dearly.

I want to use this opportunity to express my gratitude the many people, without whom this thesis would not have been possible.

I must offer my profoundest gratitude to my advisor Dipl.-Inf. Stephan-A. Posselt, for his highly helpful hints, remarks, and guidance throughout this thesis' term. He helped me to identify and solve many problems, and provided me with his immense knowledge about communications and computer networks. His guidance enabled me to learn more about scientific work, refine my writing, and bring this thesis to completion.

Sincere thanks is due to my co-advisor and MOVE 2 project manager, Dipl.-Ing. Martin Langer, for offering me this interesting topic of investigation. He provided me with very valuable advice and resources on astronautics and satellite communications. Thank you for being such a dedicated and outstanding project manager.

I would like to thank Christian Fuchs for his advice, the hours of fruitful discussions, and the substantial feedback on my thesis.

Many thanks go to Dr. Jürgen Letschnik for his professional advice on satellite communications and helping me with the FUNcube-1 experiment.

Furthermore, I would like to thank my fellow students and friends Jonas Jelten and Richard von Seck, for reading my thesis and offering helpful advice.

Finally, I want to express my deep gratitude to my family and my girlfriend Verena for their support throughout the last years.

皆さん 有難うございました。

Abstract

MOVE 2 is a nanosatellite developed according to the Cubesat standard. The associated restrictions of size and weight limit the performance of the utilized communications hardware. This results in a low bandwidth radio link between ground station and satellite. Conventional protocols cannot satisfy the demand for high bandwidth efficiency. A dedicated protocol that defines the data link layer of Cubesat radio links did not exist prior to this work. In this bachelor's thesis, the layer-2 network protocol Nanolink is therefore presented. Major design goals of Nanolink were high reliability and efficiency. To accomplish these goals, a high degree of adaptation of the protocol to the characteristics of the physical channel is required. An experiment was run using FUNcube-1 telemetry data, in order to determine the downlink channel characteristics. Building on the experimental data, a channel model comprising a Rician channel and AWGN channel was created and used to select suitable error correction mechanisms for Nanolink. The resulting hybrid ARQ scheme promises high protocol efficiency and reliability despite weak channel conditions. Efficiency is further increased by incorporating protocol control information into payload frames. Extension header structures ensures that no significant constant overhead is created by this method. The flexible design of Nanolink allows simple integration and deployment on MOVE 2 and other Cubesats.

Zusammenfassung

MOVE 2 ist ein Nanosatellit, der nach dem Cubesat Standard entwickelt wird. Die damit verbundenen Größen- und Gewichtsbeschränkungen limitieren die Leistungsfähigkeit der eingesetzten Kommunikationshardware. Daraus resultiert eine sehr geringe Bandbreite der Funkverbindung zwischen Bodenstation und Satellit. Der Forderung von hoher Bandbreiteneffizienz können herkömmliche Netzwerkprotokolle nicht nachkommen. Ein dediziertes Protokoll, das die Sicherungsschicht der Funkverbindung mit Cubesats definiert, existierte vor dieser Arbeit nicht. In dieser Bachelorarbeit wird daher das Schicht-2-Netzwerkprotokoll Nanolink präsentiert. Hohe Zuverlässigkeit und Effizienz waren die Hauptentwicklungsziele von Nanolink. Um diese Ziele zu erreichen, ist ein hoher Anpassungsgrad des Protokolls an den physikalischen Übertragungskanal erforderlich. Mit Hilfe der Telemetriedaten von FUNcube-1 wurde ein Experiment durchgeführt, um die Eigenschaften des Downlink-Kanals zu bestimmen. Aufbauend auf den Daten des Experiments wurde ein Kanalmodell entwickelt, das aus einem Rice- und AWGR Kanal besteht. Das Modell wurde genutzt um geeignete Mechanismen zur Fehlerkorrektur für Nanolink auszuwählen. Das resultierende hybride ARQ Verfahren verspricht hohe Protokolleffizienz und -zuverlässigkeit, trotz schlechter Kanalbedingungen. Die Effizienz wird zusätzlich erhöht, indem die Steuerinformation des Protokolls in Datenrahmen integriert wird. Extension Header-Strukturen stellen sicher, dass durch diese Methode kein signifikanter Overhead entsteht. Die flexible Konstruktion von Nanolink erlaubt einfachen Einsatz und Integration in MOVE 2 und andere Cubesats.

Contents

Chapter 1

Motivation

The Munich Orbital Verification Experiment 2 (MOVE 2) is an educational and scientific nanosatellite. It is being developed by students of the Scientific Workgroup for Rocketry and Spaceflight (WARR) in cooperation with the Institute of Astronautics (LRT) of the Technische Universität München. MOVE 2 is a CubeSat, a satellite standard with the goal of allowing inexpensive production and operation of satellites. The term CubeSat originates from the cubical form of the basic one unit 10x10x10 cm satellite [9]. The small size and low weight of CubeSats reduces launch and hardware expenses. This allows students and universities to find simpler access and hands-on experience with astronautics, and to deploy scientific experiments more quickly and with less cost.

Due to their smaller scale, CubeSats are even more restricted in terms of power supply and thus, hardware performance, than larger satellites. This impacts on the available resources for the communication systems. Low transmitter power renders high frequency bands unattractive, due to the associated high free-space path losses. Therefore, CubeSats utilize the VHF and UHF radio bands. Due to regulations of the channel spacing, the frequency bandwidth available on these bands is limited, and constraints the maximum achievable data rate. CubeSat transceivers have a typical data rate in the range of 1200 to 9600 bit s^{-1}. Together with the short and infrequent passes that come with low earth orbits, this puts a considerable constraint on the amount of data that can be transmitted to and from the satellite. A higher amount of data transmittable between satellite and ground station enables more complex communication and commandeering of the satellite. This in return means that more sophisticated experiments and missions are possible. Utilizing the transmittable data volume as efficiently as possible is therefore a major concern.

The search for the optimal use of the given resources begins with the reduction of overhead. Overhead are additional data that are required by the individual protocols involved with the delivery. The overhead is generated by encapsulating data units and adding address information as well as information required by services provided by the individual protocols (e.g. error detection or fragmentation). The established TCP/IP or UDP/IP protocol stacks are made to work in networks of a much larger

scale than the simple ground station-satellite configuration of CubeSats. The satellite data link is a point-to-point connection which does not require large address spaces or elaborate routing mechanisms. The overhead required for their operation is therefore not justifiable for this case of application. Another problem is maximum link utilization. For maximum link utilization, each node must send data continuously without any idle time. Idle time results in a loss of valuable transmission time. TCP reacts to packet losses by reducing throughput, which can have a crippling effect in radio links, where transmission errors are more frequent than in tethered networks. An unacceptable property, which renders the TCP useless for CubeSat missions. Nevertheless, reliability is an absolute necessity for the data link, since the data link is the only possibility to interact with the satellite. UDP does not offer any reliability and is therefore unsuitable as well.

The main focus of this thesis lies on the proverbially missing link between ground station and satellite, the data link layer protocol. There is no protocol that accommodates the requirements and restrictions of the MOVE 2 mission. For this reason, a new data link layer protocol named Nanolink is presented in this thesis.

Chapter 2 gives a short overview of relevant technical details of the MOVE 2 mission and the requirements for the protocol design. The data link layer is the first level of abstraction after the physical layer. As such, it must be adjusted to the properties of the physical link. Hence, an analysis of the characteristics and impairments of the radio links is performed in chapter 3. Based on the findings, channel coding schemes are selected in chapter 4 for the use in the MOVE 2 mission. Subsequently in chapter 5, related protocols are discussed and evaluated for the use in the MOVE 2 protocol stack. Chapter 6 illustrates the concept of the Nanolink protocol in detail. Chapter 7 provides a brief evaluation of the performance of Nanolink. Chapter 8 covers future work on the protocol. Finally, the thesis is concluded in chapter 9.

Chapter 2

MOVE 2

This chapter gives more insight on the relevant aspects of the MOVE 2 mission. Section 2.1 illustrates technical details of MOVE 2. Section 2.2 discusses the requirements for the protocol stack.

2.1 Technical Details

This section gives an overview of the technical details of the individual subsystems of MOVE 2 that must be taken into consideration.

2.1.1 Com Systems

The baseline communications concept of MOVE 2 consists of a UHF uplink and VHF downlink. The radio modules will operate on the frequency bands allocated for use by amateur radio satellites. The frequency ranges are 435 MHz to 438 MHz and 145.8 MHz to 146 MHz for UHF and VHF respectively. The frequency bands are divided into channels with a spacing of 12.5 kHz. The signals are commonly modulated with binary phase-shift keying (BPSK) or a form of frequency-shift keying (AFSK, GMSK). Commercial off-the-shelf CubeSat transceivers provide a transmit power in the range of 22 to 34 dBm. The downlink data rates range between 1200, and 9600 bit s^{-1} and the uplink data rate is typically restricted to 1200 bit s^{-1}.

2.1.2 Command and Data handling

The Command and Data Handling (CDH) unit is the core of the satellite. It controls the other subsystems and processes payload data. The CDH subsystem compiles status information of all active components of MOVE 2 and relays this telemetry data to the com systems. The prospective centerpiece of the CDH subsystem is a quad-core ARM Cortex A5 microprocessor with JTAG support. JTAG is an interface for hardware testing and debugging. It is projected to be used for in-orbit debugging of the CDH hardware and firmware upload. The operating system of the CDH system will be Linux.

2.1.3 Orbit

The orbit of MOVE 2 is projected to be highly inclined and at a height of 600 to 800 km. The frequency and duration of satellite passes depends on inclination and orbital height. For a polar orbit with 90° inclination, the ground station in Munich will have line of sight contact with the satellite for approximately seven minutes, three times a day.

2.2 Protocol Requirements

In addition to the requirement that the protocol stack be adjusted to the hardware and orbit of MOVE 2 and compatible with the respective subsystems, there are functional requirements for the protocol stack. An overview is presented in Table 2.1. Satisfying requirements marked with 'must' is mandatory. Requirements marked with 'should' are optional, but desirable.

#	Title	Obligation
1	Reliability	must
2	Bandwidth efficiency	must
3	TC & Data transfer	must
4	Telemetry beacons	must
5	Backup capability	must
6	JTAG access	should
7	High automation	should

Table 2.1: Requirement overview

Reliability—Reliability is the most important requirement for the protocol stack. The protocols must provide a communication link despite possible disturbances or interruptions of the physical link. The integrity of the transmitted data must be ensured, since corrupt or incorrect data may result in incorrect processing or undefined behavior. In addition, the protocols must be able to handle the periodical connection losses that are inherent to LEO satellite links, without loss of data.

Bandwidth efficiency—Bandwidth efficiency is the second most important requirement for the protocol stack, for the previously discussed reasons.

TC & Data transfer—The protocol stack must to provide the capability to send telecommands to the satellite and to transfer data between CDH and ground station.

Telemetry beacons—Telemetry and housekeeping data must be sent by Nanolink, if no connection to the ground station is active. The telemetry must be decodable by ground receivers, such as radio amateurs across the globe.

Backup capability—The protocol stack must also be able to work as backup for a potential secondary com system. This means that in case of a radio module malfunction, it must be possible to redirect traffic to another radio.

JTAG access—Nanolink should provide for access to the CDH JTAG interface, so that debugging of the satellite is possible after launch.

High automation—Lastly, the protocol stack should provide a high degree of automation so that human interaction is not required in the standard case.

Chapter 3

Radio Link

This chapter covers the analysis of the UHF and VHF radio channels. Good knowledge of the channel is crucial for deciding on appropriate error control mechanisms and other protocol parameters such as the size of the sequence number space or frame length.
Section 3.1 gives an approximation of the signal delay. Section 3.2 briefly discusses the various sources of signal attenuation that influence the satellite radio link. In Section 3.3, measurements from a reference satellite will be analyzed using statistical methods. Based on the data gained in the previous section, a channel model will be created and evaluated in section 3.4.

3.1 Signal delay

One of the most important characteristics of communication links is the signal propagation delay. The propagation delay limits the minimum interval between transmission and reception of a request and response. This interval commonly referred to as round-trip time (RTT).
The product of RTT and link data rate, known as bandwidth-delay product, plays an important role in the choice of automatic repeat request (ARQ) methods. The bandwidth-delay product denotes the amount of unacknowledged data on the link when working at full capacity. This data needs to be retained by the sender until the acknowledgment is received, thus requiring buffering space and an appropriate sequence number space. Depending on the ARQ type, it may also be required to buffer the data at the receiver until all missing data are received. For small bandwidth-delay products these concerns are far less severe.
The following is a brief approximation of the upper limit of the signal propagation delay between satellite and ground station. The longest delays are experienced at the begin and end of each pass, when the satellite is just above horizon and the elevation is 0° and the distance to the satellite is maximal (see Figure 3.1). Atmospheric influences like diffraction and refraction can be neglected due to their small impact on the results. For

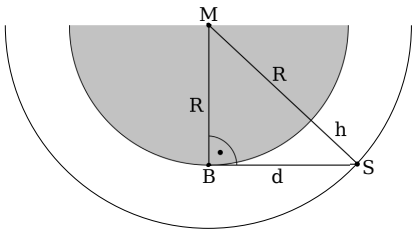


Figure 3.1: Distance of ground station (B) to satellite (S) at 0° elevation

a perfectly spherical earth, the distance to the satellite at 0° elevation is calculated with:

$$d = \sqrt{(R+h)^2 - R^2} \tag{3.1}$$

Where R is the radius of earth and h is the orbital height of the satellite. Since the influences of the atmosphere are neglected, the signal can be assumed to travel with the vacuum speed of light c_0. The propagation delay is then calculated as follows:

$$t_v = \frac{d}{c_0} \tag{3.2}$$

For a circular low earth orbit at 800 km, and an approximate radius of earth of 6367 km, the propagation delay is:

$$t_v = \frac{\sqrt{(6367\,\text{km} + 800\,\text{km})^2 - 6367\,\text{km}^2}}{299\,792\,458\,\text{m}\,\text{s}^{-1}} = 10.979\,\text{ms} \approx 11\,\text{ms}$$

For the maximum data rate of 9600 bit s^{-1}, the maximum bandwidth-delay product is approximately 26 B. For frames with hundreds of bytes, this means that acknowledgments can be received within the serialization time of the next frame. With respect to the bandwidth-delay product, large sequence number spaces and buffers are not required.

3.2 Signal attenuation

Signal attenuation can cause reception errors or transmission losses. To select suitable counter-measures it is necessary to know the sources of attenuation and their characteristics. This section sums up the most important sources of signal deterioration due to attenuation in UHF and VHF radio links based on the publications of Ippolito [22] and the ITU [18–21].

3.2.1 Noise

Like all electrical signals, radio signals are subject to noise that causes reception errors, depending on the so-called signal-to-noise ratio (SNR). The primary source of noise

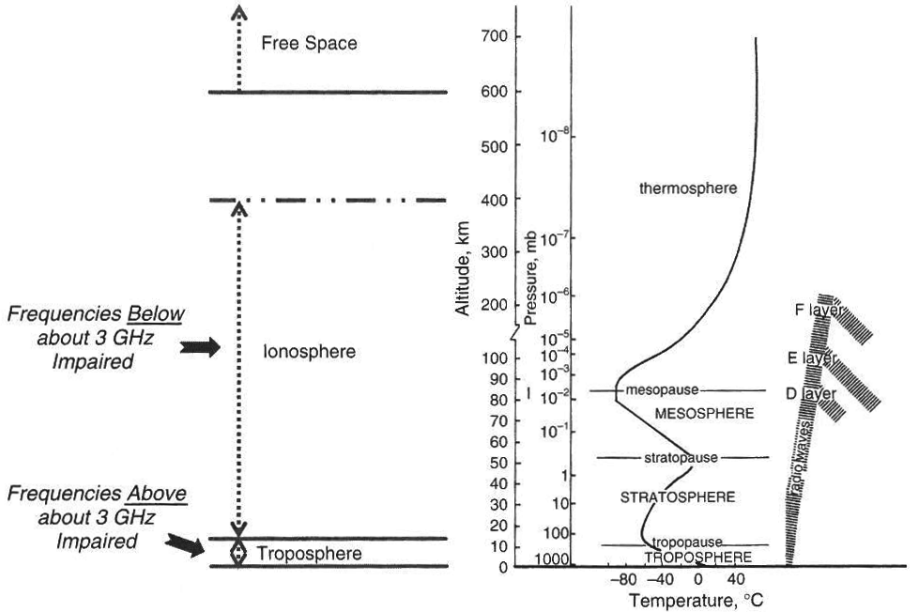


Figure 3.2: Components of the atmosphere impacting space communications [22]

is thermal noise of the receiver hardware, which is modeled as additive white noise [18]. Secondary sources of noise are celestial bodies (sun and moon), earth and cosmic background radiation [22]. The received noise is comparably strong for uplink channels since the satellite's antenna is made to receive signals from the ground and, ipso facto, also receives a greater portion of earth's noise.

3.2.2 Ionospheric effects

The carrier medium also influences wave propagation and signal strength. In the case of satellite communications, this medium is the atmosphere of the Earth. Figure 3.2 illustrates which layers of the atmosphere affect radio transmissions. VHF/UHF signals are impaired by the ionosphere, an atmospheric "region of ionized gas or plasma that extends from about 15 km to a not very well defined upper limit of about 400 km to 2000 km about the earth's surface" [22]. In the ionosphere, the level of ionization is quantified by the *total electron content* (TEC) or electrons per cubic meter. This property fluctuates i.a. with sun activity, day time and time of year.

