Foresail-1 & Foresail-1p

Skylink Protocol Specification

PUBLIC DOCUMENT

Aalto University

Authors

Name	Worked with parts
Petri Niemelä	
Markus Hiltunen	
Klaus Kivirikko	**Updates for Foresail-1p**

Table of Contents

1 General

1.1 Scope

This document describes the over-the-air protocol used on the Foresail-1 and Foresail-1p UHF telecommanding and telemetry link. The protocol is open source under LGPL-3.0 License, and publicly available at github.com/aaltosatellite/skylink. The protocol definition includes the physical and link layer definitions which are commonly understood in the standard telecommunication OSI-model. The physical layer specification defines how the radio medium is used for transferring packetized data. This includes used synchronization markers, framing format and Forward Error coding. The link layer specification includes TBU

1.2 Referenced documents

RD-01	Texas Instrument’s CC1125 user’s guide
RD-02	Phil Karn, libfec-library: http://www.ka9q.net/code/fec/

1.3 Applicable documents

AD-01	..

1.4 Abbreviations and terminology

AOS	Acquisition Of Signal
ARQ	Automatic Repeat Request (retransmission)
ASM	Attached Sync Marker

A Byte	A group of 8 information bits.
FEC	Forward Error Correction
FSK	Frequency Shift Keying
Fragment	Piece of inside LLC frame
LLC	Logical Link (layer) Control
LOS	Loss Of Signal
MAC	Medium/Media Access Control
HMAC	Hash Message Authentication
MSK	Minimum Shift Keying
An octet	A group of 8 symbols (traditional terminology for telecommunication field)
PDU	Protocol Data Unit
	For a given protocol layer, the packets exchanged with the lower layer.
PS	Packet Stream
RS	Reed-Solomon error correction coding
SDU	Service Data Unit.
	For a given protocol layer, the packets exchanged with the upper layer.
VC	Virtual Channel

Virtual Channel	A logical data transfer channel built on top of the physical radio channel

Node	An active transceiver device implementing the protocol
Transmitter	A device capable to transmit specified physical frames
Receiver	A device capable to receive specified physical frames
Transceiver	A device capable to transmit and receive specified physical frames

2 Background

This section gives general rationale why such protocol was developed for the

2.1 The purpose of the protocol

The protocol stack aims to implement:

1. Reliable transfer of commands and efficient data transfer over a point-to-point, half-duplex radio link with limited capacity and randomly occurring lost frames.
2. Satellite-to-ground telemetry broadcasting

Packetized link. No streaming support.

The protocol implements four Virtual Channels (VC). Each channel can be configured to be reliable. When the channel is configured as unreliable it can work as a beacon/broadcast channel.

Rationale: In our use case, lost frames may be caused by fading resulting from tumbling of a spacecraft, by interference bursts from other users of the same frequency band, or by a marginal signal-to-noise ratio. Capacity is limited by the combination of bandwidth (limited by licensing), transmit power (limited by power budget and hardware design) and features of the hardware used to implement the radio link (particularly the choice of modulations available). High throughput file transfer in spite of these challenges is required by the mission.

2.2 Provided services

Physical layer: The physical layer protocol provides a packetized transfer over the radio medium. The layer allows transfer of variable-length application layer packets for point-to-point and broadcasting. (Figure 1) The layer implements forward error correction coding methods to achieve robust data transfer over varying channel quality. The layer also ensures the integrity of transmitted data: the data will be transferred as it is over the link and any corruptions can be either corrected using the FEC or detected reliable. The layer defines a group of data rates which can be usable depending on the higher level use cases.

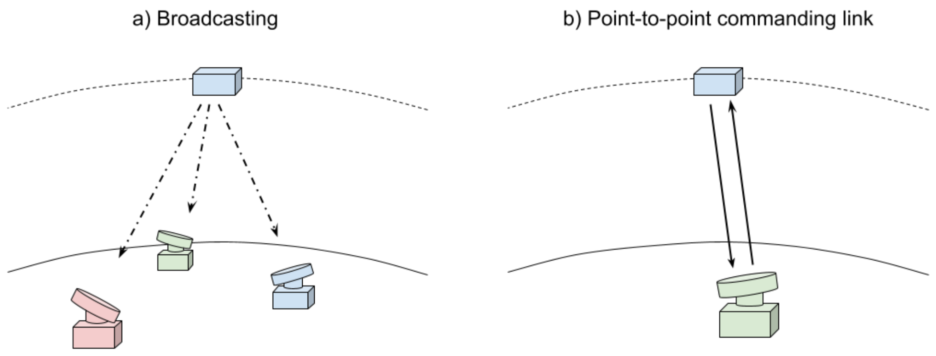


Figure 1: The skylink protocol has been designed for two use cases scenarios. a) Data broadcasting from the satellite to multiple ground stations and b) point-to-point data transfer (commanding) between the satellite and single ground station. While two ground stations could in principle have point-to-point connection to the satellite at the same time, medium access is not designed to facilitate this, and would result in loss of synch. .

Medium access: The medium access control layer provides an efficient way to share a single radio frequency allocation. Designed to take in consideration the nature of the space-to-ground communication:

asymmetric link usage, the highly varying link quality (loss due to fading), connection interruptions due to orbital mechanics and long propagation delays (one way delay 2–6 ms).

Hash message authentication: The data link implements optional message authentication. This is done with a shared secret (32 random generated bytes) known to the groundstation and the satellite. Each message is authenticated by appending the message to the secret, obtaining a hash digest of this combined bytestring using a cryptographic hash algorithm (AES2), and appending the first 4 bytes of the digest to the final message as authenticating signature.

Each message carries a strictly increasing hmac-sequence number (with period 65536) to ensure messages with otherwise identical content do not obtain the same authentication signature, and thus protects from repeat attacks. Either skylink peer can enforce the hmac-sequence they expect to receive, but not the sequence they should send.

There is a configurable number HMAC allows the peer to jump forwards in sequence, as HMAC is not intended to guard against message loss.

