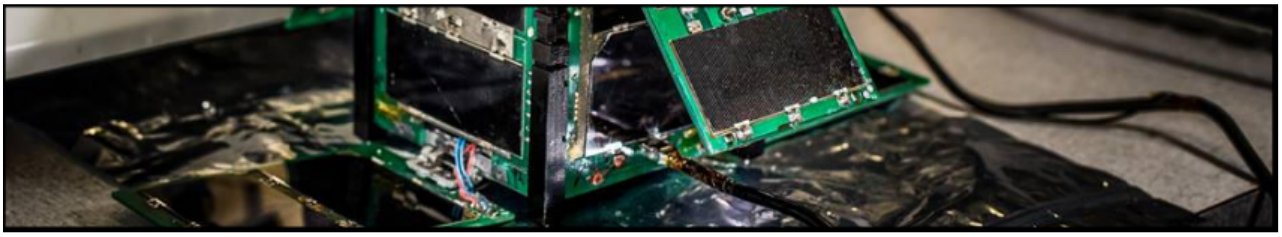
IPSA ONE 2020
ENGLISH / FRENCH

AR-020319-001



SOFTWARE DESIGN PHASE 1

DOCUMENTATION ABOUT OTHER MISSIONS' SOFTWARE DESIGN

20.03.18

Written by : Valentin Boullenger Elliot Poinsignon	Date and signature :
Checked by :	Date and signature :

Orbital **N**ano **E**xperiments
Association Law 1901 hosted at Institut Polytechnique des Sciences Avancées
63 boulevard de Brandebourg – 94200 Ivry-Sur-Seine
Mail : ipsa-one@ipsa.fr

CHANGE LOG

Ed.	Rev.	Date	Modifications
1	0	18/03/2020	Document creation

Contents

List of Figures

Nomenclature:

ADCS	Attitude Determination and Control System
AOS	Advanced Orbiting Systems
CCSDC	Consultative Committee for Space Data Systems
CFDP	CCSDS File Delivery Protocol
COM	Communication
EPS	Electrical Power Supply
OBC	On Board Computer
OSI	Open Systems Interconnection
RFMS	Radio Frequency and Modulation System
SCPS	Space Communications Protocol Specifications
SPP	Space Packet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Part I

Code Design

In this Part we will see how to write an efficient and robust program for a cubesat. First we are going to talk about the choice of the programming language (C/C++) and its consequences. In a second section we will see how to make a solid code architecture for this kind of project.

0.1 Programming Language

We chose to use the C/C++ language for the following main reasons (ref 1) :

1. This is a common language so there is a lot of documentation and good developers and teachers. Learn this language to a team will be easier than on another similar language;

2. C/C++ are very common language on university cubesat, so there are a lot of feedback and example for an OBC use;

3. There is a lot of libraries and different compilers for this language.

4. It's a efficient programming language in terms of memory use and speed.

C/C++ is not the only choice. One example we see in cubesat is the Spark/ada Language (ref 2 to 4). The best advantage of this language is that it include tools to verify the code and prevent any bug and problems.
The problem is that this language is not very often use. So there are very little documentation. Learn this language to a team will be very hard and risky.

0.2 Programming rules

We need to follow programming rules in a project of this size, mainly because of the complexity and the high number of developers working on it. The program must be clear and very well organized.

The rules:

1. **Respecting a coding style**
   It means that every developers need to write look alike codes (for example space at the same point: if "1+1 = 2" is the norm, don't write "1 + 1=2") This kind of thing can be implemented automatically, the program will adjust that alone. Developers also need to write similar looking structures. This will facilitate the understanding of the code for a reviewer;

2. **Reviewing and comments**
   Every code must be review by another developer to analyse it and maybe find error or structural problem. Developers need also to comment everything in their code to facilitate the reviewer work;

3. **Traceability**
   Every change made must be notified. We must be capable to trace every thing in the code to a specifications document.

4. **Tests**
   Every group of code shall have few tests to check that the code work as we want and still work if we add code around them. We need to think about the code structure around tests and not just put tests at the end of our code.