Scintillation—Smaller, more rapid fluctuations in the TEC result in *scintillation* or *fading* effects, perturbations that can affect the received signal phase, amplitude and refractive index, leading to multipath signal propagation [18,32]. Even in weak cases, scintillations may reduce the signal strength at the receiver by multiple decibels [20]. For the middle latitudes of central Europe, ionospheric scintillation mostly affects VHF signals [20].

Faraday Rotation—Another problem is the rotation of the plane of polarization of a radio wave by the ionosphere and the geomagnetic field, which is usually referred to as *Faraday rotation*. This effect especially affects VHF communication, if a linear polarized antenna is used at the receiver side [22] because the received signal strength diminishes with increasing rotation angle up to the point where the two polarization planes are perpendicular to each other and no reception is possible. The rotation depends on TEC and carrier frequency and reaches its maximum during the day [18].

3.2.3 Other effects

Ground effects—At low elevation angles, other critical effects occur. Obstructions of the line of sight or in the local environment, such as tall buildings, mountains or trees, scatter or diffract the satellite signal. This can result in multipath wave propagation, slow fading or signal cancellation [20–22]. Moreover, CubeSats use unregulated amateur radio frequencies and interference with other radio signals is common. These problems mainly affect the downlink channel and can be assumed to be irrelevant for the uplink.

Satellite rotation—Satellites without attitude control systems may tumble uncontrollably. For linear polarized signals, the plane of polarization is rotated with the satellite's antenna. As with Faraday rotation, this may result in reduced received signal strength. Due to the rotation, the impairment manifests as periodically varying fades.

3.3 Empirical Analysis using FUNcube-1 data

Having discussed some of the most important potential sources of attenuation, it is necessary to find out how the combination of these influences disturbs radio communication. Prior to this work, empirical data about the quality of a UHF/VHF satellite radio links with the LRT antenna system, which will be used for MOVE 2, was not available. Because one goal of this thesis is to find optimal parameters for MOVE 2's COM systems, it is crucial to adjust the systems using real world measurements.
For these reasons, an experiment was conducted to gain further knowledge about the characteristics of the VHF satellite radio link. This section focuses on the description and findings of the experiment.

3.3.1 Experiment Setup

The experimental setup comprises the FUNcube-1 CubeSat and the LRT ground control antennas. The purpose of the experiment is to get empirical data about the quality and properties of a RF link comparable to that of MOVE 2. This data is obtained by receiving telemetry beacons from FUNcube-1, decoding them and extracting the `DecodeErrorCount` and `UTCTime` values. The `DecodeErrorCount` value "represents the number of bit errors detected and corrected by the software when decoding each frame

of telemetry" [12] This quantity is henceforth referred to as *bit errors per block* (BEB). The FUNcube Dashboard source code is not available at this point of time. It is unknown how exactly this property is computed. In this context, it is assumed to be the count of all bit errors of a received encoded telemetry frame.

FUNcube-1 Details—FUNcube-1/AO-73 is a 1-unit CubeSat developed by AMSAT-UK[1]. It currently orbits earth on a 600x685 km polar orbit with an inclination of 97.8° [29]. The satellite's transceiver operates at 146 MHz downlink and transmits BPSK modulated telemetry beacons with a data rate of 1200 bits per second. In sunlight the transceiver power is 300 mW and 30 mW in eclipse. All telemetry beacons from eclipsed passes were disregarded in the further analysis because in this case, the transceiver power is not comparable to that of MOVE 2. Telemetry beacons are transmitted continuously in an interval of 5 seconds. The beacons are encoded as illustrated in Figure 3.3. The FUNcube-1 uses the OSCAR-40 FEC scheme [26, 28]. It is a serial concatenated code (see section 4.1.3) with an outer Reed-Solomon code and an inner convolutional code. The Reed-Solomon code blocks have a length of 160 B and a dimension of 128 B. The output is scrambled with a pseudo-random sequence to ensure a high frequency of bit transitions, needed for demodulator synchronization. Subsequently, the bit sequence is fed into a rate-1/2 convolutional encoder with constraint length 7. The shift register of the encoder is cleared using 6 flush bits. The resulting 5132 bits are then placed into a block interleaver, together with a sync marker of 65 bits and 3 padding bits. The interleaver protects the sync marker and convolutional code from burst errors by introducing temporal diversity. This means that the interval between two consecutive bits of a original sequence, is increased. This is done by reading the input row-by-row into a matrix of 65x80 bit and then reading from the matrix column-by-column. FUNcube-1 is a good reference for the MOVE 2 radio link, as its orbit and transmitter parameters are close to the prospected orbit of MOVE 2 and commercially available communication systems.

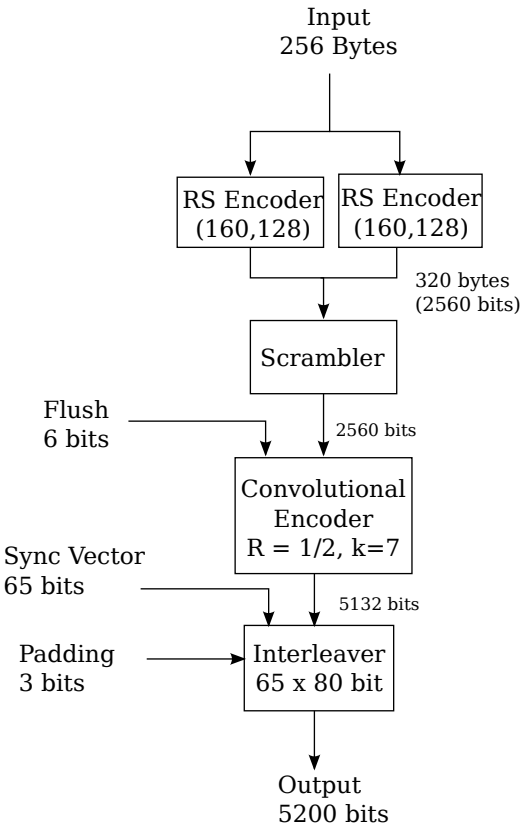


Figure 3.3: FUNcube-1 FEC Encoding [28]

[1] http://www.funcube.org.uk

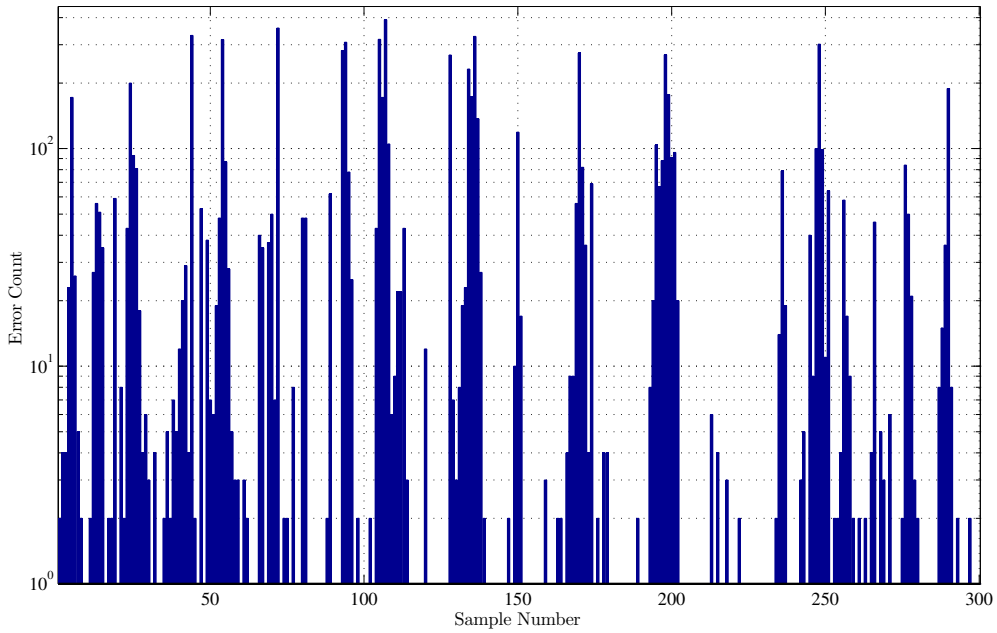


Figure 3.4: Error count of the first 300 samples

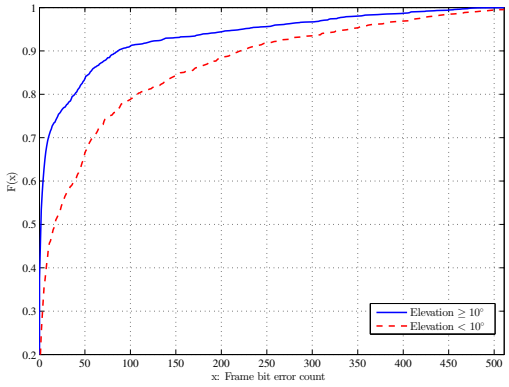
The LRT Ground Station—The ground station antenna system consists of a steerable rack holding two crossed Yagi antennas, located on the roof of the LRT building. The crossed Yagi antenna configuration mitigates the effects of polarization rotations. The antenna system tracks the satellite on its trajectory once it rises above the horizon. The antennas are connected to a computer through a FUNcube SDR Dongle, which was used to decode the satellite signals.

3.3.2 Execution and data preparation

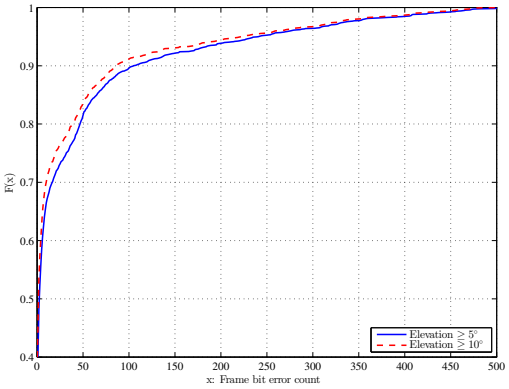
The test began on the 03.07.2014 and during this period, more than 2500 telemetry samples were successfully received. Frames that could not be decoded correctly are unfortunately not logged by the FUNcube decoder software and could therefore not be included in the statistic. The impact of this is assumed to be small, due to the relatively low frequency of frames with very high BEB. A more detailed discussion is given in the following.
Subsequently, every sample was then enriched with the respective elevation and azimuth angles of the satellite and sun, at the moment of reception.

3.3.3 Statistical Analysis

Because the LRT ground control antennas are surrounded by other antennas and buildings and the proximity to the Alps, more severe signal degradation is expected at low angles of elevation, as mentioned in 3.2.3.



(a) ECDF of frames at elevation angles above and below 10°(right)



(b) ECDF of frames at elevation angles ≥ 5° and ≥ 10°

3.3.3.1 Elevation dependent error distribution

In order to determine the impact of these additional attenuations, the data were separated by elevation angle. The influence of ground effects have been reported to be insignificant above 5° [21]. A test was performed to verify this claim and showed that the differences are indeed small, however not insignificant (see Figure 3.5b). Therefore, in this thesis, the upper bound of elevations angles considered "low" is defined to be at 10°.
Figure 3.5a shows that at higher elevation angles, about 37% of the frames are completely error-free. In contrast, only 15% of the frames received at low elevation angles are free of error. Moreover, low elevation transmissions show a dramatic tendency to accumulate a high number of errors: more than 30 percent of the frames have a BEB of 10^{-2} or worse. On the other hand, the graph clearly shows that more than 90% of the frames received at high angles and almost 80% at low elevation angles, could be received correctly if the ability to correct 100 bit errors were added to the frame in form of a forward error correction. Adding more redundancy yields only little benefit, as the curve gradually flattens out with increasing error count. This is an important information for finding an optimal forward error correction method.
As a consequence of the impact of low elevation attenuation, all data below 10° elevation will be excluded from further tests. This reduces the sample size to 1615.

3.3.3.2 Transmission losses

During the experiment, a total of 25 passes was captured. The duration is measured to be the time between the first correctly received frame and the last correctly received frame within the window of a possible pass. By this definition, the average duration was 5 min 56 s. The cumulative duration of all passes was 2 h 28 min 22 s. With a serialization time of 5 s per frame, we could therefore have received:

$$8902\,\text{s} \cdot \frac{1\,\text{frame}}{5\,\text{s}} \approx 1780\,\text{frames}$$

With respect to section 3.3.3.1, this indicates that 165 frames were lost in transmission. There are many possible sources for this, which are not clearly discernible. Knowledge of the source is however desirable because it might indicate flaws of the channel model or software. Possible explanations are:

1. Bugs in the FUNcube Dashboard or FUNcube Dongle
2. Overstrained channel coding
3. Synchronization issues
4. No line of sight to the satellite

To test these hypotheses, the correlation between frequency of transmission losses and other properties were analyzed using Matlab. For the correlation test, the respective final frame before the transmission loss occurred, is used. Since samples with low error count and low elevation angles are more common, the frequencies of the events were normalized before the test.

Property	p	R
Elevation	0.0712	-0.4118
Azimuth	0.9543	0.0137
Daytime	0.6615,	0.2036
Error Count	0.1690	0.3200

Table 3.1: Test results. p expresses the significance and R the strength of the relationship. Negative values of R indicate a reciprocal relationship.

At a level of significance of 0.05, no relationships were statistically significant. However, there is a strong but not significant relationship between elevation and transmission loss. In the light of the fact that there is virtually no correlation between azimuth and frame losses, it is unlikely that the losses can be attributed to a certain LOS obstruction. Overstrained channel coding was an unlikely explanation from the beginning, since the frequency of frames with high BEB is very low. Less than 0.01% of the received frames had more than 480 errors, therefore it is assumed unlikely that the loss of almost 10% of the transmitted frames can attributed to this source. The most likely cause are frame synchronization errors. FUNcube telemetry frames are interleaved with their sync pattern, which is extracted from the bit stream using a statistical approach. A combination of unfavorable events such as symbol slips could shift the position of the sync marker bits in the interleaver, resulting in a lost frame. Subsequently the synchronizer may need a few frames to recover. Unfortunately, this explanation is strictly notional and cannot be verified.

3.3.4 Conclusion

The data from FUNcube-1 has shown that the VHF downlink radio channel exhibits a severe burst error characteristic. The severity and number of bursts was found to increase significantly with decreasing elevation angle. It was found that the ability to correct more than 100 bit errors yields only little benefit at this frame length, since it would require considerable additional expenditures. Burst errors limit the frame length, because the probability of experiencing multiple bursts increases with frame length. Furthermore, the channel coding mechanism must be suitable for burst error channels. Finally, it was found that unknown sources of error cause frame losses. The powerful forward error correction of FUNcube-1 is not sufficient to guarantee the reception of every frame. To ensure reliable delivery, retransmissions are necessary.
For a more detailed analysis, a larger sample set is required. In order to determine the effects of seasonal variations, measurements at different times of the year are necessary.

3.4 Channel models

Channel models are used to simulate real world conditions of a communication channel. The channel model is required so that different aspects of the protocol, especially forward error correction schemes, can be tested. For this reason, the additive white Gaussian noise channel and Rician fading channel are examined and used to create a model in Matlab.

3.4.0.1 Additive White Gaussian Noise (AWGN)

Additive white Gaussian noise is a simple mathematical model for noisy channels. The AWGN models white noise, meaning that the noise power is constant for all frequencies. It is mainly applied to describe the thermal noise at the receiver end [2], but can also model other sources of white noise. Let $n_i \sim \mathcal{N}(0, \sigma^2)$ be a normal distributed random variable and x_i a message symbol, then, due to the additivity of the noise, the received message y_i can be described as:

$$y_i = x_i + n_i \tag{3.3}$$

Therefore, the received symbol is normally distributed: $y_i \sim \mathcal{N}(x_i, \sigma^2)$. The influence of the noise depends on the ratio between signal power P and noise power N_0, where the noise is expressed by the variance $\sigma^2 = \frac{N_0}{2}$ [2]. For digital transmissions the signal power is commonly normalized by the number of bits per symbol, denoted as E_b, to simplify the calculations with different modulations and symbol rates. The normalized signal-to-noise ratio (SNR) is therefore referred to as E_b/N_0 [30]:

$$\frac{E_b}{N_0} = \frac{P}{R \cdot N_0} \tag{3.4}$$

where R denotes the bit rate.