Data link layer: The data link layer implements higher level over the physical level link to serve various satellite level applications. The layer implements four virtual channels over a single radio channel (Figure 2). A virtual channel may be configured either as unreliable (unacknowledged) or reliable (acknowledged)[1]. When the channel has been configured to be unreliable it behaves as like a broadcasting channel which can have one or multiple listeners, but

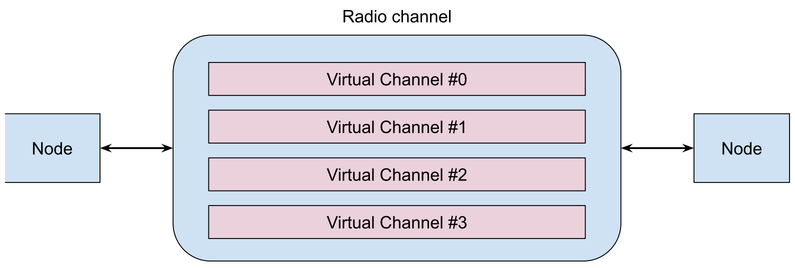


Figure 2: Multiplexing of four virtual channels (VC) over the shared radio channel between the nodes.

, lost packets are not automatically retransmitted. When a channel has been configured reliable, VC employs message numbering and automatic resend requests (ARQ) and provides messages to be read only in sequential order without missing messages, or indicates that the reliable state has been broken. The timeout parameter for ARQ to consider itself broken is configurable.

Packets within each virtual channel are delivered in order and without errors.

[1] Reliability/unreliability is a feature of a channel. Acknowledging is a method for achieving reliability.

3 Protocol Overview

The protocol stack is split into multiple layers and their sub-layers. See figure 3 for an overview.

Figure 3: Illustration of the (sub)layers and their connections in the protocol stack.

Rationale: Each sub-layer can be implemented and tested separately, at least to some degree. This helps to manage complexity, since each sub-layer is relatively simple by itself compared to the whole protocol stack.
A notable exception to the independence of protocol layers is between HMAC and MAC. Medium access control (MAC) attempts to prevent crosstalk over the channel. This includes updates to local belief of own transmission window, according to MAC metadata in received frames. This would permit a malicious actor to postpone peer transmission infinitely with continuous packet transmission. To prevent this, MAC can be configured to update its belief only with authenticated messages. This represents a dependence between protocol layers.

4 Physical layer (PHY)

This section describes the physical layer protocol: Modulation scheme, Forward error correction coding

4.1 Modulation

The used modulation scheme is 2-level Frequency-Shift Keying (FSK) with a modulation index of 0.5 and Gaussian filtering (GFSK) with a bandwidth-time product of 0.5. This modulation can also be interpreted as Gaussian Minimum Shift Keying (GMSK) with differential phase coding. The lower frequency represents a ‘0’ symbol and the higher frequency represents a ‘1’ symbol. When the signal is interpreted as GMSK on the receiver higher reception sensitivity is possible to achieve with proper demodulation implementation.

Table 1:Recommended modulation parameters

Parameter	**Value/details**
Modulation scheme	*GFSK/GMSK (**TODO BPSK)***
Modulation index (for GFSK)	*0.5*

Bandwidth time product	*0.5*

Note: If 2-GFSK modulation is used to create GMSK signal (like in integrated radio transceiver chips) the transceiver's frequency deviation shall be set to ¼ of symbol rate! For 9.6 kbps the configured deviation is 2.4 kHz. In the literature, the modulation index is usually expressed by the difference of the peak frequencies (sometimes also referred as deviation) and transceivers are configured deviations from the center frequency.

4.2 Symbol rate

Uplink symbol rate:
- 9 600 symbols/s
- 19 200 symbols/s
- 38 400 symbols/s

Downlink symbol rates
- 9 600 symbols/s
- 19 200 symbols/s
- 38 400 symbols/s

For each symbol rate, there has to be a

Rationale: The 9600 bits/s FSK is a so-called legacy which has been designed to work (barely) over 16 kHz radio amateur FM-rigs so that the FM demodulated signal is output as voice. In this protocol definition, backward compatibility for these old HAM rigs is not seen as important and a wider bandwidth received (such as any SDR) is required.

4.3 Forward error correction

The payload is encoded with a systematic Reed-Solomon (255,223) code with 8-bit symbols. RS(255,223) can detect up to 32 byte errors and correct up to 16 bytes. The generator polynomial and representation of the code is similar to the CCSDS standard Reed-Solomon code except that the conventional basis symbol representation is used rather than dual basis representation. [ref]

No convolutional coding. (elaborate...)

Implementation: For a reference implementation of the code, see the functions *encode_rs_8* and *decode_rs_8* in libfec [RD-xx].

Rationale: Reed-Solomon code is good at correcting bit errors which are bursty in their nature (long continuous sequences of errors), since multiple closeby bit errors likely fall within only one or two Reed-Solomon code symbols. If the GMSK modulation used by the physical layer is demodulated coherently as differentially encoded OQPSK, the differential encoding causes most errors to occur in bursts of two successive corrupted bits. In ⅞ of cases, they will still corrupt only one Reed-Solomon symbol, reducing the performance penalty from differential coding. The particular code was chosen for availability of existing implementations, such as libfec. The same code has also been used in other missions, such as Suomi-100.

Calculating RS codes which symbol length is 8-bits is .
With the given implementation encoding rate of 1839.9~kbit/s and decoding rate of 570.3~kbits/s were measured when coding full length messages on ARM Cortex-M0 running at 40~MHz.

Low Density Parity Codes (LDPC) are too heavy for MCUs and would require an FPGA hardware accelerator or such.

Reed-Solomon code can ensure the data integrity up to 32 byte error and doesn't not necessarily require additional inner checksum for detecting bit errors but they are possible. However, if HMAC is also utilized it will filter out the rest of corrupted frames with very high confidence.

4.4 Framing

A frame is variable length
A physical layer frame consists of the following fields(transmitted in order):

Table 2: The field structure of physical layer frame

Field	**Preamble**	**Syncword**	**PHY header**	**Data Frame**	**FEC**	**Postamble**
Length (modulated symbols)	48	32	24	N*8 (max 223 bytes)	32*8	0

Detailed description of each field:

- **Preamble** is a repeating pattern of alternating '0' and '1' symbols used to detect the frame's bit synchronization.
- **Syncword** is a fixed symbol pattern used to detect the byte synchronization.