These rules will help to obtain a robust and well-written code, this is essential for the success of our mission.

0.3 Software Structure

The code will be divided into 5 libraries, one for each module (See Part.2 for more details on the different modules). That would help organize the code

1. **OBC Library:**
   Contain all the functions of the OBC (Telecommand understanding function; telecommand execute functions; function relative to the task management; function relative to time; function for the memory management; Modes functions; change Mode functions; reboot function and multiple tests for each.)

2. **EPS Library:**
   Contain all the functions of the EPS which the OBC can use (Data return functions; multiple tests.)

3. **ADCS Library:**
   Contain all the functions of the ADCS which the OBC can use (Data return function; Change Orientation function; Enable and Disable functions; multiple tests.)

4. **COM library:**
   Contain all the function of the COM (ground to space and space to ground) (Data return function; Sending data function; Enable and Disable functions and multiple tests)

5. **Engine library:**
   Contain all the function of the Engine (Data return function; Enable and Disable functions; Change thrust function; and multiple tests)

All of the function above are just idea and reflexion about what we need. They shall evolve toward more mature and precise concept. More precise function and command can be find in (ref 5)

Part II

Modules

0.4 OBC

The OBC (On-Board Computer) is the brain of the cubesat. His role is to coordinate all of the other modules and the ground, and also take care of the potential errors and threats for the cubesat. It will periodically (The time interval will depend on the type of data and must be determined individually) send commands to the different critical modules to order them to send critical data (like battery charge for the EPS or spatial orientation for the ADCS). It is absolutely vital for the success of the mission.

0.5 EPS

The EPS manages the electricity on board. It's stand alone, it works without the OBC, but the OBC can read the state of the EPS (Battery charge for example)
The OBC can read its stats (for example battery charge) and react accordingly. For that the OBC must send periodically a command through a bus (surely an I^2C bus or similar bus) and the EPS sends the data back.

0.6 COM

The COM is the module which allows the communication between the ground and the cubesat. It tests and activates at the beginning of the "initials tests" mode (more details in Part II), then the deployment of the antenna is one of the most critical aspect of our mission, this is why the OBC will attempt to deploy it intermittently until a telecommand from the ground.
The COM is the link between the ground and the cubesat, so all telecommands and data recovery will go through this module. All data will go through 10 protocoles layer (5 for the ground station and 5 for the cubesat). In terms of data rate, the maximum of our antenna is about 9600bits/s (ref 6).