The ambiguity introduced by the noise produces errors when the signal is demodulated. The relative frequency of errors in a noisy channel is referred to as bit error rate (BER). The BER depends on received E_b/N_0 and the number of bits per symbol (modulation order). The bit error rate in an AWGN channel is approximated with (3.5) [2, 22]. Here, erfc is the Gaussian error function.

$$BER = \frac{1}{\mathrm{ld}(M)} \cdot \mathrm{erfc}\left(\sqrt{\frac{\mathrm{ld}(M)E_b}{N_0} \cdot \sin(\frac{\pi}{M})} \right) \tag{3.5}$$

Channel capacity—In noiseless channels, the maximum achievable bit rate depends on the symbol rate and number of bits per symbol. The number of symbols that can be sampled from a signal depends on the available bandwidth. The relation between these factors is expressed by Heartley's law:

$$R_{max} = 2 \cdot B \cdot \mathrm{ld}(M)\,\mathrm{bit} \tag{3.6}$$

where:

R_{max} : Maximum bit rate
B : Bandwidth
M : Modulation order

It follows that in this case, the maximum bit rate is only limited by bandwidth and modulation order. For the sake of an example, assume a standard VHF transmitter, sending BPSK modulated signals with a bandwidth of 4800 Hz. The maximum achievable data rate of this configuration is:

$$R_{max} = 2 \cdot 4800\,\mathrm{Hz} \cdot \mathrm{ld}(2) = 9600\,\mathrm{bit\,s^{-1}}$$

However, in noisy channels, the maximum achievable bit rate is not the only figure of merit, but also the maximum bit rate at which reliable communication is possible. Shannon and Hartley showed that for noisy channels, an upper bound for error free communication, called channel capacity or *Shannon limit*, exists:

$$C = B \cdot ld(1 + \frac{P}{N_0 \cdot B})\,\mathrm{bit\,s^{-1}} \tag{3.7}$$

This implies by increasing the bandwidth indefinitely, the channel capacity approaches a limit [30]:

$$\lim_{B \to \infty} C = \frac{P}{N_0}\mathrm{ld}(e)\,\mathrm{bit\,s^{-1}} \tag{3.8}$$

To increase the channel capacity, it is therefore reasonable to maximize bandwidth utilization by improving SNR and changing to a higher modulation order, so that more bit per sample are transmitted. Proakis *et al.* note that the practical upper bound of the

achievable bit rate is below the *Shannon limit* [30]:

$$R < C \overset{(3.4)}{=} B \cdot \mathrm{ld}\Big(1 + \frac{R}{B} \cdot \frac{E_b}{N_0}\Big)\,\mathrm{bit\,s^{-1}} \tag{3.9}$$

For the sake of this example, we will assume a bandwidth of 4800 Hz, and a SNR of 15 dB. The maximum data rate is:

$$4800\,\mathrm{Hz} \cdot ld(1 + 10^{15/10}) \approx 24\,\mathrm{kbit\,s^{-1}}$$

3.4.0.2 Fading

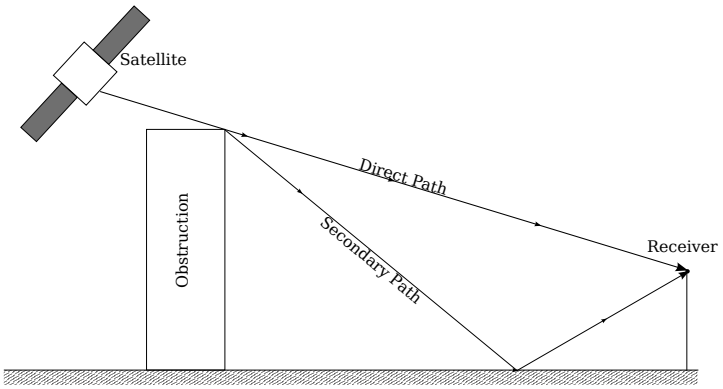


Figure 3.6: Illustration of multipath propagation due to diffraction

Constant noise is not the only source of signal attenuation in VHF/UHF channels. Ionospheric scattering and refraction as well as reflection, diffraction and scattering of radio signals in proximity to the receiver are sources of multipath signal propagation. The signal propagates through multiple paths of different lengths from sender to receiver, the delays along these paths vary. The resulting interference at the receiver causes amplification or cancellation of the received signal [24, 30]. If the relative delays along the respective paths are small compared to the symbol period of the transmitted signal, we speak of frequency flat fading or narrowband fading [14, 22]. As Ippolito notes, satellite channels with small bandwidths and low data rates can be considered to be purely narrowband [22]. FUNcube-1 and also MOVE 2 fall into this category.
The different sources of fading also result in different fading characteristics. Generally, the distinction is made between fast fading and slow fading.

Slow Fading—Ippolito [22] argues that slow fading or shadow fading is caused by obstructions along the propagation path that do not block the signal entirely. Slow fading may cause large scale signal degradation of 6 dB to 10 dB [22], with long influence periods. A major problem with slow fading is that time diversity is not an effective countermeasure.

Fast Fading—Fast fading is the result of multipath propagation and causes more rapid and stronger fluctuations of the received signal power than shadow fading. Fast fades can be reduced by introducing time diversity into the information stream using an interleaver. Multipath fading in telecommunications with line of sight component is generally modeled as Rician fading [24,30]. The Rician channel is a more general version of the Rayleigh fading channel, which models fading without line of sight. The Rician fading model is characterized by the Rice-factor K, which denotes the ratio between the power of the line of sight and other path components of the signal. For links with motion between sender and receiver, the Rician model allows for different Doppler shifts along the paths.

3.4.0.3 Matlab Model

The channel was modeled using Matlab (see Figure 3.7). The channel comprises a source of AWGN and Rician fading. The pseudo-random number generator produces blocks of 5200 bit to correspond with the length of FUNcube-1 frames. The blocks are BPSK modulated, sent through the channel and demodulated. The resulting statistic, the number of bit errors per block, is calculated by comparing received and sent blocks.

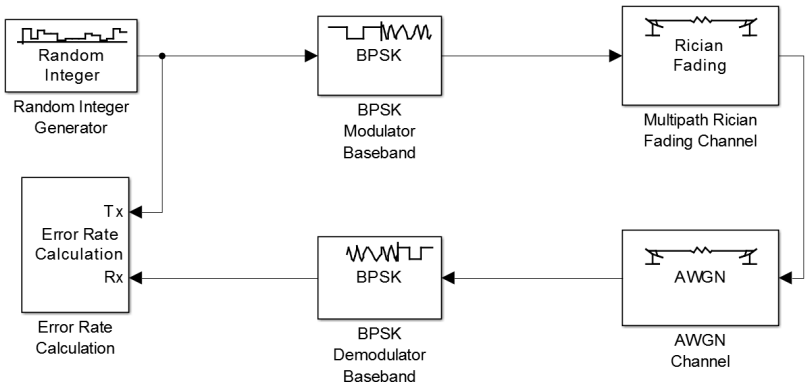


Figure 3.7: Simulink model of the channel

To test the parameters of the model, the cumulative distribution of the simulated BEB samples is compared to the measured samples from FUNcube-1. Ideally, the quality of the parameters should also be evaluated by the time between the individual bursts. Unfortunately, the samples from FUNcube-1 do not allow such detailed considerations, because the BEB counts only measure the cumulative number of errors but not their individual location.

For the flat fading channel, a good fit was found with $E_b/N_0 = 14.7$ dB, $K = 3.7$, a maximum Doppler shift variation of 1.21 Hz along the scattered paths, a Doppler shift of 3600 Hz on the direct path and a sampling rate of 1200 Hz. A E_b/N_0 ratio of 14.7 dB indicates a very high SNR (BER $\approx 10^{-14}$). Figure 3.8 shows the graphs of the CDF of the simulated and measured samples. The figure shows that the model yields results that

correspond closely to the measurements. This implies that the main impairments are due to the modeled sources. The maximum measurable number of errors per frame is capped due to the capacity of the forward error correction. This is why the simulation yields higher maximum BEB than the samples from FUNcube and also a possible explanation for the discrepancy between measured and simulated samples for higher BEB values. There is also a slight discrepancy for BEB values above about 20 and below about 100. Since the model considers only constant AWGN and fades due to multipath propagation, it is possible that this discrepancy stems from other sources.
The model represents a non-varying channel that corresponds to the average-case of the FUNcube-1 passes. Situational or seasonal variations, as well as other sources of impairments are not accounted for. For these reasons, the main knowledge gain of the model is the awareness of the influences of fading on the downlink channel and high receiver SNR. Also, the model can be used to conveniently simulate the downlink channel for FEC testing.

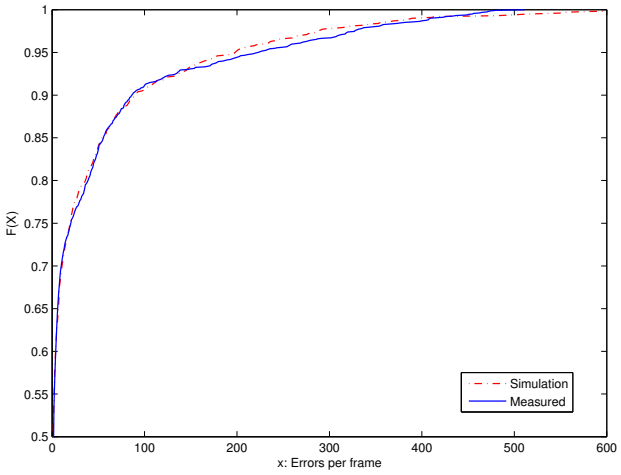


Figure 3.8: ECDF of simulated and measured samples

3.5 Conclusion

The signal propagation delay between ground and LEO satellites was found to be in the range of milliseconds, resulting in a very low delay-bandwidth product. This implies that only small sequence number spaces and buffers are required. Four sources of signal attenuation can be distinguished: Noise, ionospheric scintillations and polarization rotation, and receiver environment. Ionospheric effects and objects in the proximity of the receiver may cause the downlink signal to arrive at the receiver along different paths, which can have adverse effects to the signal quality. The UHF uplink is mostly subject to noise and Faraday rotation.
An experiment with the purpose to get empirical data about the VHF downlink was performed. The downlink channel was found to have strong burst error characteristics

while a major part of the frames were received without error. Also, we showed that the burst characteristic increases for very low elevations. Approximately 9% of the frames transmitted during the passes were lost. A correlation test was found to be inconclusive. It was concluded that the use of forward error correction methods is not sufficient to provide reliable communication.
The bit error rate of an AWGN channel is a function of the normalized signal-to-noise ratio E_b/N_0 . The theoretical limit to the maximum data rate at which reliable communication can be attained, depends on the E_b/N_0 ratio. In the subsequent discussion of narrowband fading, the fast and slow fading effects were distinguished. It was concluded that fast fading is mostly caused by multipath propagation which can be modeled using a Rician channel. A Matlab model comprising an AWGN and Rician channel was developed. The model was found to perform with reasonable accuracy for the average case. The received E_b/N_0 was found to be very high, which indicates good signal quality apart from fading. We thus conclude that the good average-case accuracy is sufficient for comparing forward-error correction schemes.

Chapter 4

Channel Coding

The previous chapter discussed the various sources of transmission errors on the respective channels. This chapter lays focus upon channel coding, a technique that strives to render messages robust to erroneous transmission. Channel coding, or forward error correction, works by adding additional redundancy to a message which can be used to correct defective message parts. Section 4.1 presents an overview of the most commonly used channel coding mechanisms used in satellite communication. Afterwards, the performance and applicability of these codes for the use in MOVE 2 is evaluated in 4.2.1.

4.1 Error Correcting Codes

This section reviews error correcting codes that are most commonly used in astronautics. First, convolutional codes will be illustrated. Afterwards, Reed-Solomon codes will be elaborated. Subsequently, serial-concatenated codes and turbo codes will be covered. Finally, low-density parity-check codes will be discussed.

4.1.1 Convolutional Codes

Convolutional codes are a family of linear error-correcting codes, invented by Peter Elias in 1954 [10] with the goal to find a family of codes with good performance in binary symmetric channels and AWGN channels [11, 14]. The following summarizes the respective sections in the publications of Glavieux [14] and Bossert [3].
Unlike block codes, convolutional codes operate on a continuous sequence of input data, with virtually no limitation on message length. To encode a message sequence d, it is divided into blocks d_k of length K. Subsequently, every block d_k is fed into the encoder f, a linear function that maps d_k, along with m preliminary blocks, to a code block c_k [3]:

$$d = d_0, d_1, d_2, \ldots \rightarrow c = c_0, c_1, c_2, \ldots, \tag{4.1}$$

$$\text{with } c_k = f(d_{k-m}, \ldots, d_k) \text{ and } d_k \in GF(2)^K, c_k \in GF(2)^N \tag{4.2}$$

N denotes the number of code blocks at the output of the encoder so that $R_c = K/N$ is

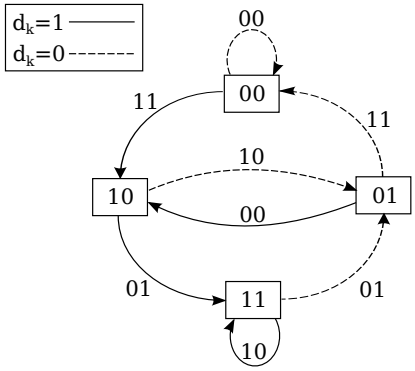


Figure 4.1: State diagram of a convolutional code with $R_c = 1/2$ and $m = 2$

the code rate. The dependency of c_k on d_k and m previous message blocks introduces a memory effect in the encoder. The quantity $\nu = m + 1$ is called constraint length, it represents the number of message blocks that affect a code block. Because the information is more widespread, an increased constraint length generally results in a higher error correcting capability.

Glavieux *et al.* note that "the operation of a convolution encoder may be represented by a graph called a *state transition diagram*" [14]. Such a diagram is found in Figure 4.1 for a convolution encoder with $R_c = 1/2$ and $m = 2$. The state labels represent the state of the memory and the transition labels the output code block. A dashed transition indicates the input of a "1" into the encoder, a solid transition a "0" respectively.

The number of states of a certain encoder is 2^{Km}, resulting in exponential growth of encoder and decoding complexity with constraint length and message block length. K adds no value to the error correcting capability, hence typical convolutional codes used in a practical setting have $K = 1$ and $\nu = 7$.

This in return, sets a limit to the code rate, so that $R_c \in \{1/N | N \in \mathbb{N}\}$. A way to achieve high code rates without increasing decoding complexity is *puncturing* or *perforating* rate $1/N$ codes. Convolutional are perforated by deliberately "punching out" certain code symbols from the encoder output.

As an example, let us return to an arbitrary rate $1/2$ encoder, with two binary symbols d_1, d_2 as input and four binary c_1, c_2, c_3, c_4 symbols as output. Removing one of the four c_k yields an encoder with $K = 2$ and $N = 3$, a rate $2/3$ encoder. Glavieux *et al.* point out that the downside of punctured codes is that the "performances of perforated codes are generally a little lower than those of non-punctured codes of the same rate and of the same constraint length" [14].

The coded sequence at the output of the encoder can also be interpreted as sequence of traversed states of a finite discrete-time Markov process [11]. The state sequence, and thus the message symbols, can be obtained by traversing the Markov chain once more. Because the convolution encoder begins in state `00` and not all states communicate with each other, some code symbol sequences are not possible, e.g. '0010' or

'000001' for the encoder illustrated in Figure 4.1. Receiving an impossible sequence is an indicator for a symbol error. This property is used by the Viterbi Algorithm, a maximum likelihood decoder, to find the most likely state sequence of the encoder. If the correcting capability is exceeded, the Viterbi algorithm produces burst errors [11]. For this reason, convolutional codes are often concatenated with Reed-Solomon codes (see section 4.1.3).

To improve the performance of convolution codes in burst error channels, an interleaver comes into consideration. Interleavers permute the code sequence so that burst errors are dispersed into single bit errors which can be handled by the convolutional code. This is an extremely effective method to reduce the BER and used in Turbocodes (see section 4.1.4). However, large interleavers increase decoding delays which must be taken into account.

4.1.2 Reed-Solomon Codes

Reed-Solomon codes (RS codes) are a class of cyclic error correcting block codes with a wide range of applications, from data storage to satellite communications. Forney *et al.* argue that the popularity of RS codes stems from their "substantial error correction power with relatively small redundancy" [11] in channels with low BER, and good burst error correction performance.

Since there are multiple equal possibilities to encode a message with RS codes [3, 34], the most common, the generator polynomial approach is discussed. A Reed-Solomon polynomial code is defined over finite fields $GF(q = p^m)$. In this approach, code or message symbols are elements of the $GF(q)$. Sequences of symbols are interpreted as coefficient vector of a polynomial. The code words $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ are the coefficients of the code polynomial $c(x)$:

$$c(x) = (c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1}), c_i \in GF(q) \tag{4.3}$$

The degree of polynomials in $GF(q)$ is limited to $\leq n = q - 1$ because for every element $\beta \in GF(q)$: $\beta^n = 1$. A set of code words $\mathcal{C}$ is defined by a polynomial $g(x)$, called generator polynomial, of degree $n - k$. Every code polynomial in $\mathcal{C}$ must be a multiple of the generator polynomial:

$$c(x) = m(x) \cdot g(x) \tag{4.4}$$

The number of different code words in $\mathcal{C}$ therefore is q^k [34], which is why k is usually referred to as *dimension* of the code, whereas n is its *length*.