- **PHY header** contains the required information, incl. length information, for decoding the payload section. Encoded with Golay FEC code.
- **Data Frame** is the Reed-Solomon codeword with octets transmitted in order, with whitening. The payload contains both the carried frame and the FEC. The structure of this field is described later in Sections 5 and 6. It always begins with a data header of 11 bytes..
    - If the FEC is disable the maximum length of the field is 256 bytes.
- **FEC:** Reed Solomon syndrome bytes
- **Postamble** is an optional field after payload.

Rationale: A similar preamble–syncword–length-word–payload style of frame format (sometimes referred as IEEE 802.15.4) is supported by a large variety of integrated radio transceiver chips, including the CC1125 [RD-01] used in FORESAIL-1. It is used by several other systems as well, including Aalto-1 and Suomi-100.

Table: Frame lengths in time using various symbol rates

	9600 bit/s	**19200 bit/s**	**38400 bit/s**
Minimum frame 11 data bytes	40 ms	20 ms	10 ms
Maximum frame (223 data bytes)	220 ms	110 ms	55 ms

4.4.1 Preamble

The preamble is a repeating pattern of alternating ‘0’ and ‘1’ symbols (like 0101010101…) used to detect the frame’s bit synchronization. The preamble pattern is 0xAA and it shall be at least 6 bytes (48 symbols) long. Thus, at 9600 symbols/s the duration of the preamble is 5 ms.The preamble is a repeating pattern of alternating ‘0’ and ‘1’ symbols (like 0101010101…) used to detect the frame’s bit synchronization. The preamble pattern is 0xAA and it shall be at least 6 bytes (48 symbols) long. Thus, at 9600 symbols/s the duration of the preamble is 5 ms.

Table 2: Preamble durations for different data rate and byte lengths

Data rate (kbit/s)	**Preamble length (bytes)**	**Preamble duration (ms)**
9 600	6	~5
9 600	30	25
19 200	6	~2.5
19 200	30	12.5

Rationale: CC1125 can generate from ½ to 30 bytes long preamble depending on the configuration. The preamble can be configured to be 0xAA, 0x55, 0x33 or 0xCC. A receiver may use the preamble to estimate frequency offset and symbol timing.
Transmit power is ramped up during the first (TBD) preamble symbols.

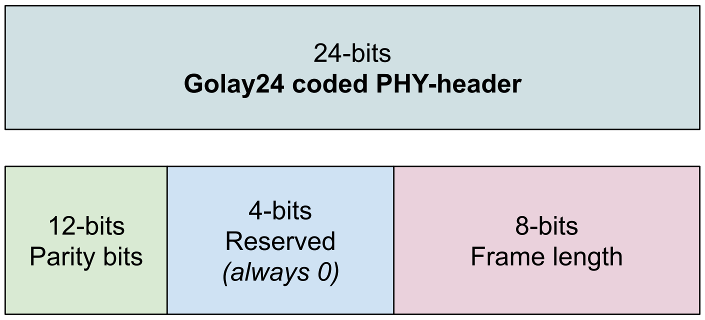
4.4.2 Syncword

The syncword is a 4 bytes long unique byte string used to detect the beginning of the frame and establish the byte synchronization. At the moment the used sync word is `0x1A 0xCF 0xFC 0x1D`. This sync word was picked from the CCSDS documentation (https://public.ccsds.org/Lists/CCSDS%201420R1/142x0r1.pdf).

Rationale: The used syncword cannot be any random byte string. For the good syncword its autocorrelation should be as small as possible. Modems supporting GOMSpace radio systems are flexible on the syncword definition.

4.4.3 PHY header

The PHY header field contains information required for reception of the payload section. This information includes. The length of the header field is 24 bits (3 bytes) and it can contain 12-bits of information. 8 least significant bits of the coded field contain the length of the payload section (Data + FEC). The four most significant bits are not used and shall be set to 0.



Implementation: Golay(24,12)...
SatNOGS' Golay implementation:
GR-Satellite implementation

Rationale: By the IEEE 802.15.4, the length information contained in the header is an uncoded byte but a decision to use FEC coded length was seen as important for robust reception. With uncoded length byte a single bit error in the length bytes section can corrupt the whole frame. The length byte must use a separate error coding method from the payload section because the length information is needed before the decoding of the payload section can start.

The Golay (24, 12) code is also used in GOMSpace AX-series radios and is known as Mode 5. This specification adapts GOMSpace's mode in many ways and is designed to be receivable using existing modem designs. In older GOMSpace's U482C radios, the three most significant bits were used as flags to indicate the presence of scrambler, Reed-Solomon and convolutional coding. This convention was dropped in GOMSpace's AX100 radio which is much more popular. Also implementing these flags behavior is complicated. 1) Randomizer enable flag should be known before reception phy header. 2) Viterbi decoder synchronization is a bit hacky when you first receive phy header without convolutional coding and then turn it on in middle of reception. Especially if a soft bit decoder is used. More about the compatibility with GOMSpace's mode can be read from the Appendix A.

4.4.4 Payload

The payload field contains the actual data conveyed by the PHY frame. The maximum length of the payload information is 223 bytes (RS message length). After the payload information there are 32 bytes of Reed-Solomon

4.4.4 Postamble

TBD
During the postamble the transmit power is ramped down.

Rationale: The postamble may help some receiver implementations (such as a coherent demodulator with feed-forward carrier phase recovery) which have a higher error rate for the last few symbols if power is immediately cut.

4.5 Data whitening

Data whitening or so called randomization or scrambling is recommended to be used. The existence of the whitening is indicated in the high bits of the PHY header. The whitening is applied to the data frame and FEC. The used scrambling algorithm is the CCSDS pseudo random. The pseudo random follows the polynomial: $h(x) = x^8 + x^7 + x^5 + x^3 + 1$. The implementation for the algorithm can be found in the Appendix C.