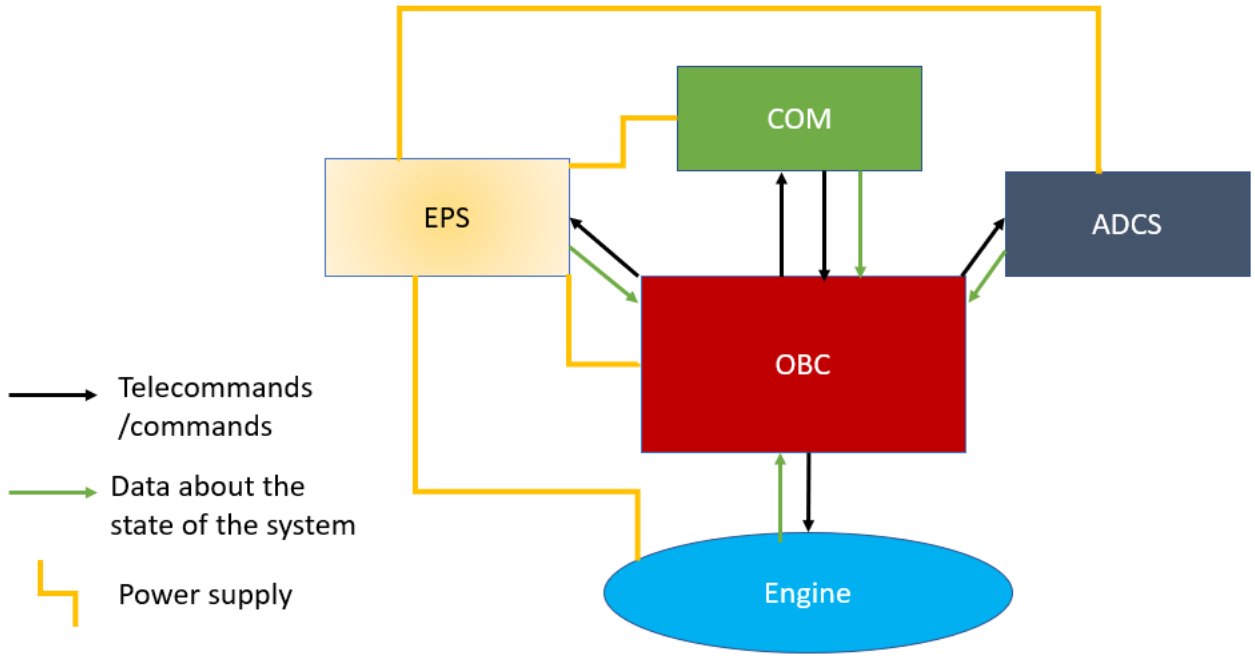
0.7 ADCS

The ADCS is the module which controls the position of the satellite in space (attitude) and also analyses the environment to determine its position in space.
It is very important for our mission because of our payload, we must control the direction of our engine to control our trajectory.
It will be controlled by the OBC with different commands.

0.8 Engine

The Engine is the payload of our cubesat. It will be managed by the OBC trough different commands.

Figure 1: Schematic of the different Modules and links



Part III

Modes

A cubesat must have different modes to accommodate different situations. For our cubesat we have 6 different modes. Each of these modes are designed to answer a specific situations.

1. Just after the release (Deployment);
2. ignition and test of the cubesat (Initials Test);
3. in case of a critical failure (Survival);
4. a passive mode (safe);
5. in case of massive communication with the ground (Comm);
6. when we use the engine (Thrust).

An other example of modes in cubesat can be find in (ref 7) and it's very similar with what we find.

0.9 Deployment

This is the first mode, it is activated after the release of the cubesat. The duration of this mode will depend on our Launch vehicle and our software configuration capability.After this mode, OBC launch the Initials tests.This mode is only active one time.
In this mode, the COM,the ADCS and the Engine are shutdown.

0.10 Initials Tests

In this mode, the OBC launches different tests on the COM and the ADCS, at the beginning alone, and after together, to be sure that all systems are operational. It will also try to deploy the antenna. If the test is successful, the cubesat goes to safe mode, if its not it goes to survival mode.
The test order will be Vital module first alone. (Which vital module will be first will be discuss in the specification) After in pairs, in threes and at the end all modules together.

0.11 Survival

This mode is only activated in a case of a major/critical error on a module, or in case of energy supply problems (under a certain batteries critical charge limit) or by a telecommand from the ground.
In this mode all modules works in slow motion, engine and ADCS are shutdown and communications are very rare. The cubesat waits in this mode until the ground has a solution for the problem.
List of condition which can lead to this mode:

1. Critical error on EPS (Batterie bellow certain charge; Problem in energy distribution; No batterie charge during day [problem with solar panel deployment for example]);
2. Critical error in COM (Problem to send message to ground; and maybe Problem with the deployment of the antenna: the necessity to activate or not the survival mode here will depend on our batteries and the energy need to deploy the antenna.);

3. Critical error in ADCS (Critical error in ADCS;No orientation movement possible);
4. Critical error on OBC (Error in data transmission between different modules [Bus problem]; Major problem detected in the software [infite loop for example]; too much data charge)
5. Critical error on Engine (Single-event upset; over-heat)
6. Command from Ground

0.12 Safe

This is the normal mode, the engine is shutdown, but the other modules works nominally. In this state the cubesat waits order(s) from the ground.

0.13 Comm

This mode is used in need of a massive communication between the satellite and the ground. In this mode the COM modules works at its maximum and the Engine is shutdown.