Reed-Solomon codes may be systematic or non-systematic. To generate a systematic codeword from a message $\mathbf{m} = (m_0, m_1, \ldots, m_k - 1)$, the message symbols must be part of $\mathbf{c}$. To do so, the coefficients of $\mathbf{m}$ are placed into coefficients $c_{n-k}, \ldots, c_{n-1}$ of $\mathbf{c}$. Since $c(x)$ must satisfy (4.4), the remaining $n - k$ coefficients are the residuals of the

polynomial division of $c(x)$ by $g(x)$ (4.5) [3]:

$$(c_{n-1}x^{n-1} + \ldots + c_{n-k}x^{n-k}) : g(x) = m(x) + r(x) \tag{4.5}$$

$$c(x) = c_{n-1}x^{n-1} + \ldots + c_{n-k}x^{n-k} - r(x) \tag{4.6}$$

According to Wicker *et al.* , "*The generator polynomial for a t-error-correcting code must have as roots 2t consecutive powers of* α" [34], where α is the primitive root of the finite field.

$$g(x) = \prod_{j=1}^{2t}(x - \alpha^j) \tag{4.7}$$

It follows that t-error-correcting generator polynomials have degrees from $2t$ to $n-1$ or $n-k$. From this follows that the error correcting capability of a (n,k) RS code is [34]:

$$2t = n - k \Rightarrow t = \left\lfloor \frac{n-k}{2} \right\rfloor \tag{4.8}$$

Reed-Solomon codes are maximum distance separable (MDS) [3, 34]. They have the largest possible minimum distance d for codes of this length and dimension. The minimum distance of a code is the minimum hamming distance of a code and an important metric for the error-correcting capability of a code. For RS codes $d = n - k + 1$ [34]. In an error free channel, they can reconstruct the message of after up to $n - k$ erasures [3] of the code word. RS codes are non-binary codes with symbol sizes greater than 1 bit. RS codes are often chosen so that the length of the code words aligns with the lengths of binary units, e.g. bytes. Codes over GF(256) are therefore frequently used and can be interpreted as sequence of single byte values. Since RS codes operate on symbols rather than single bits, single bit errors contribute as much to the faultiness of a symbol as multiple bit errors. This is the reason why RS codes are good burst-error-correcting codes but renders them also vulnerable to uncorrelated errors.

RS codewords are of fixed length, which might be inconvenient in certain practical settings. Here, systematic codewords can excel, because they can be shortened. Codewords are shortened by padding a message to full length, encoding the message and finally, removing the padding symbols from the codeword. This can be undone easily at the receiver side and is a common way to produce codewords of arbitrary size.

Reed-Solomon codes are furthermore characterized by low encoding and decoding complexity, as table 4.1 shows. However, with the Berlekamp-Massey algorithm, the standard algebraic decoding algorithm for RS codes, the decoding complexity is in $\mathcal{O}(d^2)$ [11], rendering large block sizes and higher distances unattractive for practical purposes. There is a some research on soft-decision decoding, although the "Holy Grail" [34], soft-decision maximum-likelihood decoding, has yet to be found.

Reed-Solomon codes are cyclic, which means that cyclically left-shifted codewords are also valid codewords [34]. Kaiser *et al.* [23], found that cyclic and quasi-cyclic codes are therefore vulnerable to *symbol slip*, resulting in undetected decoding errors. Symbol

slips are the result of temporary synchronization issues, causing symbol insertion or deletion. As a consequence, a shifted codeword is presented to the decoder [23]. Kaiser *et al.* report that randomizing a codeword by bitwise exclusive-ORing it with a pseudo-random sequence represents an effective low-cost solution for the problem.

4.1.3 Serial Concatenated Codes

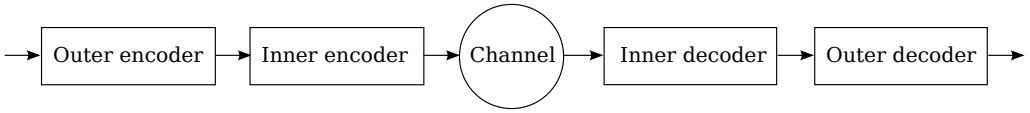


Figure 4.2: Schema of a serial concatenated code

The idea of concatenated codes is to combine multiple (complementary) codes to form one superior code, while keeping the decoding complexity low. In fact, Forney showed that with concatenated coding, it is possible to decrease the probability of errors exponentially with polynomial complexity [11].
Serial concatenated codes consists of an outer and an inner code (see Figure 4.2). The message is first encoded with the outer encoder and subsequently encoded with the inner encoder. Decoding is performed vice versa. Usually, an interleaver is added to spread possible burst errors as wide as possible. The interleaver itself can be seen as code with $R_c = 1$ that maps the message symbols to a permutation [14]. A simple interleaved convolution code may therefore be considered a concatenated code.
A common concatenated code comprises an inner convolutional code rate with $R_c = 1/2$ and $\nu = 7$, an outer RS(255,223) code and a interleaver between the two codes. It was used in Voyager 2 and remains popular in CubeSat projects. The performance of this code can be seen in Figure 4.5.

4.1.4 Turbo Codes

Turbo codes are a class of parallel concatenated convolutional codes, first published by Berrou *et al.* in 1993 [1]. Turbo codes provide significant coding gain compared to concatenated codes, with medium decoding complexity [11,14]. The structure of a turbo code is illustrated in Figure 4.3. The encoder is comprised of two identical recursive convolutional encoders with memory $\nu = 4$ and a pseudorandom interleaver π. For a message sequence u, the encoder outputs the same sequence, a parity sequence $\mathrm{v}^{(1)}$ and a second parity sequence $\mathrm{v}^{(2)}$ of the randomly interleaved message u. Of both encoders, the systematic information is omitted. Higher rate codes can be obtained by perforating parity symbols or by increasing the symbol size of the convolutional encoders (see 4.1.1). Although it is possible to concatenate more interleaved convolutional encoders in this scheme, Glavieux argues that two encoders are sufficient if the permutation is chosen wisely [14].

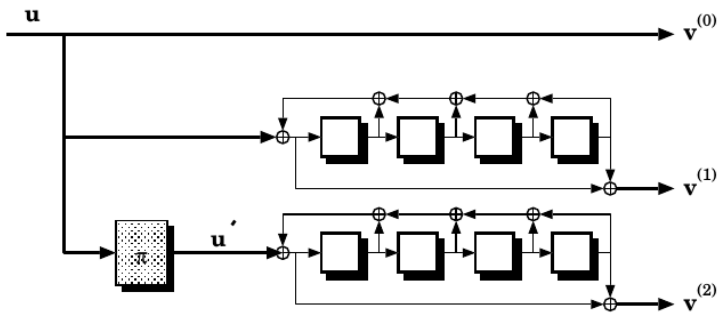


Figure 4.3: Parallel concatenated rate-1/3 turbo encoder [11]

Glavieux argues, that the advantage of turbo codes comes from the two different permutations that help mitigate the vulnerability to burst errors. Because the constraint of the parity symbols is spread across the block size of the interleaver, the probability of burst errors is reduced [14].

Bossert reports, that the low constraint size of the convolutional encoders are the cause of a sharp bend in the BER graph in the range of 10^{-4} and 10^{-8} [3], which is known as "error floor", leading to reduced performance of the code (see Figure 4.4). Turbo codes may be concatenated with Reed-Solomon codes to clean up the few remaining errors.

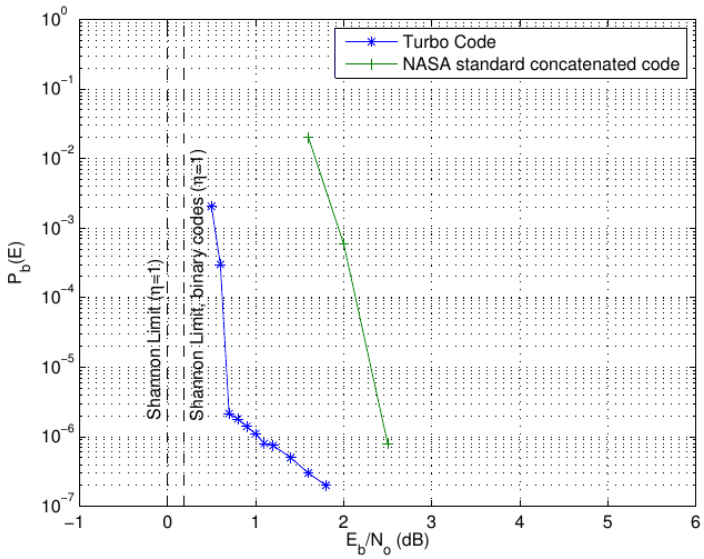


Figure 4.4: "Performance of a rate-1/2 turbo code with interleaver length $N = 216$, compared to the NASA standard concatenated code and the relevant Shannon limits for $\eta = 1$" [11]

4.1.5 Low-Density Parity-Check Codes

Low-density parity-check codes (LDPC codes) were invented by Robert Gallager in 1962 [13]. LDPC codes are parity-check codes with the particular characteristic that the parity-check matrix is sparse. Although Gallager's publication showed the good performance of LDPC codes in BSC, AWGN and Rayleigh channel, they remained unused until 1995. This may be due to the impractical complexity of the decoding procedure: Gallager reported that "(...) block lengths of about 500, an IBM 7090 computer requires about 0.1 second per iteration to decode a block(...)" [13].
In their 1997 publication MacKay *et al.* [27] showed that LDPC codes can provide error correcting capabilities similar to that of Turbo codes and can outperform the (7,1/2) convolution codes despite a higher code rate.
MacKay *et al.* [27] approximated that brute-force decoding takes about $6Nt$ floating point multiplications per iteration. For the standard CCSDS (8176,7156) LDPC code [4], this means decoding (30 iterations) requires about 47 million floating point multiplications per block. Since then, countless publications introduced different decoding methods, mostly based on the a posteriori probability decoding algorithm proposed by Gallager [13], also known as sum-product algorithm. These decoders usually reduce decoding complexity by using look-up tables.
As with Reed-Solomon codes, randomizing the codewords prior to transmission is recommended for quasi-cyclic LDPC codes to prevent synchronization issues and symbol slips [7, 23].

4.2 Evaluation

In this section a few instances of the channel coding schemes discussed above are compared by throughput in the noisy Rician channel. Afterwards their decoding complexities are compared and the problem of additional delay due to non-alignment of frames and codeblocks are addressed. Finally, the compliance of these codes with the requirements of the MOVE 2 mission is assessed.

4.2.1 Performance

Here, the performance of the previously discussed channel codes in the channel of section 3.4 is evaluated and compared. The codes are evaluated by throughput, the product of correctly received codewords and code rate. The simulations are performed using Matlab.
Reed-Solomon codes provide the best performance in terms of maximum throughput when the channel noise level is rather low. Figure 4.5 reveals, that for high E_b/N_0 ($\geq$15 dB), the tested Reed-Solomon code allows higher throughput than any other code, but leaves only small tolerance margins. This is problematic for transceivers with variable bit rate, because it would require the code to provide robust performance over ranges

of multiple decibel. For example, if the receiver E_b/N_0 is 16 dB at 1200 bit s^{-1}, it would drop to 7 dB at 9600 bit s^{-1}. At this rate, a RS(160,152) code would decode approximately 50% of the received code blocks incorrectly and render communications impossible. Figure 4.6 shows the performance of other Reed-Solomon codes. Although the matrices

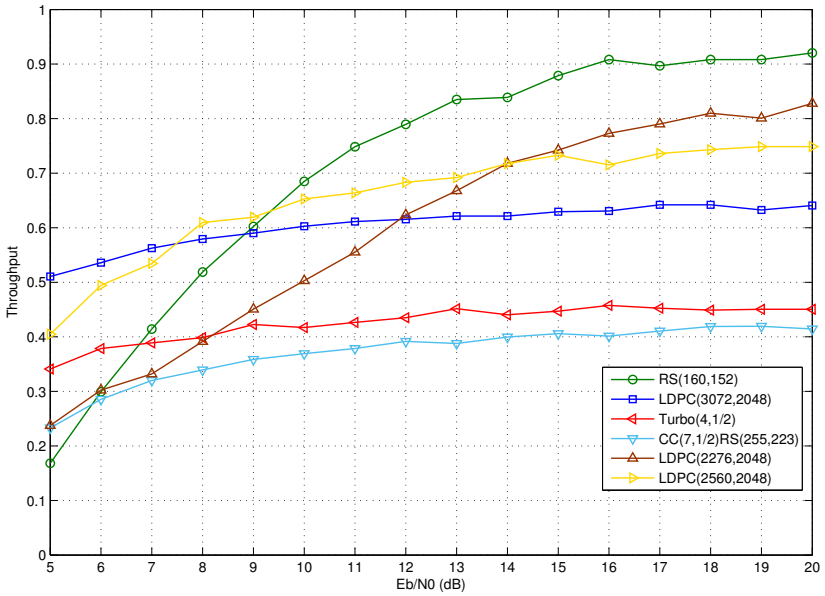


Figure 4.5: Throughput of different FEC schemes

were generated randomly, is is visible that low-density parity-check codes can yield very good error correction performance with wide ranges of code rates. The tested LDPC codes are inferior to the Reed-Solomon code for high E_b/N_0 but offer the best performance for medium to low E_b/N_0 values. Especially the rate-2/3 LDPC code is worth mentioning. It offers higher throughput than the rate-1/2 turbo code with less or equal information loss. Turbo codes do not seem to perform very well in burst error channels and cannot benefit from the high E_b/N_0 , possibly due to their error floor. The serial concatenated code performs worst.

4.2.2 Decoding Complexity and Delay

Another important selection criterion for codes is the computational power required to use them at the required data rate. While encoding is, generally speaking, a simple task, decoding can be very complex. Spacecraft hardware has very limited computing power, due to electrical power and weight constraints, which is unfavorable for complex codes. To attain a basic orientation of the decoding complexity of the individual codes, a test was performed. Afterwards, the delays added by the use of block codes are discussed.

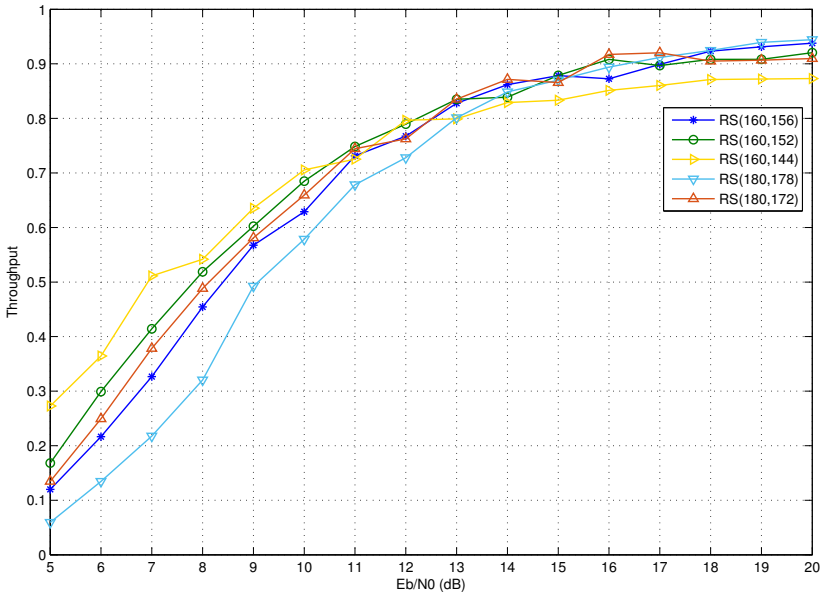


Figure 4.6: Throughput of different RS codes

4.2.2.1 Complexity

To compare the decoding complexity of each FEC scheme, the decoding of 512 kB in total of random data from the Rician-AWGN channel, was simulated and timed. The test was performed in Matlab on a computer with a Intel i7-2600K processor and a GeForce GTX570. The results are displayed in table 4.1. Trials with GPU accelerated decoding are marked with an asterisk(*).

Code	**Trials**	$\sum \Delta t$
LDPC(3072,2048)	2000	68.9 s
LDPC(3072,2048)*	2000	11.5 s
Turbo(4,1/2)	1000	10.4 s
Turbo(4,1/2)*	1000	6.4 s
CC(7,1/2)RS(255,223)	2296	5.4 s
CC(7,1/2)RS(255,223)*	2296	3.3 s
RS(160,152)	3369	0.70 s
RS(168,152)	3369	0.76 s
RS(184,152)	3369	0.91 s

Table 4.1: Decoding time of codes

Evidently, LDPC decoding is the most costly procedure of all by a wide margin. However, it must be noted that the LDPC decoder also benefits most from parallelized computation. With GPU acceleration decoding of LDPC codes was reduced by a factor of 6, whereas the decoding time of Turbo codes and the concatenated codes could only be reduced by

38% and 42% respectively. This suggests good performance of LDPC codes on FPGAs and may be improved even further with more efficient algorithms.

4.2.2.2 Delay

Additionally to the delay due to decoding, coding causes an additional reception delay depending on its block length and the channel bit rate. For convolutional codes, the block length is the constraint length. The delay here is negligible, since the constraint size rarely exceeds 10 bit. Block decoders on the other hand, need to receive the entire block before decoding can take place. As a result, the message bits also arrive in blocks. The interval between two waves is the serialization time of a code block. In this context, it is also referred to as *block delay* to distinguish it from the frame deserialization time. If frames are not aligned with code blocks, block delay has a high impact on the frame deserialization time.
For a code block of length N and data rate r, the block delay is:

$$t_b = \frac{N}{r} \tag{4.9}$$

Neglecting decoding delay, the minimal and maximal frame serialization time is calculated as follows:

$$t_{f,min} = t_b \cdot \left\lceil \frac{N_F}{K} \right\rceil \tag{4.10}$$

$$t_{f,max} = t_{f,min} + t_b = t_b \cdot \left(\left\lceil \frac{N_F}{K} \right\rceil + 1 \right) \tag{4.11}$$

where N_F denotes the frame length and K is the dimension of the code block. The consequences of this are best illustrated with an example: A frame of length $N_F = K + 2$ may be spread across three code blocks instead of two. Alternatively frame of length $N_F = K$ may be spread across two blocks instead of one.
For these reasons, the block length should also be chosen with respect to data rate and maximum frame length. It is sensible to use block lengths smaller than the maximum frame length, so that the delay remains manageable for smaller frames.