The CC1125 transceiver chip is capable of automatic data whitening if configured so. The used data whitening algorithm is Texas Instrument's own pseudo random algorithm called PN9 (Pseudorandom Noise 9). Reference: Texas Instruments Design Note 905: Data Whitening and Random TX Mode

Rationale: Data whitening algorithm is used to scramble the transmitted bits so that the bitstream is close to white gaussian noise. The benefits of using data whitening are:

1. This removes the constant DC-bias from the FSK.
2. Helps adaptive bit slicer algorithms (PLL, Mueller&Mueller etc.) to maintain the bit synchronization during the frame.
3. Less change to have a syncword inside the frame. [ref CCSDS]
4. Less spectral lines [ref CCSDS]

5 Data Frame

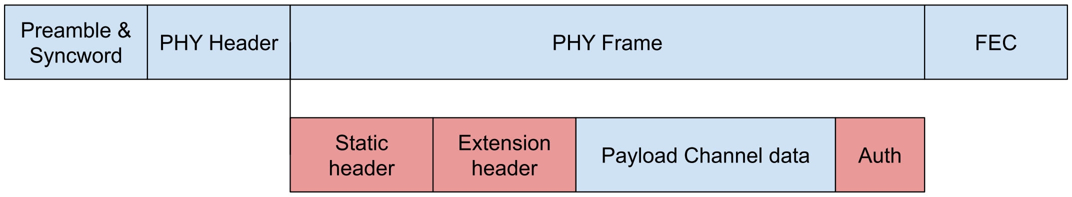


Figure: Overall physical frame structure

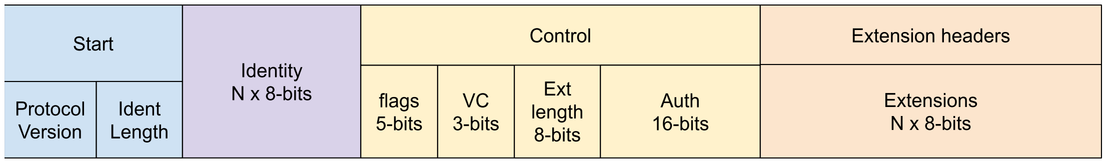


Figure: Skylink header (static and extension) structure

Protocol Version					**Identity Length**		
0	1	1	0	0	1	1	0

Identity: (preferably a HAM callsign)

Extension headers

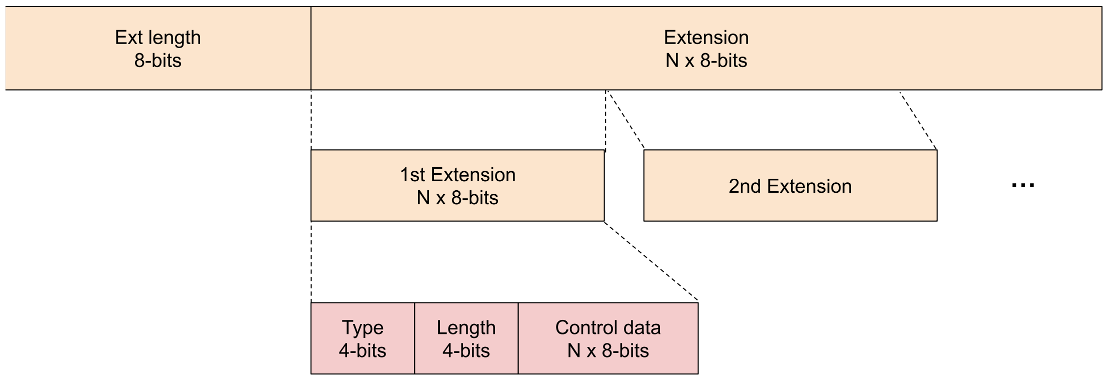


Figure: Extension header field structure

The used header extensions defined by this protocol version are defined in the following table. The detailed structures and uses of the

Table: Extension header types

ID	Name /mnemonic	Length	Description
0	ARQ Sequence	1	ARQ sequence number
1	ARQ Retransmit Request	3	Contains the current ARQ sequence number and bit mask of received sequence numbers.
2	ARQ Control	2	ARQ Idle Control header
3	ARQ Handshake	5	ARQ Connection handshake
4	TDD Control	4	
5	HMAC Reset	2	Control response from HMAC logic. The control data field contains the

Table: ARQ Sequence (ID: 1) extension

Extension type 4-bits	Extension length 4-bits	ARQ Sequence Number 16-bits
0x4	0x1	0x??

Table: ARQ Retransmit Request (ID: 2) extension

Extension type 4-bits	Extension length 4-bits	ARQ Sequence 16-bits	Received Mask 16-bits

0x4	0x3	0x??	0b???? ???? ???? ????

Table: ARQ Control (ID: 3) extension

Extension type 4-bits	**Extension length 4-bits**	**ARQ TX Sequence 16-bits**	**ARQ RX Sequence 16-bits**
0x4	0x3	0x??	0x??

Table: ARQ Handshake (ID: 4) extension

Extension type 4-bits	**Extension length 4-bits**	**Peer ARQ state 8-bits**	**Session identifier 32-bits**
0x4	0x3	0x??	0x??

Table: HMAC Sequence Reset (ID: 5) extension

Extension type 4-bits	**Extension length 4-bits**	**HMAC Sequence Number 16-bits**
0x4	0x2	0x?? 0x??

5 Medium Access Control layer (MAC)

The Medium Access Control (MAC) layer’s purpose is to describe how the operation on the shared radio medium is controlled. The Skylink protocol is designed to operate on single frequency allocation using Time Division Duplexing (TDD). In TDD, the usable transmission time is divided in time slots which are used by the link node in coordination.

If the TDD has been configured, each frame must have an extension header with

5.1 MAC Header

The length of each time slot is determined by protocol configuration parameters. TBD.

The TDD logic also defines how much forehand preparation can be started.

Table: TDD Control (ID: 5) extension

Extension type 4-bits	**Extension length 4-bits**	**TDD Window 16-bits**	**TDD Window remaining 16-bits**
0x4	0x3	0x????	0x????

Rationale: Including the direction in the header avoids problems arising if a station receives its own transmissions and thus echoes them back to the link layer.

The repeated bits don’t appear very useful since the frame is already protected by Reed-Solomon coding, but they may help an advanced implementation to track TDD synchronization even using frames for which Reed-Solomon decoding failed. Other values for these bits are also reserved for future extensions.

5.2 Time Division Duplexing

Terminology:

Transmission slot (TS): a fixed-length time segment, the building block for the protocol time synchronization. In one transmission slot either none, one or multiple packets can be transmitted.

Transmission window (TW): a series of transmission slots when one side is transmitting.

Round trip time (RTT): time between the end of the transmission and the start of the response to it (including the other side switching delay)

To make efficient use of a half-duplex radio link, time division duplexing (TDD) is used. Each end of the link has its own time window when it is allowed to transmit. One of the ends is called a master and the other is called a slave. For space-to-ground link, the ground segment is defined to be the master and the satellite is defined to be the slave.