0.14 Thrust

This mode is the only one where the engine is activated. In this mode communication is disable

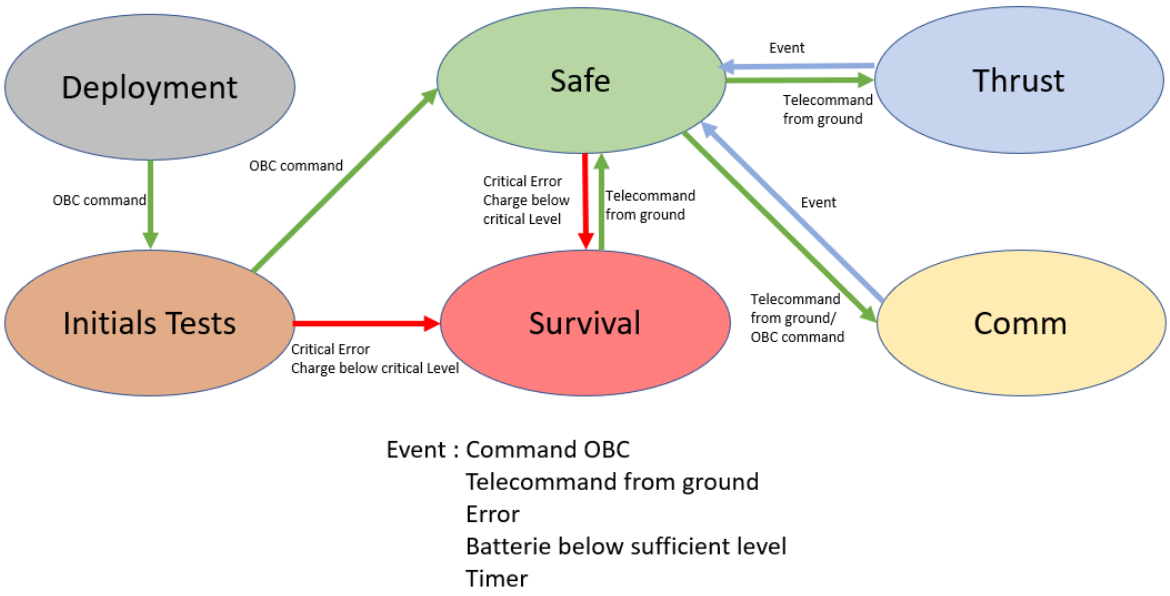


Figure 2: Schematic of the behavior of the cubesat

	OBC	EPS	COM	ADCS	Propu
Deployment	Yes	Yes	No	No	No
Initial Test	Yes	Yes	Test	Test	No
Survival	Yes (min)	Yes (min)	Sometimes	No	No
Safe	Yes	Yes	Yes	Yes	No
Comm	Yes	Yes	Yes (max)	Yes	No
Thrust	Yes	Yes	Passive	Yes	Yes

Figure 3: Module status table

Part IV

Protocols

In this part we will discuss the different communication protocols we can use for a cubesat. This is based on the OSI model, but only with 5 layers (and not 7).

1) Physical Layer
2) Data Link Layer
3) Network Layer
4) Transport Layer
5) Application Layer

Presentation and Session layers are rarely used in our case this is why they are not here.

The following figure regroups all the proctocols we can use for our project (ref 8).