4.2.3 Assessment

The decision for the most suitable FEC scheme must be separated into up- and downlink to account for the different requirements and channel characteristics.
In the previous chapter, the uplink channel was found to be mostly affected by AWGN in contrast to the strong fading of the downlink channel. The uplink FEC scheme therefore needs not provide the capability to compensate for fades. Also, the FEC scheme is not required to provide significant coding gains as the received E_b/N_0 can simply be improved by increasing the ground station transmitter power. The limiting factors of the uplink are the low data rate of 300 bit s^{-1} to 1200 bit s^{-1}, and the low decoding

power, which are imposed by the available hardware. The combination of low decoding complexity, their MDS properties, and variable block length render Reed-Solomon codes the best choice for the uplink.
The downlink requirements are quite different. The low transmitter power must be compensated for by the downlink FEC scheme, so that even low gain antennas of radio amateurs on the ground can receive the satellite telemetry signals. Because of the low data rates, it can be assumed that all commercially available PC systems provide sufficient computation power to decode all of the aforementioned codes in real time. The FEC code must be able to operate efficiently despite possible fades. Additionally, the FEC scheme must use as little bandwidth as possible to still meet the other requirements. As discussed above, the rate-2/3 LDPC code performs considerably better than the NASA serial concatenated code and the rate-1/2 Turbo code. Therefore LDPC codes are the preferred choice for the downlink channel. Additionally, high-rate Reed-Solomon codes can be used for situations where a higher data rate is required (e.g. downloading larger amounts of data).

4.3 Conclusion

In this chapter, several forward-error correction schemes typical for the use in satellite communications, were introduced and evaluated for the use on MOVE 2. In the comparison with respect to maximum throughput in the noisy fading channel of the previous chapter, high-rate Reed-Solomon codes emerged as most suitable for very high E_b/N_0 values. Rate-2/3 LDPC codes were found to perform best for high to medium E_b/N_0 values. The examination of the respective decoding complexity revealed, that LDPC code decoding is most computationally intensive, while Reed-Solomon decoding is fastest. Turbo decoding and the NASA concatenated code are of moderate complexity. It was shown that LDPC decoding benefits from increased parallelization more than any other code. When compared to convolutional codes, block codes were shown to increase the frame deserialization time, especially when frames and codeblocks are not aligned.
In the assessment, Reed-Solomon codes were found the be most suitable for use on the uplink channel. LDPC codes were chosen for the downlink channel because of their good performance for wide E_b/N_0 ranges.

Chapter 5

Related Work

This chapter illustrates three protocols with related areas of application, and that influenced the design of the Nanolink protocol. Section 5.1 discusses CSP, a simple protocol stack for Cubesats. Section 5.2 gives an overview of the Proximity-1 protocol of CCSDS. Afterwards, the Satellite Transport Protocol is briefly introduced in section 5.3. The chapter is concluded in section 5.4.

5.1 Gomspace Cubesat Space Protocol

The Cubesat Space Protocol (CSP) is a small network-layer protocol intended for use in microcontrollers [15]. The protocol was developed by students of the Aalbord University in 2008 and now is maintained by the company GomSpace. A user space implementation, LibCSP[1], is available under the GNU Lesser Public License (LGPL) for FreeRTOS and POSIX compatible operating systems.
LibCSP provides support for I^2C and CAN bus systems as well as KISS and loopback interfaces. It is possible to add additional drivers. In the standard topology, CSP is envisaged to be implemented on each satellite subsystem and the ground control server. The subsystems can be addressed directly from other subsystems or from the ground.
The CSP protocol stack was designed to provide the similar services as the TCP/IP, or UDP/IP protocol stack, but with reduced resource requirements. The CSP protocol stack provides an end-to-end connectivity and routing. To do so, the protocol stack comprises two transport-layer protocols. The first is a connectionless unreliable datagram protocol, comparable to UDP. The second is RDP (RFC908 [33]), a reliable datagram protocol designed as alternative to TCP for situations where less complexity and overhead are required. RDP is connection-oriented and provides reliable delivery, flow control. In-order delivery is optional. Unlike TCP, RDP is packet-oriented. For security reasons, LibCSP supports packet encryption and authentication using XTEA and SHA-1 based HMAC.

[1] `http://www.libcsp.org`

CSP itself represents the network-layer part of the CSP protocol stack. It provides routing capabilities for small fixed networks. The 32-Bit CSP header is an amalgamation of transport and network routing headers. It mainly contains port and address information of source and destination. CSP can address 32 different hosts with 64 different ports each. Since the network is assumed to be static, CSP features no dynamic routing functionality. However, mechanisms to alter the routing tables at runtime are available in CSP. The CSP header does not contain a length field. A length field is only provided by the RDP header.
The protocol is reported to be incorporated in the launched GOMX-1, GOMX-2[2] and AAUSAT3 Cubesats.

5.2 CCSDS Proximity-1 Space Link Protocol

The Proximity-1 Space Link Protocol is a recommended standard (Blue Book) of the Consultative Committee for Space Data Systems (CCSDS) [6, 8]. The proposed application is bi-directional communication "among probes, landers, rovers, orbiting constellations, and orbiting relays" [6]. Proximity-1 was designed to operate in low delay links with distances up to 100,000 km. The name 'Proximity' was chosen to express the difference to deep-space protocols.
Proximity-1 is used in multiple Mars missions of NASA and ESA. It is used for the communication of the NASA Mars Science Laboratory [8]. Proximity-1 handles communication between the rover Curiosity and the orbiter. The orbiter relays data (e.g. telemetry and telecommands) between Mars and Earth (see figure 5.1).

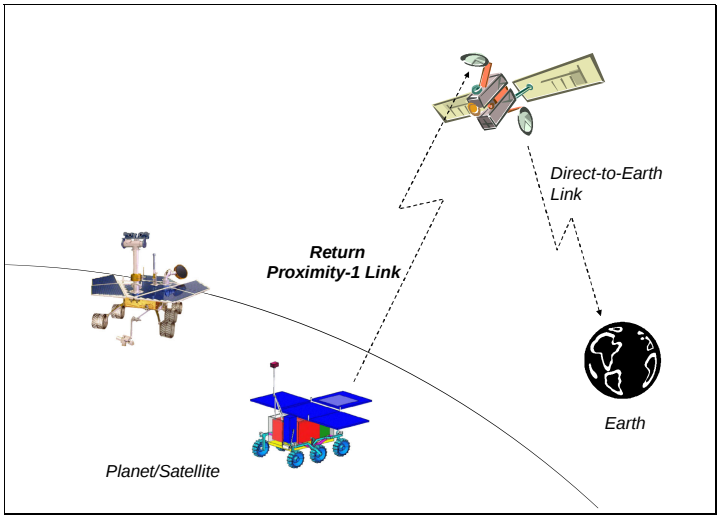


Figure 5.1: Proximity-1 Relay Telemetry Link [8]

Proximity links are point-to-point connections typically with asymmetric communica-

[2]GOMX-2 was destroyed when the launch vehicle exploded on October 28th, 2014

tion resources. Connections are possible in either full duplex or half duplex modes. A simplex transmission and receive mode is also available. The primary forward error correction scheme for Proximity-1 are the (2048,1024) LDPC codes of the AR4JA family (see reference [5]). Concatenated codes, Convolutional codes, and no encoding are also possible. To allow for a flexible frame length, Proximity-1 frames are not aligned with the codewords. Additionally to forward error correction, the protocol uses a go-back-n ARQ scheme. The go-back-n mechanism is chosen over selective ARQ for the sake of reduced complexity. Data sent over Proximity links are Space Packets, encapsulation packets, and arbitrary data. A distinctive feature of Proximity-1 is its flexibility. It is possible to change transceiver parameters such as frequency, data rate, modulation as well as coding and duplex mode.

5.3 Satellite Transport Protocol

STP is a transport protocol for satellite networks, developed in 1997 by Henderson and Katz [17]. STP is an adaption of SSCOP, an ATM based protocol. The protocol is reported to work efficiently in high bandwidth-delay or BER links. STP was shown to perform at similar or higher efficiency than TCP while requiring less backward channel bandwidth [17].

To reduce packet overhead, control information and payload data are separated into different packets. For this reason, there are four different packet types in STP, named “STAT(us)”, “USTAT”, “POLL” and “SD”. The first three types are part of the STP ARQ mechanism, SD packets contain payload. The ARQ scheme STP and SSCOP is based on selective-acknowledge (SACK) and negative acknowledge (NACK) messages. The STAT and POLL packets form the SACK unit. POLL packets are sent periodically either due to timeouts or number if a certain number of packets was sent. A POLL packet contains a timestamp and the sequence number of the next in-order packet. Upon reception, the receiver answers with a STAT packet containing the POLL timestamp, the current receiver window value, the sequence number of the last received in-order packet and the sequence numbers of all missing packets within the current window. USTAT packets are negative acknowledge packets that complement the rather costly SACK procedure with a more prompt and inexpensive retransmission request.

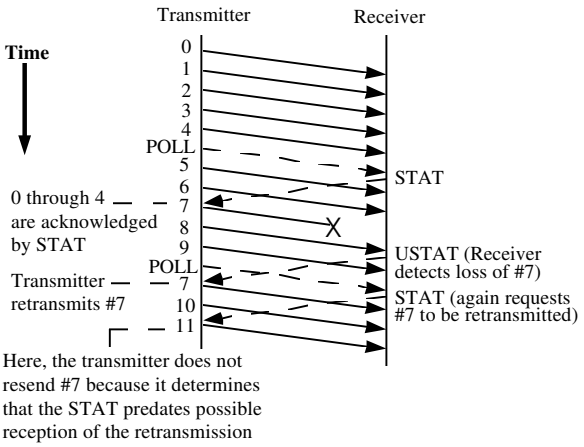


Figure 5.2: Example of SSCOP ARQ [17]

5.4 Conclusion

The Cubesat Space Protocol is an interesting replacement for TCP/IP, and respectively UDP/IP, for low data rate communication links, such as the radio link of MOVE 2. Its POSIX compatibility and low memory usage render it an ideal candidate for the use in the MOVE 2 on-board data handling systems. CSP can be used to for the end-to-end delivery of data between applications running on the satellite's CDH computer and ground control. It can be used to route the traffic of different applications to different transceivers. This is helpful in case a second transceiver, or transmitter, is used. In this case, payload data could be sent exclusively over the secondary radio device, while all other traffic is sent over the primary radio. If one of the radio devices becomes inoperative, the traffic can be rerouted to the other radio. Since Nanolink already offers reliability and sequenced delivery, these features are not required from CSP.
Proximity-1 includes many features that are also required for MOVE 2. The connection-based full-duplex transmission mode is ideal for the needs of MOVE 2. The ability to change transceiver and protocol settings during flight adds the ability to adapt to changes in the environment. The expedited service is an interesting feature for expendable data such as telemetry information. Also not aligning the protocol frames with the codeblocks is a good abstraction for when different sized frames or coding schemes are needed. On the other hand, there are some downsides to Proximity-1 that render it not a good choice for the MOVE 2 mission. The first is that the required combination of connection-based modes and a simplex telemetry beacon mode is not provided for. The flexibility of Proximity-1 adds overhead in form of frame header fields that are not required in this case. The biggest downside of Proximity-1 is its ARQ protocol. The go-back-n mechanism is not optimal since valid but out of sequence frames are dismissed. This problem increases with window size and frame error rate. Unfortunately, smaller window sizes do not scale well with asymmetric data rates, and require more bandwidth on the reverse channel [8].
STP is made for situations different to those typically encountered with CubeSats. Long delays or high bandwidths are both not found in the MOVE 2 mission. The associated big sequence number spaces and port number ranges are not required for a link with two nodes and low bandwidth-delay product. However, the ARQ mechanism of STP is designed with asymmetric bandwidths and efficiency in mind and also shown to work well in high BER conditions. Except for the high bandwidth-delay products, the requirements for the Nanolink ARQ mechanism are the same. It also must operate efficiently on asymmetric links with occasionally high error rates. It is therefore adapted in Nanolink.

Chapter 6

Concept

This chapter discusses the concept of the Nanolink Data Link Layer Protocol. Section 6.1 contains an overview of the architecture and capabilities of Nanolink. Subsequently, a detailed description of the Nanolink Transfer Frame format and the Extension Header format is found in section 6.2.1 and section 6.2.2 . Section 6.3 discusses the communications procedures of Nanolink. The chapter is concluded in section 6.4, where the coding and synchronization of Nanolink is presented.

6.1 Overview of the Nanolink Protocol

This section offers an overview of application and features of the Nanolink protocol. First, an introduction to Nanolink is provided. Subsequently, the application in the MOVE 2 protocol stack and the core features of Nanolink are illustrated.

6.1.1 Introduction

The Nanolink protocol is a data link layer protocol designed to provide reliable and efficient communication with remote assets in low Earth orbit. The focus here is on nanosatellites such as CubeSats.
The increasing capabilities of CubeSat hardware bring a higher demand for complex mission and satellite design. This results in a need for more throughput on the inherently slow radio links. The intent of Nanolink is to satisfy this demand while also retaining the rich feature set that is expected of a modern radio link layer protocol.
Nanolink is intended to operate in low bandwidth-delay radio links with high asymmetry and moderate to weak signal quality. Reliability and efficiency are achieved by utilizing a hybrid ARQ scheme that is designed to minimize the required reverse channel bandwidth and to be robust to short connection losses. In consequence, higher code rates are possible because the channel coding is not required to handle all errors. Additional losses are handled by retransmissions. Thus, the code rate can be adjusted to achieve maximum efficiency in the average case. The associated higher code rate results in a higher upper bound of the protocol efficiency for best case conditions. Overall, hybrid ARQ can react

to changes in the channel conditions more dynamically, since the overhead of redundant transmission only occurs if necessary. For bursty, therefore quickly varying channel conditions, like on the VHF downlink, this is ideal.
The satellite radio links are point-to-point connections with full duplex support. It is assumed that the radio channel and respective frequencies are used exclusively by the ground station and satellite. Therefore, the protocol features no special media access control mechanisms for the physical layer.
Two different types of data services are most commonly needed in CubeSat missions:

1. Satellite-to-ground telemetry broadcasting
2. Full duplex connection based point-to-point communication

Due to the periodical line of sight loss that comes with low earth orbits, bidirectional communication in Nanolink is connection based. The connection-orientation is necessary to handle the physical connection loss by transitioning to telemetry broadcast. Connections are suspended at the end of each pass and resumed upon the next pass. Nanolink provides both stream- and packet-oriented services. The streaming service is intended for use with protocols other than CSP and to relay arbitrary user data. The packet oriented service is intended for use with CSP. CSP packets do not provide information about their individual length, which is why this information is transported to the receiver by Nanolink.

6.1.2 Application

The prototypical application scenario of Nanolink is the employment and operation on the MOVE 2 CubeSat. The protocol implementation will exists as separate hardware within the COM subsystem.
Nanolink is the connector between the satellite subsystems and ground control. The protocol manages the physical connection and provides a link for upper-layer protocols. The reliable transfer offered by Nanolink enables the use of simple transport protocols that do not need to implement their own reliable service or packet reordering. This has the advantage that no additional handshakes and transfer of acknowledgments are required. Since it is expected that multiple applications will run on the satellite, this can increase the efficiency of the protocol stack.
On the send side, the Nanolink interface must be supplied with either a byte-stream or packets, the target address (PID, see 6.3.3), and information about the required quality of service, depending on the PID. The choice whether a PID is packet or stream oriented is implementation specific. On the receiving side, Nanolink delivers packets or a byte-stream to the respective target address.
In the proposed MOVE 2 protocol stack (see Figure 6.1), Nanolink is complemented by the CSP protocol suite. CSP provides communication services between the CDH and ground control applications. Potential applications are telecommand (TC), telemetry

(TM) retrieval, and file transfer (e.g. experimental data). Its routing capabilities are beneficial if multiple COM systems (e.g. an additional downlink transmitter) are used. To connect Nanolink with a CSP network, a CSP node on the Nanolink hardware is required. The node fetches the CSP packets from the data bus and supplies them to the Nanolink interface using a driver which must be written for this purpose.

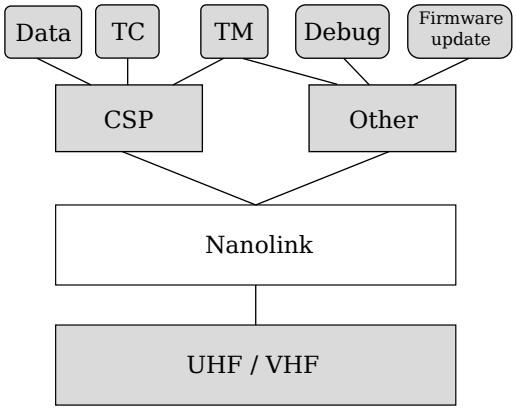


Figure 6.1: MOVE 2 Protocol Stack

The design of Nanolink enables accessing remote interfaces without requiring additional encapsulation. In order to do so, Nanolink acts as transparent data relay between the interface on the satellite and the ground control software. Since Nanolink provides reliable transmission and fragmentation, no additional protocol logic is necessary. In MOVE 2, the stream-oriented service can be used e.g. for debugging and firmware upload via JTAG ('Other' in Figure 6.1).