Both master and slave use the same timing loop and in the current version of the protocol this distinction is made only to differentiate Rx and Tx messages on every node in case of receiving their own message.

The TDD timings are specified based on the following parameters:

1. Transmission slot length (in milliseconds)
2. Ground window length (current and maximal) (in TSs)
3. Space window length (current and maximal) (in TSs)
4. Estimated RTT
5. Switching delay

These parameters are controlled through the Skylink's configuration system.

When the own window is closed:

$T_{new} = T_{now} + 2 * T_{RTT} + (N_{slots} + 1) * T_{slot} + T_{slack}$

(This estimated shall be pessimistic and will improve when frames are received)

Update routine when a new frame is successfully received:

$T_{new} = \min(T_{old}, T_{now} + T_{slack} + T_{slot} * N_{slot})$

When preamble is received:

$T_{new} = \max(T_{old}, T_{now} + T_{fallback})$

The TDD functionality is built based on following loop:

1. Switching to transmission and evaluation of the following transmission window length (described in Window length calculation)
2. Transmitting for evaluated amount of slots
3. Switching to the receiving mode and waiting until:
   a. $Estimated\ RTT + Maximum\ TW\ length\ of\ the\ other\ party$passes, in which case loop immediately goes back to 1)
   b. A packet received, in which case transition to 1) happens after $TS\ length * remaining\ slots$time passes. In case other packets are received, this time is updated again according to their $remaining\ slots$field

Window length calculation:

To determine the next transmission window length, the following set of rules is used:

1. On the first transmission after any parameter change the length equals to $Maximum\ TW\ length$

2. If there are less packets in the buffer then there are slots in a window, decrease the window size by one, if more - increase by one
3. Window size should be more than or equal to one and less than or equal to $Maximum\,TW\,length$

Examples of corner cases operation:

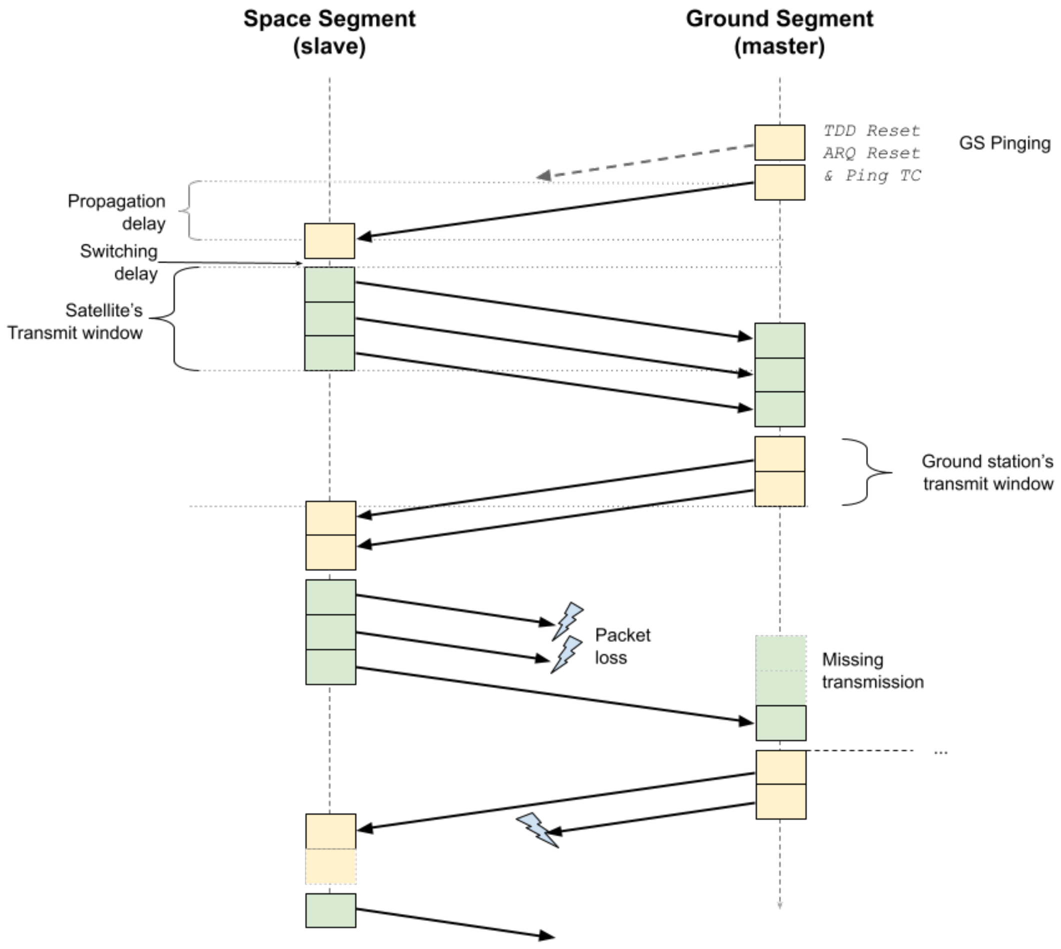


Figure 4: TDD diagram

AOS: Acquisition of Signal

During this out-of-sync phase the satellite will be broadcasting using .
TODO: Collision avoidance

"Joining" behavior where a ground station would join to first by is not defined by this specification. The

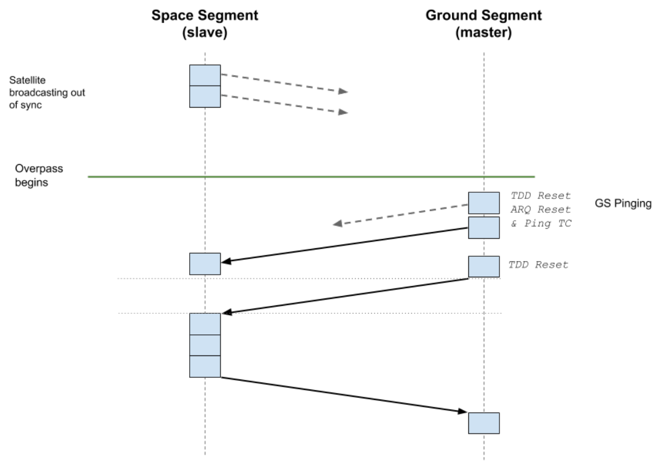


Figure: TDD timing at Acquisition of Signal (AOS)

LOS: Loss of Signal

TODO: Something in the higher level (in syncword?) should indicate is MAC header existing.