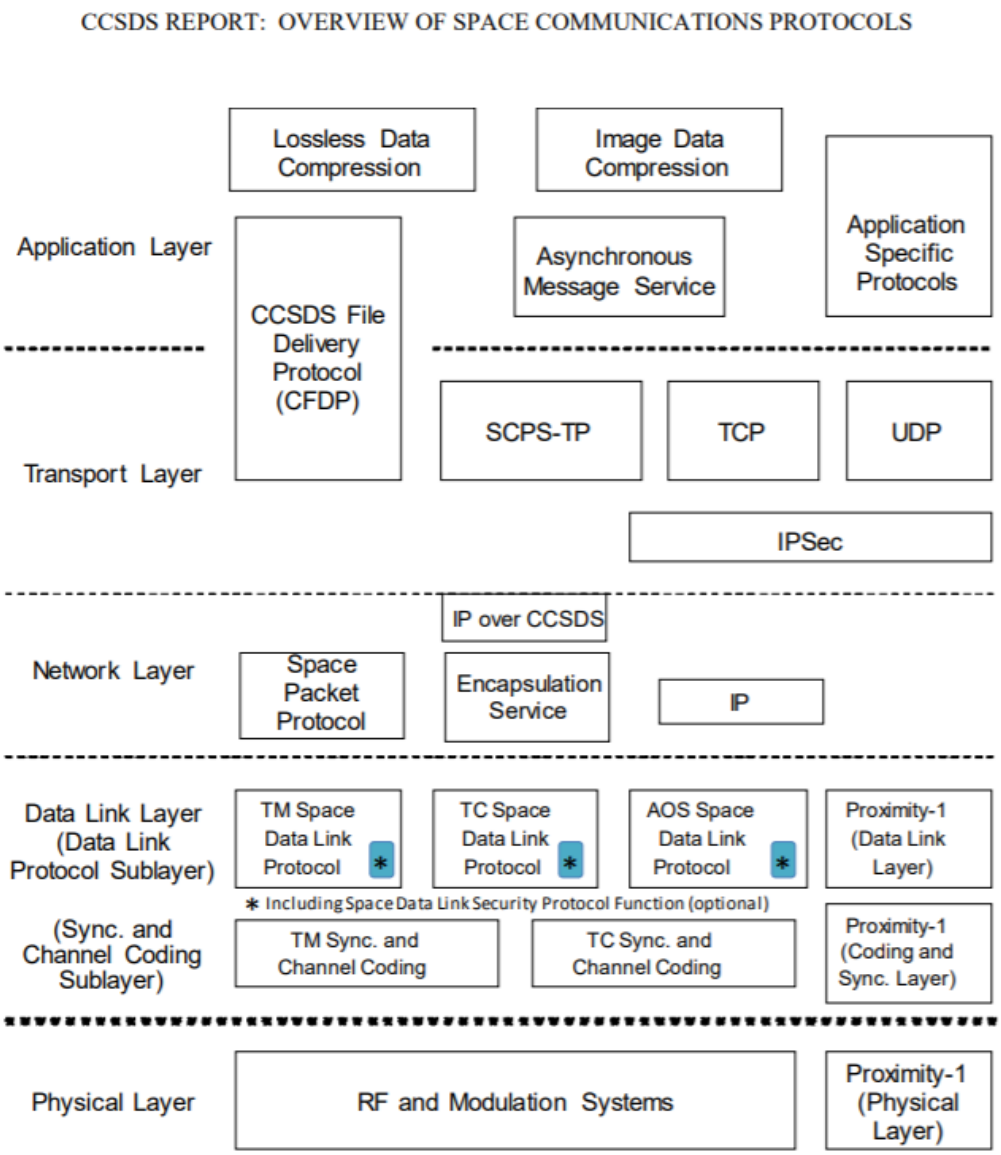


Figure 4: Space Communications Protocols Reference Model

0.15 Physical layer

This is the protocol who defined how communicate through space with the ground station.
For this layer we have 2 options : RF and Modulation System and Proximity-1.
For our case RFMS is preferable, because this is the best to communicate between a satellite and a ground station. Proximity-1 is mostly used for close communication, between a rover and a spacecraft for example (ref 10).

0.16 Datalink layer

In this layer, we have two sublayers :

- Sync and channel Coding: prepare frames to be transferred on a space links;
- Data Link Protocol: convert the data units from the upper layers to frames (or reversal)

For the Sync and channel Coding, we have 2 options (Proximity-1 is not used in physical layer, so same for upper layers), TM (for Telemetry) and TC (for Telecommand). TM is most of time used for Space to ground or Space to Space Telecommunications (But not only) and TC for ground to space (But not only). This is why we will probably use both of them (ref 8.1).