6.1.3 Protocol Features

Transmission Modes—Nanolink offers a simplex telemetry beacon mode and a connection oriented mode. The beacon mode is the default modus operandi of Nanolink and enables amateur radio operators to receive telemetry data when the satellite is not passing over the LRT ground station. It also serves as satellite discovery mechanism, which is used for connection establishment. The connection oriented mode provides point to point data transfer services between ground station and satellite. This mode is entered only by the initiative of the ground station. The connection is suspended once the physical connection between satellite and ground control breaks and automatically resumed once the satellite's beacon signals are discovered.

Forward Error Correction—To compensate for the frequent impairments on the radio channel, Nanolink employs forward error correction. Since the channel characteristics may change during the operation phase, it is possible to switch between forward error correction schemes, codeword length and code rate. The forward error correction system provides a logical communication channel with lower BER than the actual

physical channel. The advantage of this concept is a loose coupling between frames and codewords and the resulting variable frame lengths and codewords.

Quality of Service—Nanolink offers two different types of quality of service: A sequence controlled service that offers reliable and in-order delivery and an unreliable service without sequence control.
To ensure reliable transmission, Nanolink employs two different ARQ types. The first is negative acknowledge, which is used to stimulate prompt retransmission. The second is combines selective negative acknowledge and cumulative acknowledge (see section 6.3.3.1).
The unreliable service is thought to be used in cases where reliability is of secondary importance or ensured by other means.

Addressing—Nanolink is implemented on separate hardware. This hardware is connected with the other satellite subsystems via one or many data buses. These buses are associated with a respective subsystem and can be addressed directly using the port identifier (PID). Additionally, this allows for direct transmission of debug or telemetry data, without the need of further encapsulation. Prioritized traffic based on the address is possible.

Payload—The payload data of Nanolink can either be a stream of data or packets. The content of the stream or packet data is not evaluated by Nanolink at any point and therefore ensures loose coupling of protocol logic. Nanolink can combine multiple packets into a single frame and offers fragmentation for larger packets.

6.2 Structure

This section discusses the structure and contents of Nanolink protocol data units (PDU). First, the format of Nanolink transfer frames is illustrated. Afterwards, the concept of extension headers is presented and discussed.

6.2.1 Transfer Frame Format

Nanolink PDUs are frames of variable size. Payload and control data are incorporated into a framing structure as illustrated in Figure 6.2. A frame starts with a 3 B header containing information about the contents of the frame. The byte order of the contents of a frame is big endian, 1 byte is exactly 8 bits in length. The frame header is followed by the frame data field, containing optional additional header information and payload data. The maximum length of the frame is 512 bytes, so that the total maximum data field length is 509 bytes. Each frame is immediately followed by a CRC-16 checksum that is calculated over the frame.

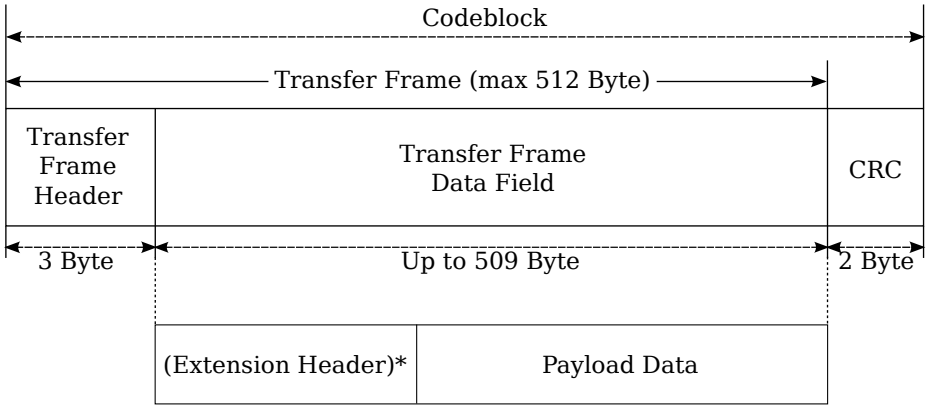


Figure 6.2: Structure and Contents of a Transfer Frame

6.2.1.1 Transfer Frame Header Format

Transfer Frames begin with the Transfer Frame Header, a 3 Byte long data field comprised of the following fields:

1. Frame Sequence Number, 8 Bits
2. Retransmit Flag, 1 Bit
3. Extended Header Flag, 1 Bit
4. Fragmentation Flag, 1 Bit
5. Length Field, 9 Bit
6. Port Identifier (PID), 2 Bits
7. Reserved, 2 Bits

The structure of the frame header is illustrated in Figure 6.3. The purpose of the individual fields and flags is discussed in the following.

Sequence Number—The Frame Sequence Number is an 8 Bit unsigned integer. It is used to identify individual Transfer Frames, and to detect transmission losses. Because of the low bandwidth-delay product, a bigger sequence number space is not required. Another reason for 8 Bit sequence numbers is the byte-alignment, which means that no padding bits are required if sequence numbers are used elsewhere, e.g. in ARQ messages.

Retransmission Flag—The Retransmission Flag indicates whether or not the frame is sequence controlled and is to be retransmitted in case of transmission losses. If the Retransmission Flag is set to '0', the sequence number is not checked and the frame is not placed in the sent queue after transmission. This mode is used for telemetry beacons, during connection establishment and expendable data.

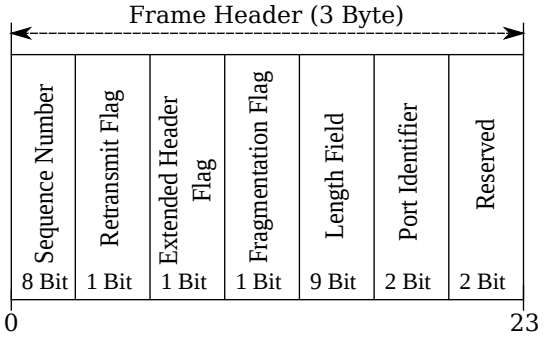


Figure 6.3: Transfer Frame Header

Extended Header Flag—The Extended Header Flag indicates the presence of Extension Header Packets in the Transfer Frame Data Field.

Fragmentation Flag—The Fragmentation Flag is set if the Transmission Frame Data Field holds data that is part of a larger fragmented packet.

Length Field—The Length Field is a 9 Bit unsigned integer. Its value represents the length of the uncoded Transfer Frame excluding the CRC. It is used to delimit the size of the frame and to locate the CRC checksum.

Port Identifier—The Port Identifier (PID) specifies the destination of the frame payload data. Possible destinations are: *OBDH/CDH*, *Debug* and *Telemetry*. If the frame does contain telemetry data, the PID is to be set to *Telemetry*. If no payload data are present, the PID field is ignored. The remaining PID is reserved for future use. Each transfer frame must only contain payload data of exactly one destination or source.

6.2.1.2 Transfer Frame Data Field

The Transmission Frame Data field contains payload and extension header packets. Its contents are byte aligned with a maximum of 509 bytes, to set the length of the uncoded frame to a power of two.

6.2.1.3 Cyclic Redundancy Check

To minimize the probability of undetected errors, every Transfer Frame is followed by a CRC-16 checksum. A 16 bit CRC was chosen because it is assumed to provide sufficient error detection in the combination with the forward error correction. Frames with invalid checksums are discarded.

6.2.2 Extension Headers

The Extension Headers are a distinctive feature of the Nanolink Protocol. They are optional protocol control elements that allow increased versatility and loose coupling without introducing significant constant overhead.

Because the protocol is designed with low bandwidth radio links in mind, an important requirement is low protocol overhead. Not all headers are required in the standard cases and therefore removed from the frame header, but not moved into an special frames, as was done with Proximity-1 or STP. Control frames, although possibly small, also need a complete frame header and syncword to detect the frame, and to distinguish it from data frames. The minimum size of the frame header is one byte, due to byte alignment, and therefore just as long as a minimum extension header. Due to the additional syncword, this solution offers high overhead while adding no value for low bandwidth-delay products.

Instead, the concepts of control frames and data frames are merged, so that only one frame format exists within Nanolink. This may cause additional delay, since long frames increase the delay for the receipt of control information. However, this is not a concern, since the bandwidth-delay product is assumed to be very small. Additionally, higher bandwidths/delays or channel asymmetries can be accommodated since it is possible to send frames without payload, containing only control information.

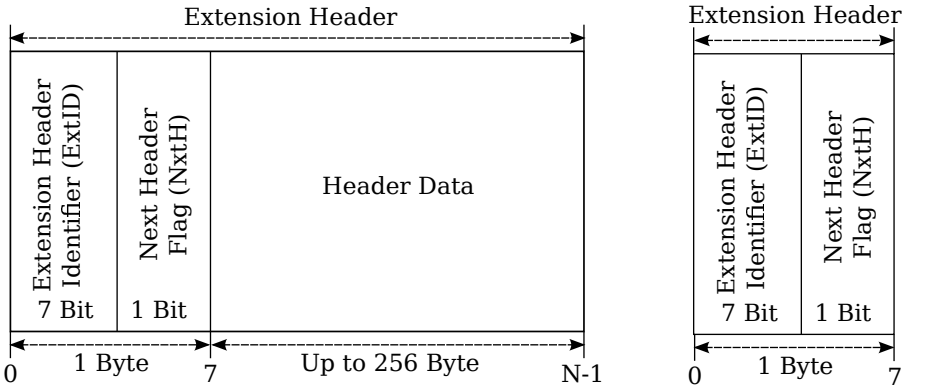
Extension headers are designed to be daisy chained, so that multiple extension header packets can be placed within one frame. The presence of a further extension header is indicated by setting the NxtH flag of the extension header.

The advantage of this method is that it allows a very high number of extension headers inside a frame without the permanent overhead of a dedicated header counter field. The byte alignment requires the header packets to be at least 8 bit in length. A 8 bit ExtID field could distinguish 256 different header types, an excessive amount, which is not required. Repurposing 1 bit of the header byte brings therefore no considerable limitations. However, this bit can be used as NxtH flag for header chaining. Therefore, only minimal overhead is introduced.

A header counter would add permanent overhead, even if no extension header is used. Also, it would either limit the possible number of extension headers in a frame severely if the reserved bits of the frame header were used, or would require an entire additional byte. In any case, this solution is inferior to the chaining method.

6.2.2.1 Extension Header Format

The very basic extension header structure (see Figure 6.4b) comprises an identifier field and next header indicator. Depending on the purpose of the header, it may contain additional header data of up to 256 bytes (see Figure 6.4a).



(a) Parameterized (b) Without Parameters

Figure 6.4: Extension Header Formats

The Extension Header Packet (see fig. 6.4) comprises the following fields:

Extension Header Identifier—The Extension Header Identifier (ExtID) is the first field of an Extension Header, spanning from bit 0 to 6. It indicates the purpose of the Extension Header Packet and what parameters are to be expected. See Table 6.1 for a list of all possible values. The length of this field is 7 bits so that the minimal header size is byte aligned.

Next Header Flag—Bit 7 contains the next header flag. If this flag is set to '1', this Extension Header is followed by another Extension Header Packet.

Header Data—The Header Data field is the last Extension Header field and contains the parameter data. If the parameter size is variable, the first byte represents the length of the parameters. This field is optional and must only be present if the ExtID specifies so.

6.2.2.2 Extension Header Types

The ExtID field allows for 128 different header types. The header types required for basic protocol functionality are listed in Table 6.1 along with a short overview below.

NACK—Negative Acknowledgment – The purpose of the NACK header is to immediately request retransmission when a frame was lost or incorrectly decoded.

STAT—Status – The STAT header reports the current status of the receive buffer. The header data comprises the sequence number of the last correctly received frame, the sequence number of the last correctly received in-order frame and the sequence numbers of all missing frames within that range.

POLL—Status Polling — Requests a STAT packet from the receiving node.

Mnemonic	Function	Param Count
NACK	Negative Acknowledgment	1
STAT	Selective Acknowledgment	N+2
POLL	Request STAT	0
CC	Close Connection	0
RST	Reset Protocol	0
NMD	No More Data	0
FRG	Begin Fragmented Packet	2
MLT	Multiple Packets in Frame	N+1
CHC	Change Forward Error Correction	1

Table 6.1: Extension Header Types

CC—Connection Close — This header closes the connection once both participants have no more data to send.

RST—Protocol Reset — This header forces a complete reset of the protocol at the receiving side. It is intended as ultima ratio of ground control in the event of unrecoverable errors. This header cannot be sent by the satellite, because all commandeering is done by the ground station.

NMD—No More Data — This header indicates that the sender's send queue is empty.

FRG—Begin Fragment — This header indicates the beginning of a fragmented packet. The header data contains a two byte length field that represent the length of the unfragmented packet. See 6.3.3.5 for more a detailed description of the fragmentation procedures.

MLT—Multiple Packets — This header indicates that the Frame Data Field comprises multiple packets that follow after the Extension Headers. The header data contains the number of packets and the length of each individual packet. In this case, the maximum length of a packet is restricted to 256 bytes.

CHC—FEC change — This header urges the receiver to change its FEC scheme. See 6.3.2.3.

6.3 Communications Procedures

This section discusses the operation of Nanolink in detail. The section is structured in three parts. First, the two different transmission modes of Nanolink are explained. Subsequently, the procedures for connection management are discussed. Finally, the

procedures involved in the transmission of data and protocol control information are presented.

6.3.1 Transmission Modes

To accommodate the different requirements imposed on the protocol, the protocol operations are divided into a simplex "beacon mode" and a full duplex connection based transfer mode.

Beacon Mode—The beacon mode is the default modus operandi of Nanolink. In this mode, the satellite broadcasts a continuous stream of telemetry data to potential listeners on earth. Its primary purpose is to enable reception of telemetry information by amateur radio operators and other institutions, when the satellite is not passing over the ground station. Thus, it is possible to keep track of the satellite status during periods between contact with ground control. Telemetry beacons are normal Nanolink frames with the PID set to '*Telemetry*', sent using the unreliable service. A secondary use of this mode is satellite discovery and transceiver adjustment. The continuous transmission allows easier adjustment of the Doppler shift and constant symbol lock until connection establishment. In this mode, the ground station transceiver is idle and waits for signal from the satellite.

Connection Mode—The full duplex data transmission mode is the most important mode of operation of Nanolink. In this mode, ground control can control the satellites systems in real time, e.g. update the operating system, install new software, retrieve data or debug the OBDH subsystem via JTAG. Satellite and ground station enter this mode after completing connection establishment (see 6.3.2.1).

6.3.2 Connection Procedures

The following contains a description of the connection establishment and termination procedures. Afterwards the procedure for changing protocol parameters during deployment is discussed.

6.3.2.1 Establishment and Resume

The connection establishment process is illustrated in Figure 6.5. Once the ground receiver is synchronized with the satellite signal, the ground transmitter is turned on. The transmitter then radiates an acquisition pattern, long enough for the satellite receiver to synchronize with the signal (symbol lock). The acquisition pattern is followed by a codeword containing a frame with a STAT header (STAT message), to synchronize the ARQ queues. The following codewords contain an idle pattern until another STAT message is sent or the connection is established. Connection establishment is finished when a STAT message is received.

Once the satellite receiver is synchronized, the transmitter switches to connection mode and transmits a STAT message. Once a STAT message is received, the satellite resumes data service.

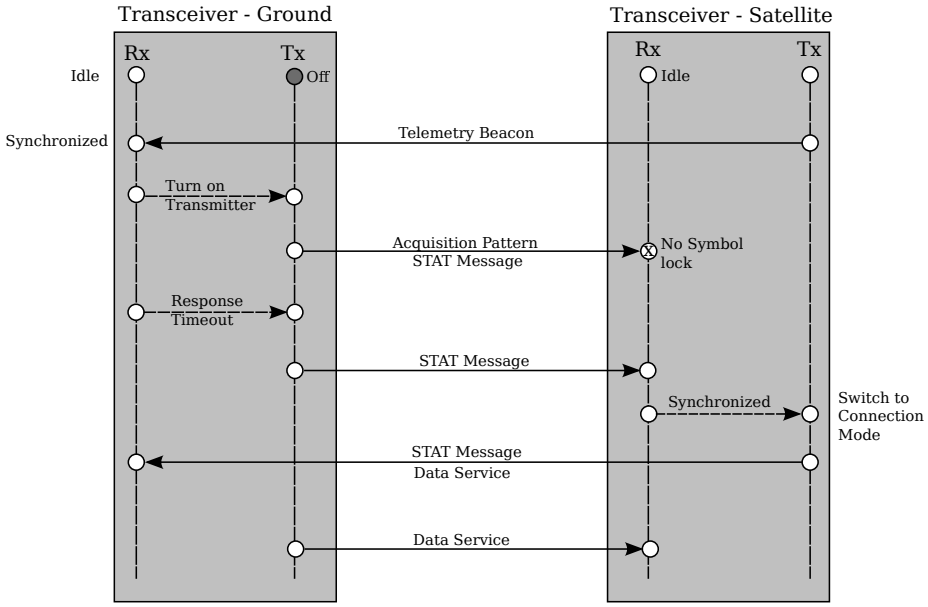


Figure 6.5: Connection establishment

To be more robust against fades and to detect the end of a connection, the process is governed by a carrier loss timer. The carrier loss timer is started once the signal of the other user is lost. The timer is stopped and reset if the signal is regained before timer expiration. Once it expires, both parties return to their initial states and the connection attempt is terminated.