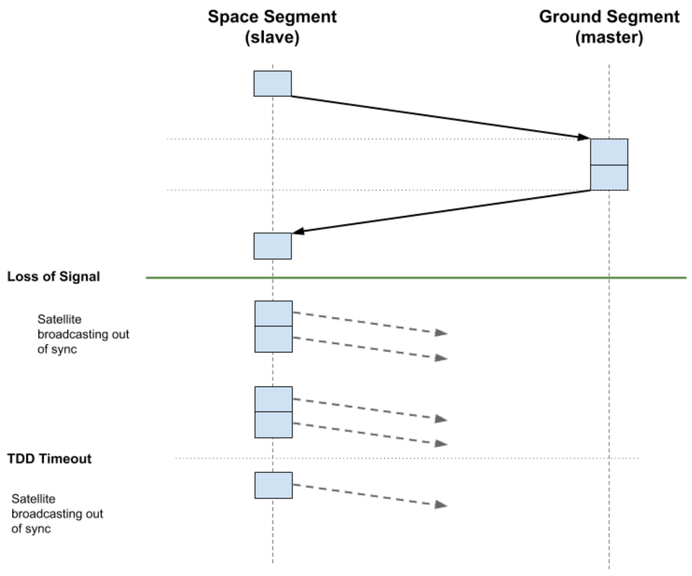


Figure: TDD timing at Acquisition of Signal (LOS)

The TDD is designed to work with 1 ms timing resolution. The smallest frame length considered by the PHY layer is around 20 ms.

How the TDD variables are configured?

5.3 Collision avoidance

Additionally to the TDD logic, collision avoidance is implemented with a simple fallback of the windowing when the carrier is sensed.

6 Data Link Layer

The Data Link Layer defines how the is transported over the link provided by the Physical Link layer (PHY).

IDEA: Config to have only data link frames from a single virtual channel in a physical link layer frame.

6.1 Overview

The data link layer
The link layer also implements a logical connection (session) between two transceivers.
Inside a single Physical Layer frame, there can be multiple of data link layer **fragments**. The number of fragments can be 1 to K till the maximum lengths of the PHY frame (223 bytes) is reached.

(Service expected from the physical layer)
The link layer **assumes** that the physical layer delivers only correct frames, i.e. erroneous frames shall be dropped before they reach the link layer. It also assumes that frames are delivered in the order in which they have been transmitted.

- Version and type 8-bits
  - T
- APID 16-bits
  -
- Flags
  - Is authenticated -flag
- Sequence control 16-bit

- 2-bit for VC?
- 14-bits for frame

Table 4: Information field conveyed in the payload of the physical layer frame

Field	**MAC/TDD header**	**Fragment PDU**	**Fragment PDU**
Length	1 byte	*MUX PDU format + data*	
Description			

The lowest 2-bits (?) of the header indicate the virtual channel

Table 5: Example of virtual channel usage

Virtual Channel	**Name**	**Description**
VC0	Commanding	Used for OBC commanding and command acknowledgements. Highest priority VC.
VC1	Mass data	Used by OBC mass telemetry generators such as housekeeping beacons and scheduled downlinks.
VC2	Service Channel	Used for UHF, EPS & direct avionics bus commanding and diagnostics.
VC3	HAM	Used for radio amateur repeater

Structure
All acknowledged virtual channels are multiplexed together into one ARQ process. Then, unacknowledged virtual channels, ARQ PDUs and the reverse channel of the received ARQ process are multiplexed together to become LLC PDUs. This structure is also illustrated in Figure 1.

Rationale: For better control of prioritization, a separate ARQ process could be used for each acknowledged virtual channel. This, however, would increase the protocol overhead due to additional reverse channels and headers, and might also increase memory requirements of an implementation.

6.2 Multiplexing sublayer (MUX)

MUX handles fragmentation and multiplexing of SDUs from multiple virtual channels. A single MUX PDU may contain data from multiple SDUs. If fragmentation is enabled, data of a single SDU may be fragmented into multiple PDUs.

When it's possible to transmit a frame, the underlying lower layer requests a MUX PDU with a given maximum length. MUX fills it up to the maximum length with fragments of upper layer SDUs. If the lower layer supports only fixed length frames, the end of a MUX PDU may be padded with zeros.

MUX may apply priorities to choose which channels to send. Fragments from multiple channels may be interleaved, allowing, for example, sending of high-priority packet in the middle of sending a long, fragmented low-priority packet.

MUX PDU consists of concatenated structures with the following format:

Table 6: MUX PDU format

Position		Name	Details
Octet 0	Bits 7-5	Function (0 - 7)	~~0 reserved for zero padding~~
	Bits 4-3	Virtual channel (0 - 4)	Virtual channel number
	Bits 2-1	Framing info	00 = Full SDU 01 = First fragment of an SDU (more data to come) 10 = Last fragment of a fragmented SDU 11 = Middle fragment of an SDU (more data to come)
	Bit 0	(Variable) length bit	0 = Fragment continues either until end of the frame or length is constant specified by the function code. 1 = Next octet determines the length of the fragment
Octet 1 (if length bit=1)		Length (N)	Length of the fragment in octets

Octets 2 to N+1 (if length bit=1) OR Octets 1 to M (if length bit=0)	Data	Data of the SDU fragment

Table 7:

ID	Mnemonic	Function
0	IDLE	
	ARQ RST	
	ARQ RET	

Rationale:

6.2 Automatic Repeat Request sublayer (ARQ)

~~Each packet stream inside can implement data transfer reliability using automatic retransmission.~~

~~Additionally to the physical layer FEC, the Link Layer Control layer implements Automatic Retransmission which can be understood as error correction method. This is sometimes called as hybrid FEC.~~

The automatic repeat-request (ARQ) uses so called **selective-reject automatic repeat request.** ARQ operates with a sliding window with a fixed size of 16. For a sender, the window consists of the latest 16 SDUs from the upper layer, also called packets here in the context of ARQ. For a receiver, the window consists of packets that have been received but haven't been delivered to the upper layer yet.