For the Data Link sublayer we have one option if we use TC for the Sync and channel Coding : TC Space Data Link.
If we use TM we have 2 possibilities : TM or AOS. AOS is useful for huge data transfer like photo or video (ref 8.1), in our case we don't have any camera on our cubesat or any big data transfer. this is why AOS is not very useful for us.

0.17 Network Layer

In this layer you have two options :
1) Space Packet Protocol
2) Encapsulation Service + IP over CCSDS (+ IP)
The main protocol we will use is SPP, it is developped for space applications and for ground to space and space to ground communications. But the packet which are used by SPP must have a Packet Version Number authorized by CCSDS (ref 9). This is why it can be interesting to use Encapsuation Service in rare cases.

0.18 Transport layer

For this Layer we have many options. If we use SPP, we can use the CFDP or SPP can be use as Transport layer too. CFDP also contain Application layer function (files managements).
CFDP can be used directly over Encapsulation, IP over CCSDS or other Transport layer protocols put on IP. such as TCP, SCPS or UDP (ref 8.2).
If we use IP over CCSDS and TCP it will be much easier for the devellopement, because IP and TCP protocols are very often use in a lot of context today. We have more documentation about them and they are robust in terms of cyber-security.

0.19 Application Layer

Protocols used in this layer depend mostly on the payload and what kind of data we want to send.
For example Lossless Data Compression is useful if we want the least loss on our data.
Image Data Compression is use in case of image, and this is not our case.
There are a lot of other protocols depending of what kind of data we want to send ref (8.3).

In the following figure you can see some possibilities of protocol configurations (ref 8).