6.3.2.2 Suspend and Termination

Connections are keep-alive by default. Both parties send idle patterns until the connection is suspended due to carrier loss timeout or terminated by the CC command. With a CC command, the ground station signals the satellite that there is no more data to send and the satellite may disengage from the connection once all its data is sent and acknowledged. The CC command is superseded by frames containing payload data. Ground station and satellite may explicitly report that no more data is to be sent using a NMD status header. Ground control may then decide to close the connection.

After the connection is terminated or suspended, the Nanolink instances default to beacon mode.

6.3.2.3 Mode switching

To enable the change of certain protocol parameters at runtime, Nanolink supports mode switching. Mode switching is primarily intended for changing the forward-error correction scheme. However, its use is not limited to this. The procedure was adapted from Proximity-1 for the use in Nanolink. It is illustrated in Figure 6.6. The mode switch is depicted to affect both up- and downlink transmitter-receiver pairs, however this is not necessarily the case as a mode switch might only affect the uplink and not the downlink. The process is explained using the example below.
At the beginning, satellite and ground are assumed to operate in connection mode. Ground control transmits a mode switch request frame to the satellite containing the new parameters (e.g. FEC method, code rate, etc.). The frame contains no payload data. Afterwards the transmitter radiates the idle pattern. The satellite did not receive the mode switch correctly and therefore continues operation without change. Upon carrier or synchronization loss or the reception of a sufficiently long acquisition sequence, the ground receiver applies the new parameters and performs the mode switch. After a timeout, the ground transmitter interrupts the idle pattern and repeats the mode switch request. Upon arrival at the satellite receiver, the new parameters are applied to both receiver and transmitter. The satellite transmitter resumes operation by radiating the acquisition pattern followed by codeblocks. The first frame in the first codeblock is a non-sequence controlled frame containing a poll request. The ground receiver synchronizes with the acquisition sequence and receives the valid poll frame. Subsequently, the transmitter executes the mode switch. Afterwards, it resumes operation by radiating the acquisition pattern followed by codeblocks. The first frame sent contains a STAT message as response to the POLL.
The advantage of this method is its robustness to signal impairments or losses of the physical connection e.g. due to LOS obstructions. Mode switches are persistent and affect all following connections. The reason behind this is that mode switches are intended to allow for changes in the mission requirements or adjustments to changes in the environment. Thus, switching modes for every connection is not desirable.

6.3.3 Transmission Procedures

The following discusses the various procedures involved in the transmission of data. First, the procedures of reliable and unreliable services are presented. Afterwards, rules and procedures concerning ports and extension headers are discussed. Finally, the fragmentation procedure is described.

6.3.3.1 Reliable Service

The Nanolink ARQ scheme is derived from STP [17] (see section 5.3). To reduce overhead even further, control information is no longer represented by separate packets or frames but by extension headers. The POLL/STAT mechanism is adapted to the needs of

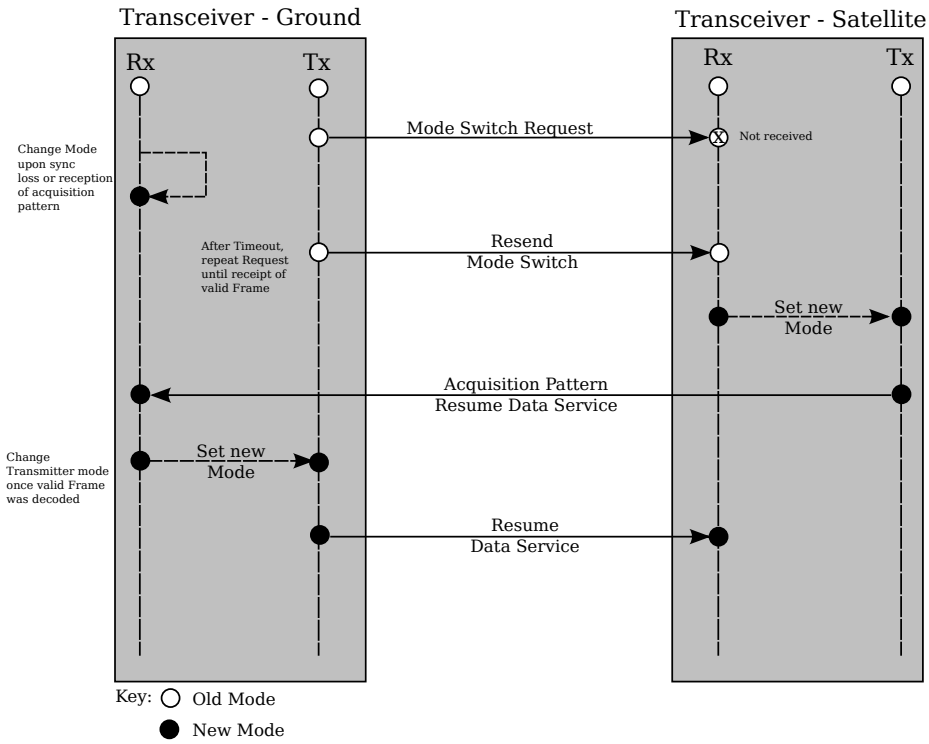


Figure 6.6: Exemplary mode switch procedure

Nanolink by removing the timestamp and window size information. They are not necessary since Nanolink links comprise only two nodes with low delay. Furthermore, the STAT message is adjusted and comprises the sequence number of the last correctly received frame, the sequence number of the last correctly in-order received frame L(R), and the sequence numbers of all missing frames within that range. The POLL message assumes a more rudimentary function as its only purpose is requesting a STAT message without carrying additional information. POLL messages may be issued to control the size of the sent queue. The advantage of this modification is that STAT messages can now be sent independently from POLL messages, and be used to control the receiver queue size. STAT messages are sent after a timer expires or the threshold for the size of the receive buffer is attained or the loss of a consecutive number of frames exceeding a certain threshold is detected.

USTAT messages are renamed to NACK to reflect their usage more appropriately. NACK messages are sent once a single frame was missed or corrupted. Their purpose is to trigger retransmissions with as little overhead and delay as possible. The detection of missed frames is based on the sequence numbers of the last two correctly received frames, for the sake of this example called $S1$ and $S2$. If $S2 = S1 + 1$ no miss is detected. If $S2 = S1 + 2$, a single miss is detected and handled by issuing a NACK message if $S1 + 1$ has not been received previously. The case $S2 \geq S1 + 3$ is handled by comparing the sequence numbers in the interval with the receive buffer and sending a STAT message

if necessary. If $S2 < S1$ and $S2$ is the second lowest missing sequence number, a NACK message for the lowest missing sequence number is issued. Else, the miss is handled by STAT messages if appropriate.
Frames are discarded if one of the following cases applies:

1. $S2$ is not within the current sequence number space
2. $S1 = S2$
3. A frame with sequence number $S2$ is already in the receive buffer

The sequence number space is delimited by L(R) and L(R) + 127 (mod 256), to account for sequence number integer overflow.
For the reliable service, each instance of Nanolink employs a receive buffer, a send and a sent queue.

- The receive buffer holds all received frames that were correctly received but could not be processed yet. Reasons for this can be out of order delivery or fragmentation.
- The send FIFO queue contains payload data pending for transmission.
- The sent queue contains already sent but not acknowledged frames. The size of the sent queue effectively limits the transmission window size.

6.3.3.2 Unreliable Service

The unreliable service is the alternative to ARQ controlled transmissions. It is primarily thought for frames with expendable information such as control messages or telemetry data. Secondary users are upper-layer protocols that implement their own ARQ and packet reordering mechanisms.
Data sent via the unreliable service is removed from the send queue and not placed in the sent queue. Unreliable frames have higher priority over reliable frames. The reason for this is that real-time data expires quickly and swiftness is preferred over reliability. The sequence number field of unreliable frames is not evaluated by protocol logic. Received unreliable frames are processed immediately. Fragmentation is not possible with this service.

6.3.3.3 Ports and Prioritization

Nanolink supports four virtual addresses, which are represented by the PID header field. The ports are used to distinguish different sources of traffic and to associate them with the appropriate destination. This is useful for when several separate or redundant bus systems are used or when interfaces shall be accessed directly. A frame must not contain data of different sources since an unambiguous association of the individual destination is not possible in this case. The outgoing data of different ports are ordered

by priority so that the most important data are sent first. The priority of the individual ports is determined by the implementation.

6.3.3.4 Extension headers

Due to losses and retransmissions, frames may arrive out of order. Extension headers containing state or control information are only valid for a short period of time and must be processed immediately. Extension headers that are not affiliated with the payload data must not be retransmitted.

6.3.3.5 Fragmentation

To circumvent the otherwise very limiting maximum payload length of 509 byte, Nanolink supports payload fragmentation of contiguous data with up to 65535 byte length. A frame with fragments in the payload data is identified with the fragment flag.
Concurrent fragmentation is only possible for different PIDs. Like all other in-order data, fragments are cleared from the receive queue and stored in a separate buffer. This is necessary, since the sequence number space is limited to 127 items and fragmenting 65536 bytes of data would fill at least 129 frames.
The first frame of a fragment sequence contains a FRG extension header with information about the length of the unfragmented packet as well as the first chunk of the packet. The following frames contain no FRG header and contiguous chunks of the packet. Fragmentation is complete once the received size of all chunks combined matches the value indicated in the first frame.
To avoid blocking from large fragmented data, frames containing data from other ports may be placed between the fragment frames.

6.4 Coding and Synchronization

This section focuses on the channel coding and synchronization methods. First, the structure and purpose of the acquisition and idle patterns is explained. It follows an overview of frame and codeblock synchronization mechanisms and their use in Nanolink. Finally, the channel coding methods are reviewed.

6.4.1 Acquisition and Idle Pattern

At the beginning of a transmission (e.g. connection establishment), an acquisition pattern is sent long enough for the receiver to synchronize with the signal. The pattern is an alternating sequence of 0 and 1, to provide enough bit transitions.
If there is no data to send, codewords are padded with the acquisition pattern. In this context, it is referred to as *idle pattern.*

6.4.2 Synchronization

The data received from the physical layer are a stream of bits. In order to distinguish the different messages in the bitstream, it is required to determine their starting symbol and length. The most common solution to the problem of finding the starting symbol is the attachment of a special symbol pattern, called *sync marker*, to the message. This gives rise to two new problems:

1. The marker may appear in the message data
2. The marker may be corrupted due to transmission errors

A simple solution to the first problem is bit stuffing. Unfortunately, this is ineffectual due to the problem 2. Transmission errors may corrupt the marker or result in the marker pattern to appear in the message data. To remedy these problems, longer marker patterns and more refined detection algorithms can be used. The increased length of the pattern renders its occurrence in the message data is more unlikely, while also helping the detection algorithm to be more robust against errors. Simple synchronizers are based on state machines and require multiple sync markers to acquire and lose synchronization [5]. More sophisticated synchronizers utilize maximum-likelihood or argmax algorithms. These algorithms constantly scan a window of symbols for sync markers and periodically report the most likely location of a valid marker [5]. A comparison of the performance of decision rules in the AWGN can be found in [5]. In 1992, Robertson [31] published a maximum likelihood decision method for better frame synchronization in flat Rayleigh fading channels. For increased robustness, the synchronization symbols are hereby interleaved with the data symbols. Kopansky *et al.* showed [25] that the same performance can be acquired with periodically embedded synchronization markers. Therefore, interleaving is not required for robust synchronization in fading channels. Additionally, Kopansky *et al.* observed that the synchronizer performance strongly depends on the utilized marker pattern.
Marker based synchronization is not the only option to find the beginning of messages. Hamkins published a maximum-likelihood brute-force synchronization method for LDPC codes, which does not require sync markers [16]. Within a window of input symbols, the algorithm selects random locations in the bitstream and performs decoding with the sequence for a number of iterations. For every location, a maximum-likelihood metric is computed. The most likely location is then decoded with a higher number of iterations. Correct decoding indicates that synchronization has been acquired. The proposed method is compatible with standard attached synchronization markers. The reduced performance for lower SNR is not a concern in the MOVE 2 mission, since low SNR are not to be expected. No information about the performance of the method in a fading channel is provided in the publication. Also, its complex computation and incompatibility with Reed-Solomon codes render it unsuitable for the use on the satellite. In order to keep to protocol complexity and asymmetry low, this method is not used in Nanolink.

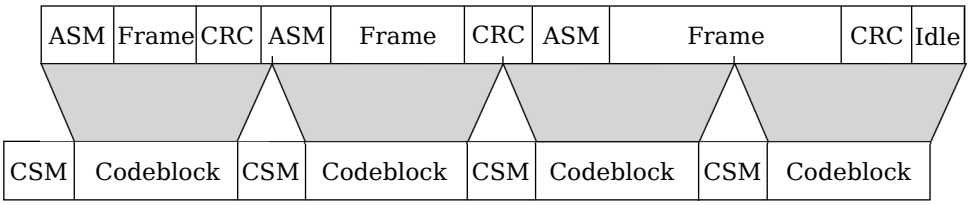


Figure 6.7: Nanolink coding and synchronization

Since the length of code words is known by the receiver, additional measures to retrieve this information are moot. This does not pertain for frames, since they are variable in size. Therefore, frame headers contain a length field, which indicates the length of each frame excluding the CRC checksum. The maximum size of 512 bytes is a compromise between the higher overhead of smaller frames and the increased delay and frame error rate of longer frames.

6.4.2.1 Codeword synchronization

For codeword synchronization in high SNR pure AWGN channels, CCSDS report 32-Bit synchronization markers to be sufficient for LDPC and Reed-Solomon codes.
Since Rician fading and modem output format (soft or hard output) also affect the decision, a final specification of the codeword synchronization markers for the downlink channel is not possible at this time.

6.4.2.2 Frame Synchronization

Frames and Codewords are not aligned, thus additional markers for frames are required. Therefore, frames are delimited with a 24-Bit attached synchronization marker (ASM). The value of the marker is in hexadecimal notation: `0x4E4D45`. The ASM length is relatively short compared to that of the CSM. The rationale behind this is that the decoder output can be assumed to be almost error-free and therefore a very rigorous matching algorithm can be used to make up for the higher probability of occurrence of the pattern.
Two frames may be separated by idle periods, where only the idle pattern is sent. Therefore, the ASM cannot be used to delimit the size of a frame. This task is therefore performed by the length field of the frame header.

6.4.3 Error control and detection

Two layers for error detection and control are used in Nanolink. The raw bitstream is protected against errors by forward error correction, whereas the Transfer Frames are secured from undetected errors using a CRC checksum. Similar to Proximity-1, different codeblock lengths and variable frame lengths are enabled by decoupling codeblocks and frames, as illustrated in Figure 6.7.

6.4.3.1 Reed-Solomon Code

The default codes for the uplink are systematic Reed-Solomon codes over $GF(256)$, with a symbol length of 8 bit. The codes are specified by codeword length n and dimension k. The codewords may be shortened arbitrarily. To be more robust against error bursts and to enable larger block sizes, codeblocks comprise one or more interleaved codewords. The number of codewords per block I is called interleaving depth. The process for

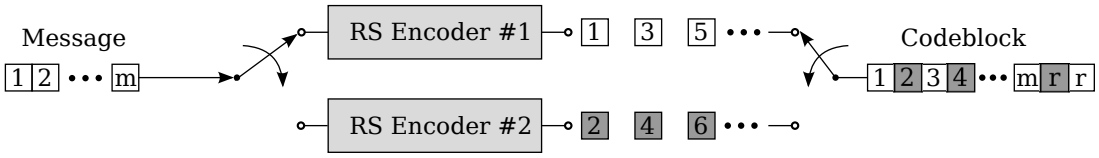


Figure 6.8: Illustration of the interleaving process for RS codes

$I = 2$ is illustrated in Figure 6.8. The m message bytes are fed into the two RS encoders on a rotating basis. After encoding, the two symbol streams are merged so that the symbols of the two encoders alternate. The message symbols are followed by the also interleaved code symbols r. To shorten codes, the message is padded with $v = I(255 - n)$ zero-symbols, which are then removed after encoding. For decoding the codeblock is prepended with the v zero-symbols.

The interleaving depth must also be chosen with respect to the ratio of uplink and downlink data rates and transmission window sizes.

6.4.3.2 Low-Density Parity-Check code

LDPC codes are the default coding scheme for Nanolink downlinks. They achieve a significant coding gain over Reed-Solomon codes, and are superior codes at low E_b/N_0 . The codes have to be chosen in combination with an appropriate decoder and with consideration of uplink and downlink data rate. Moreover, memory restrictions may render larger matrices impractical.

6.4.3.3 Code word randomization

Long runs of 0 or 1 symbols on the physical channel may cause synchronization errors at the receiver side [4]. As previously discussed in sections 4.1.2 and 4.1.5, RS codes and quasi-cyclic LDPC codes are vulnerable to codeword shifts caused by symbol slips. To mitigate these problems CCSDS recommends the use of a pseudo-randomizer generated by the polynomial $h(x) = x^8 + x^7 + x^5 + x^3 + x + 1$ [4]. The schematic of the pseudo-randomizer is depicted in Figure 6.9.