Packets shall be delivered to the upper layer in order and without missing packets. When a receiver receives a new packet, it places it in its buffer. Whenever a contiguous sequence of packets is available in the buffer, the ARQ receiver delivers them to the upper layer, removes them from its buffer and moves the start of its window to make room for new received packets.

Acknowledgements sent in the reverse channel work by always sending the latest sequence number up to which all packets have been successfully received, together with a bitmap indicating which packets following that have been received. Thus, when the sender receives the latest reverse channel packet from the receiver, it knows the current state of the receiver even if reverse channel packets have been lost in between.

Since lower layers are not allowed to reorder ARQ PDUs, an increase in sequence number of more than 1 between successive received PDUs indicates a lost PDU. This is signaled back to the sender in the bitmap in the reverse channel, allowing the sender to retransmit lost PDUs.

ARQ exchanges SDUs with an upper multiplexing sublayer.

Table 8: ARQ PDU format

Position	**Content**	**Details**
Octet 0 Bits 7-6	Frame type or command	00 = Reserved 01 = Normal frame 10 = Protocol reset command 11 = Reserved
Octet 0 Bits 5-0	Sequence number	
Octets 1-N	Data	ARQ SDU, i.e. upper MUX PDU

Table 9: ARQ reverse channel PDU format

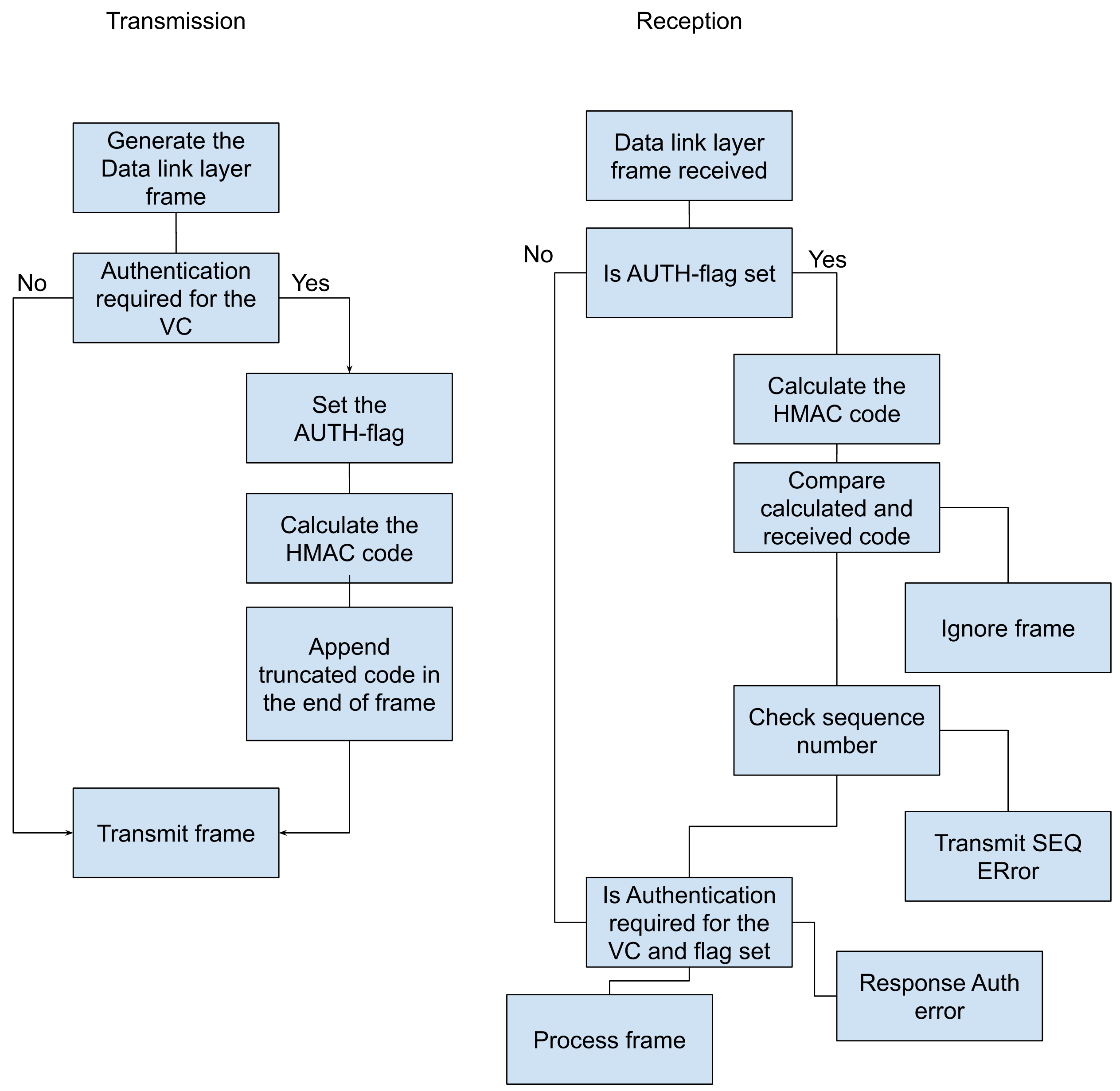
Position	**Content**
Octet 0 Bits 7-6	ARQ Receiver status 00 = Reserved 01 = Normal operation 10 = Receiver in reset state, 11 = Receiver is confused
Octet 0 Bits 5-0	Latest contiguously received sequence number. Receiver window starts from the sequence number following this.
Octets 1-2	Bitmap of successfully received frames in window

7 Application layer

7.1 Authentication

Authentication can be enabled for a given virtual channel. The authentication is designed to ensure that only authorized parties can operate the channel over the commanding virtual channels without encrypting the content of the packet. The data link layer frame header contains a flag whether the frame is authenticated or not. If the flag is set, an authentication code check is performed for the frame. The authentication code is a 4 byte long trailer section which can be appended to the end of the data link frame. The code is so-called Hash-based Message Authentication Code (HMAC) and it’s calculated using a cryptographic hash function. This method is known as: Hash-based Message Authentication Code. The used Hash-function is SHA-256. The output hash is truncated to length of 4 bytes to reduce the overhead.

A frame can be processed as an unauthenticated frame and later virtual channel.