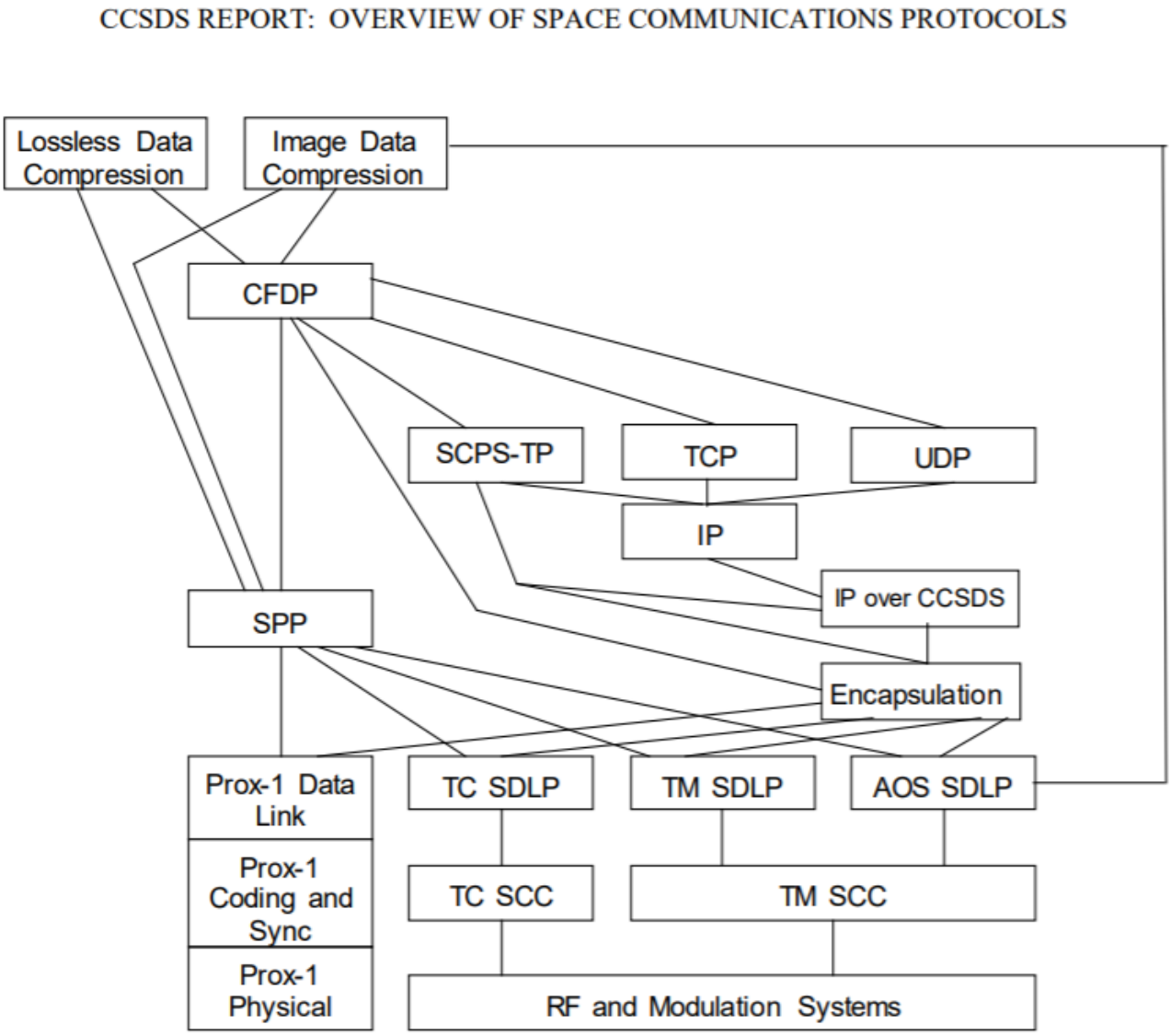


Figure 5: Combinations of Space Communications Protocols

Part V

Bibliography

0.20 Code Design

(1) https://en.wikipedia.org/wiki/C_(programming_language)
(2) https://en.wikipedia.org/wiki/SPARK_(programming_language)
#Verification_conditions
(3) https://blog.adacore.com/ten-years-of-using-spark-to-build-cubesat
(4) https://www.researchgate.net/publication/296671392_A_SPARKAda_
CubeSat_Control_Program
(5) https://www.academia.edu/29912309/Documentation_of_AAU-Cubesat_
On_Board_Computer_Software

0.21 Modules

(6) from Analysis report ARAGO : TELECOM - MODULE (2018, AR-150418-001) :
https://drive.google.com/drive/u/0/folders/1GrIdBMEIote-2I03yt8BK6JO3reiQELs

0.22 Modes

(7) An example of mode in a other cubesat can be fund with the document: Ingénierie système sur le nanosatellite IGOsat,Phase A by Moufida Chariet.

0.23 Protocols

All of the information are from the CCSDS:
https://public.ccsds.org/default.aspx
(8) The two figure are from:
https://public.ccsds.org/Pubs/130x0g3.pdf
(8.1) chapter 3.2.1 GENERAL FEATURES OF DATA LINK PROTOCOLS
(8.2) chapter 3.4 TRANSPORT LAYER
(8.3) chapter 3.5 APPLICATION LAYER
https://public.ccsds.org/Pubs/130x0g3.pdf
(9) chapter 2.1 CONCEPT OF ENCAPSULATION SERVICE
https://public.ccsds.org/Pubs/133x1b2c2.pdf
(10) chapter 2.1 PHYSICAL LAYER OVERVIEW
https://public.ccsds.org/Pubs/211x1b4e1.pdf

0.24 Examples and other documentation

http://www.leodium.ulg.ac.be/cmsms/uploads/11-12_OBC_DickAlexei.pdf (fr)
http://www.goldstem.org/Swiss%20Cube/05%20-%20Flight%20software/S3-BC-SE-1-0-Fligh
Software_Architecture.pdf
https://www.researchgate.net/publication/228371338_Dependable_Software_BOSS_
for_the_BEESAT_pico_satellite
https://www.researchgate.net/publication/336885538_A_Comparative_Survey_
on_Flight_Software_Frameworks_for_'New_Space'_Nanosatellite_Missions