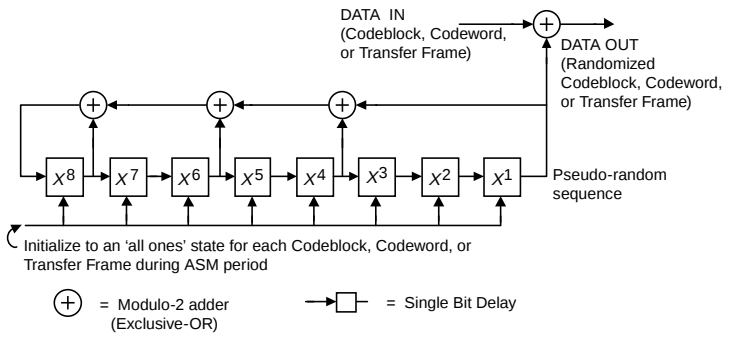


Figure 6.9: CCSDS Pseudo-randomizer logic diagram [4]

6.5 Conclusion

In this chapter, the concept of the Nanolink protocol was illustrated. The protocol fulfills the requirement for reliable communication by employing robust forward error correction and automatic repeat request schemes. In addition, all aspects of the protocol are designed with the physical link in mind. Therefore, protocol mechanics take connection losses or heavy data corruption into account. The protocol is designed in a way that the control flow reaches well defined states despite the occurrence of protocol errors. The protocol overhead was minimized by employing extension headers, which enable a rich feature set while causing no overhead if not used. The ARQ mechanism is designed to operate efficiently despite high frame error rates, while also requiring minimum bandwidth in perfect conditions.
Nanolink has a high level of automation and its operation requires no human interaction. The only exception to this are mode switches which are intended adjust the protocol operation to changing mission requirements during deployment.

Chapter 7

Evaluation

In this chapter, the performance of Nanolink is evaluated. For this purpose, its efficiency is compared to comparison to Proximity-1 in section 7.1 and section 7.2. Section 7.3 determines the maximum bandwidth-delay product at which Nanolink can operate with maximum throughput. The following evaluation is based on the assumption that the radio links are working to capacity, since efficiency is only a concern in this case. This implies that both nodes exclusively transmit data frames of maximum size.

7.1 Perfect Transmission

This section evaluates the efficiency of Nanolink and Proximity-1 in perfect transmission conditions.
The maximum length of Proximity-1 data frames is 2048 B. The frame comprises a header of 5 bytes and a data field with maximum 2043 byte. It is preceded by a ASM of 4 bytes and followed by a CRC of 4 bytes. This results in a frame overhead of 0.63%. In contrast, Nanolink has a frame overhead of 1.5%, due to the smaller maximum frame length. Proximity-1 control frames have a total length of 15 bytes, including CRC and ASM. For the same task, Nanolink only requires 3 bytes for the STAT header. For this example (Table 7.1), the downlink *sent* buffer is assumed to be 8 KiB in size. For Proximity-1, this results in a maximum window size of 4. The maximum window size for Nanolink is 16, in this case. Assuming a downlink coderate of $R = 2/3$, and a downlink data rate of 9600 bit s^{-1}, the serialization time of 8 KiB is approximately 10 s. Thus, the acknowledgment of the downlink can be assumed to require a data rate of 12 bit s^{-1} for Proximity-1. This is equivalent to 1% of the uplink data rate of 1200 bit s^{-1}. Nanolink requires only 2.4 bit s^{-1} to acknowledge the transmission window.

7.2 Retransmission Overhead

The advantage of Nanolink over Proximity-1 lies in its fault tolerance. Assuming both protocols to use the same code, Nanolink requires less codewords to encode a frame,

since the frames are smaller. The following is based on the assumption that the transmission is subject occasional burst errors (e.g. fades) that corrupt codewords. For the sake of simplicity, it is assumed that the probability for codewords to be incorrectable does not vary. A frame is erroneous if more than one codeword is incorrectable. The probability of frame errors (FER) can thus be approximated with:

$$\mathrm{FER} = 1 - (1 - p_e)^n \tag{7.1}$$

Where p_e is the probability of a codeword to be incorrectable, and n the number of codewords per frame. In this context, is assumed that codewords do not contain data of multiple frames. Table 7.2 illustrates the impact of $p_e = 0.01$ for an exemplary transmission of 100 maximum size Proximity-1 frames, or 402 Nanolink frames, respectively. For the sake of this example, both protocols are assumed to use codes with dimension $K = 173$ byte. Thus, for a Proximity-1 frame, 12 codewords are required. Nanolink frames require 3 codewords. With (7.1), this results in a FER of 0.11 for Proximity-1, and 0.29 for Nanolink frames.

Including the losses of retransmissions, 13 additional Frames have to be transmitted with Proximity-1, and 12 with Nanolink. The overhead on the forward link is 12% for Proximity-1 and 4.4% for Nanolink. These figure include the overhead of retransmissions, framing, ASM and CRC. The reverse link traffic is increased by 195 B with Proximity-1. It is assumed that Nanolink handles retransmissions entirely with NACK messages, thus requires 24 B on the reverse link.

Table 7.3 shows the situation for $p_e = 0.02$. Compared to $p_e = 0.01$, the forward overhead of Nanolink is increased by 2.9 percentage points, the reverse link traffic is increased by 108%. The overhead of Proximity-1 is increased by 10 percentage points on the forward link, and by 115% on the reverse link.

7.3 Maximum Delay

This section approximates the maximum bandwidth-delay product at which Nanolink can operate at 100% throughput. To ensure maximum throughput on the faster link, the slower link must complete transmission of an acknowledgment before the faster link has completed transmission of its window. Hence, the transmission window size is assumed to be maximal (127 frames) on the faster link. For the slower link, the minimum frame size at which user data can still be transported, is assumed. Including ASM, CRC and STAT header, the frames on the slower link are thus 12 byte in length. The minimum transmission unit size on the slower link L_U, is thus assumed to be 20 byte, including a CSM of 4 byte and FEC redundancy. The relation between window size W, faster link transmission unit size L_D, data rates R_D (faster) and R_U (slower), and propagation delay

Protocol	Proximity-1	Nanolink
Window Size	4	16
Forward Overhead	0.63%	1.5%
Reverse Traffic	12 bit s^{-1}	2.4 bit s^{-1}

Table 7.1: Overhead of Proximity-1 and Nanolink for unimpaired transmission and buffer size 8 KiB

Protocol	Proximity-1	Nanolink
Codewords per Frame	12	3
FER	0.11	0.029
Number of Frames	100	402
Retrans. Frames	13	12
Forward Overhead	12%	4.4%
Reverse Traffic	195 B	24 B

Table 7.2: Overhead of Proximity-1 and Nanolink for $p_e = 1\%$

Protocol	Proximity-1	Nanolink
Codewords per Frame	12	3
FER	0.22	0.059
Number of Frames	100	402
Retrans. Frames	28	25
Forward Overhead	22%	7.3%
Reverse Traffic	420 B	50 B

Table 7.3: Overhead of Proximity-1 and Nanolink for $p_e = 2\%$

Δt, can be roughly approximated with:

$$\frac{W \cdot L_D}{R_D} > \frac{L_U}{R_U} + 2\Delta t \tag{7.2}$$

The equation does not consider other delays, such as block delay or decoding and processing delays, or transmission errors. For the sake of this example, L_D is assumed to be 780 bytes, which corresponds to a code rate of 2/3 and a CSM size of 4 byte. Furthermore, it is assumed that the data rate ratio is 1:8. This results in a bandwidth-delay product of 49 450 B. For a data rate of 9600 bit s^{-1} on the faster link, this corresponds to a maximum propagation delay of 41.2 s or a distance of 1.24×10^{10} m between the nodes, respectively. For a propagation delay of 11 ms, this corresponds to a maximum data rate of 4.5 MB s^{-1}.

Chapter 8

Future Work

This thesis contains the concept of the Nanolink protocol. For the protocol, the next step is to formulate a comprehensive and more explicit specification. A reference implementation based on the specification must then be created, which can be used to simulate the protocol behavior and verify the specification.
For the use in MOVE 2, determining the transceiver hardware is most important. Furthermore, the hardware on which Nanolink will operate must me chosen. The protocol parameters must then be chosen to comply with these constraints. A detailed link budget for the uplink channel is required for the selection of the default Reed-Solomon code rate.
A focus of future research should lie on finding optimal codeword synchronization markers for fading channels. Also, more research on LDPC codes and corresponding decoders for fading channels is needed. For this purpose, coding experts should be consulted.

Chapter 9

Conclusion

In the past, there was no dedicated protocol stack for Cubesats that allowed efficient and reliable communication. For this reason, the Nanolink data link layer protocol was designed in this thesis.

One of the problems that had to be solved by the design of Nanolink was the combination of reliability and efficiency. The uplink channel is mainly impaired by noise. An experiment with the FUNcube-1 satellite revealed that the reliability of the downlink channel is seriously degraded by frequent signal fading. To compensate for these impairments, Nanolink messages are coded using Reed-Solomon or LDPC codes. Of a selection of common channel codes, these codes proved to be most qualified for this application.

Due to the occasional deep fades, relying solely on forward error correction was not a viable solution. Especially since the low code rate that accompanies a high correction capability would reduce the maximum achievable throughput, regardless of the channel's condition. Therefore, forward error correction was complemented by ARQ. The advantage of this is that higher code rates are possible because the additional losses are handled by retransmissions. This in return means that there is a higher upper bound to the possible throughput for better channel conditions. Overall, this hybrid solution can react to varying channel conditions more dynamically than the individual techniques and is therefore a very good choice for quickly varying fading channels, like the VHF downlink.

The primary requirement for the ARQ mechanism was reliability, meaning that it must ensure the receipt of all data. Since frequent retransmissions are part of the hybrid ARQ, they have to be performed with minimum cost. For this reason, massive losses should be recovered from without causing excessive overhead. Additionally, occasional frame losses should be handled with minimum effort. Moreover the ARQ mechanism must require only low bandwidth on the reverse channel, so that asymmetric communication is no hindrance. Finally, the mechanism should meet all these requirements while using minimal bandwidth in the case of perfect transmission.

The problem was solved in two steps: first, by adapting the ARQ mechanism from STP.

It uses a combination of cumulative selective acknowledgment and negative acknowledgment. This combination can handle single frame losses without much overhead by sending negative acknowledgments. The cumulative selective acknowledgment is perfect for acknowledging a larger span of received frames, with only few misses.
Secondly, the expenditure for acknowledgments and retransmission requests was reduced using a concept called extension headers. Extension headers is a flexible header mechanism that allows to extend frame headers simply by attaching control information. Using extension headers, a retransmission request requires only two bytes of space in a frame.
Efficiency and reliability were not the only design criteria of Nanolink. The protocol was designed for MOVE 2 and the development process and individual requirements of the satellite mission are part of the consideration.
The telemetry status broadcast mode is one of the features that originated out of these requirements. It enables reception of the satellite's status all around the world, facilitates satellite discovery and is used for connection establishment. The combination of full-duplex data transfer and simplex broadcasting modes is a unique feature of Nanolink.
Nanolink is also flexible. The mode switch mechanism from Proximity-1 allows for reconfiguration of the protocol even after deployment. This allows to adjust the satellite's COM systems to changes of mission requirements or environmental changes. This gives satellite operators more options in case of malfunctions of the satellite subsystems. This is possible because loose coupling is a core concept of Nanolink. As with Proximity-1, framing and coding concepts are decoupled, which allows variable framing without restricting the channel coding mechanisms. Thus, the choice of channel coding does not interfere with the other parts of the protocol.
The stream-oriented service of Nanolink can transport data of arbitrary format, while the packet oriented service accommodates for the Cubesat-specific protocol CSP, and thereby generates only minimal dependencies and requirements for the rest of the satellite. The extension header concept allows for a large number of additional protocol functions. Hence, it is possible to implement additional features without altering the basic design and functionality of the protocol. This renders Nanolink extensible, scalable, and ready for future developments.
Nanolink combines the advantages of STP and Proximity-1. The result is a specialized, yet flexible protocol tailor-made for the needs of the MOVE 2 mission and Cubesats in general. We hope that Nanolink can be of use not only for MOVE 2, but also for other Cubesat teams who also wish for better communication with their satellite.

Appendix

List of Tables

List of Figures

Bibliography

[1] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1. *Proceedings of ICC '93 - IEEE International Conference on Communications*, 2, 1993.

[2] Martin Bossert. *Einführung in die Nachrichtentechnik*. Oldenbourg Verlag, München, 2012.

[3] Martin Bossert. *Kanalcodierung*. Oldenbourg Verlag, München, 3rd edition, 2013.

[4] Consulative Committee for Space Data Systems (CCSDS). TM Synchronization and Channel Coding. CCSDS 131.0-B-2, Blue Book, August 2011. Issue 2.

[5] Consulative Committee for Space Data Systems (CCSDS). TM Synchronization and Channel Coding—Summary of Concept and Rationale. CCSDS 130.1-G-2, Green Book, November 2012. Issue 2.

[6] Consulative Committee for Space Data Systems (CCSDS). Proximity-1 Space Link Protocol — Data Link Layer. CCSDS 211.0-B-5, Blue Book, December 2013. Issue 5.

[7] Consulative Committee for Space Data Systems (CCSDS). Proximity-1 Space Link Protocol—Coding and Synchronization Sublayer. CCSDS 211.2-B-2, Blue Book, December 2013. Issue 2.

[8] Consulative Committee for Space Data Systems (CCSDS). Proximity-1 Space Link Protocol—Rationale, Architecture, and Scenarios. CCSDS 219.0-G-2, Green Book, December 2013. Issue 2.

[9] Cubesat Design Specification. `http://www.cubesat.org/images/developers/cds_rev13_final.pdf`, 2014. Revision 13.

[10] P Elias. Error-free Coding. *Information Theory, Transactions of the IRE Professional Group on*, 4(4):29–37, September 1954.

[11] G. David Forney and Daniel J. Costello. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95:1150–1177, 2007.

[12] Funcube Dashboard Notes. http://download.funcube.org.uk/FUNcube_Dashboard_Notes_Release_1-6.pdf, 2014. Accessed: 31.10.2014.

[13] Robert G Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, Cambridge, Massachusetts, 1963.

[14] Alain Glavieux, editor. *Channel Coding*. ISTE Ltd., 1st ed. edition, 2007.

[15] Gomspace. *CubeSat Space Protocol*, 2011. User Manual.

[16] Jon Hamkins. Frame Synchronization Without Attached Sync Markers. In *IEEE Aerospace Conference Proceedings, 2011 IEEE*, pages 1–7, March 2011.

[17] Thomas R. Henderson and Randy H. Katy. Satellite Transport Protocol (STP): An SSCOP-based Transport Protocol for Datagram Satellite Networks. In *2nd International Workshop on Satellite-based Information Services*, Berkeley, 1997. University of California in Berkeley.

[18] International Telecommunication Union. *Handbook on Satellite Communications*. John Wiley & Sons Ltd., 3rd ed. edition, 2002.

[19] International Telecommunication Union. Characterization of the variability of propagation phenomena and estimation of the risk associated with propagation margin. ITU-R P.678-2, September 2013.

[20] International Telecommunication Union. Ionospheric propagation data and prediction methods required for the design of satellite services and systems. ITU-R P.513-12, September 2013.

[21] International Telecommunication Union. Propagation data and prediction methods required for the design of Earth-space telecommunication systems. ITU-R P.618-11, September 2013.

[22] Louis J. Jr. Ippolito. *Satellite Communications Systems Engineering*. John Wiley & Sons Ltd., Chichester, West Sussex, United Kingdom, 2008.

[23] Alexander Kaiser, Sam Dolinar, and Michael K. Cheng. Undetected errors in quasi-cyclic LDPC codes caused by receiver symbol slips. In *GLOBECOM - IEEE Global Telecommunications Conference*, 2009.

[24] Karl-Dirk Kammermeyer. *Nachrichtenübertragung*. Vieweg+Teubner Verlag, Wiesbaden, 4 edition, 2008.

[25] A Kopansky and M Bystrom. Frame synchronization for noncoherent demodulation on flat fading channels. In *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, volume 1, pages 312–316 vol.1, 2000.

[26] Richard Limebear. The AMSAT-UK FUNcube Handbook. http://funcubetest2.files.wordpress.com/2010/11/funcube-handbook-en_v13.pdf, 2013. Accessed: 31.10.2014.

[27] D J C MacKay and R M Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 33(6):457–458, March 1997.

[28] James Miller. Oscar-40 FEC Telemetry. http://www.amsat.org/amsat/articles/g3ruh/125.html, 2003. Accessed: 31.10.2014.

[29] Online Tracking of FUNCUBE-1. http://www.n2yo.com/satellite/?s=39417. Accessed: 14.09.2014.

[30] John Proakis and Masoud Salehi. *Communication Systems Engineering*. Pearson Education, Upper Saddle River, New Jersey, 2002.

[31] Patrick Robertson. Maximum likelihood frame synchronization for flat fading channels. In *Communications, 1992. ICC '92, Conference record, SUPERCOMM/ICC '92, Discovering a New World of Communications., IEEE International Conference on*, pages 1426–1430. IEEE Computer Society, 1992.

[32] YURDANUR K Tulunay and Peter A Bradley. The impact of space weather on communication. *Annals of Geophysics*, 47(2-3 Sup.), 2004.

[33] David Velten, Robert Hinden, and Jack Sax. Reliable Data Protocol, 1984. RFC 908.

[34] Stephen B. Wicker and Vijay Bhargava, editors. *Reed-Solomon Codes*. IEEE Press, New York, New York, USA, 1994.