Transmission
Reception
Generate the Data link layer frame
Authentication required for the VC
No
Yes
Set the AUTH-flag
Calculate the HMAC code
Append truncated code in the end of frame
Transmit frame
Data link layer frame received
Is AUTH-flag set
No
Yes
Calculate the HMAC code
Compare calculated and received code
Ignore frame
Check sequence number
Transmit SEQ ERror
Is Authentication required for the VC and flag set
Response Auth error
Process frame

TODO: Running sequence numbering somewhere (?) prevents packet repetition attacks.

Implementation: The implementation for the authentication is greatly based on Cifra library (https://github.com/ctz/cifra) which includes a great collection of cryptographic algorithm implementation for embedded systems.

File transfer

To be expanded.

Files must be split to blocks

Appendix A: GOMSpace AX.100 Mode 5 compatibility

The physical layer protocol is compatible with GOMSpace's AX.100 Mode 5 with following settings:

- No HMAC
- No CRC
- RS enabled
- No randomizer
- Sync: 0x1ACFFC1D, 0xC3AA6655

More about the GOMSpace's frame implementation:
https://bytebucket.org/bbruner0/albertasat-on-board-computer/wiki/1.%20Resources/1.1.%20DataSheets/COMMs/GS-DS-U482C-5.0.pdf

Appendix B: Requirements

1 Functional System Requirement

Requirement: Prioritization & Quality of Service
The virtual channels shall be prioritised based on their number.
VC0 is the highest priority channel.

Requirement: Addressing
??? How the protocol should

2 Physical Layer Requirements

Requirement #6:
The PHY protocol shall support variable length frames. Length up to N bytes.

Requirement #6:
Shall apply forward error coding (FEC) techniques such as Reed Solomon and Convolutional coding.

Requirement #7:
Shall implement multiple waveforms (modulation-symbol rate-coding schemes) which can be changed on demand.

3 Medium Access Control Requirements

Requirement #4:
Shall work over Time Division Duplex (TDD) channel

4 Logical Link Layer Control Requirements

Requirement #1:
The protocol shall implement 1-4 Virtual Channels (VC) which can be used for different operational to optimize bandwidth usage

Requirement #3:
Shall implement reliable data transfer channels

Appendix C: Randomizer implementation

```
/*
 * The pseudo random sequence generated using the polynomial  h(x) = x8 + x7 + x5 + x3 + 1.
 * Ref: CCSDS 131.0-B-3, 10.4.1
 */
#define WHITENING_LEN   256
static const uint8_t whitening[WHITENING_LEN] = {
       0xff, 0x48, 0x0e, 0xc0, 0x9a, 0x0d, 0x70, 0xbc,
       0x8e, 0x2c, 0x93, 0xad, 0xa7, 0xb7, 0x46, 0xce,
        0x5a, 0x97, 0x7d, 0xcc, 0x32, 0xa2, 0xbf, 0x3e,
        0x0a, 0x10, 0xf1, 0x88, 0x94, 0xcd, 0xea, 0xb1,
        0xfe, 0x90, 0x1d, 0x81, 0x34, 0x1a, 0xe1, 0x79,
        0x1c, 0x59, 0x27, 0x5b, 0x4f, 0x6e, 0x8d, 0x9c,
        0xb5, 0x2e, 0xfb, 0x98, 0x65, 0x45, 0x7e, 0x7c,
        0x14, 0x21, 0xe3, 0x11, 0x29, 0x9b, 0xd5, 0x63,
        0xfd, 0x20, 0x3b, 0x02, 0x68, 0x35, 0xc2, 0xf2,
        0x38, 0xb2, 0x4e, 0xb6, 0x9e, 0xdd, 0x1b, 0x39,
        0x6a, 0x5d, 0xf7, 0x30, 0xca, 0x8a, 0xfc, 0xf8,
        0x28, 0x43, 0xc6, 0x22, 0x53, 0x37, 0xaa, 0xc7,
        0xfa, 0x40, 0x76, 0x04, 0xd0, 0x6b, 0x85, 0xe4,
        0x71, 0x64, 0x9d, 0x6d, 0x3d, 0xba, 0x36, 0x72,
        0xd4, 0xbb, 0xee, 0x61, 0x95, 0x15, 0xf9, 0xf0,
        0x50, 0x87, 0x8c, 0x44, 0xa6, 0x6f, 0x55, 0x8f,
        0xf4, 0x80, 0xec, 0x09, 0xa0, 0xd7, 0x0b, 0xc8,
        0xe2, 0xc9, 0x3a, 0xda, 0x7b, 0x74, 0x6c, 0xe5,
        0xa9, 0x77, 0xdc, 0xc3, 0x2a, 0x2b, 0xf3, 0xe0,
        0xa1, 0x0f, 0x18, 0x89, 0x4c, 0xde, 0xab, 0x1f,
        0xe9, 0x01, 0xd8, 0x13, 0x41, 0xae, 0x17, 0x91,
        0xc5, 0x92, 0x75, 0xb4, 0xf6, 0xe8, 0xd9, 0xcb,
        0x52, 0xef, 0xb9, 0x86, 0x54, 0x57, 0xe7, 0xc1,
        0x42, 0x1e, 0x31, 0x12, 0x99, 0xbd, 0x56, 0x3f,
        0xd2, 0x03, 0xb0, 0x26, 0x83, 0x5c, 0x2f, 0x23,
        0x8b, 0x24, 0xeb, 0x69, 0xed, 0xd1, 0xb3, 0x96,
        0xa5, 0xdf, 0x73, 0x0c, 0xa8, 0xaf, 0xcf, 0x82,
        0x84, 0x3c, 0x62, 0x25, 0x33, 0x7a, 0xac, 0x7f,
        0xa4, 0x07, 0x60, 0x4d, 0x06, 0xb8, 0x5e, 0x47,
        0x16, 0x49, 0xd6, 0xd3, 0xdb, 0xa3, 0x67, 0x2d,
        0x4b, 0xbe, 0xe6, 0x19, 0x51, 0x5f, 0x9f, 0x05,
        0x08, 0x78, 0xc4, 0x4a, 0x66, 0xf5, 0x58, 0xff
};

int skylink_apply_rand(csp_packet_t* packet) {

        for (unsigned int i = 0; i < packet->length; i++)
                packet->data[i] ^= randomizer[i % RANDOMIZER_LEN];

        return SKY_RET_OK;
}
```