UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

telecos
BCN

Contributions to simulation and emulation of Inter-Satellite Links

Degree Thesis
submitted to the Faculty of the
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Dolz Puig, Arnau

In partial fulfillment
of the requirements for the degree in

TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Advisors: Dr. Ruiz-de-Azua, Joan A.,
Prof. Camps Carmona, Adriano José
Barcelona, June the 21st 2022

telecos
BCN

Contents

Listings

List of Figures

List of Tables

Abbreviations

ADC Analog to Digital Converter

ARTES Advanced Research in Telecommunications Systems

BER Bit Error Rate

BW Bandwidth

DAC Digital to Analog Converter

DSS Distributed Satellite System

DSS-SIM Distributed Satellite System Simulator

ECI Earth-Centered Inertial

ESA European Space Agency

EuCNC European Conference on Networks and Communications

FPGA Field-programmable gate array

FSS Federated Satellite System

GS Ground Stations

IoSat Internet of Satellites

IP Internet Protocol

ISL Inter-Satellite Links

ISS International Space Station

ITU-R International Telecommunication Union Radiocommunication Sector

LAN Local Area Network

LO Local Oscillator

MSPS Mega Samples per Second

NASA National and Aeronautic Space Administration

NB-IoT Narrow Band Internet of Things

NS-3 Network Simulator 3

RF Radio Frequency

Rx Reception

SatSim Satellite Communication System Simulator

SDR Software Defined Radio

SNS3 Satellite Network Simulator 3

STK	System Tool Kit
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
VSAT	Very Samll Aperture Terminal
WLC	Water Liquid Content

Abstract

Recently, there has been a paradigm shift in space missions. They no longer rely on a single satellite to achieve all the objectives of a mission. Responsibility is shared between devices that can communicate and share resources to achieve these objectives. This paradigm shift has increased the need to assess communication performance within FSS. Despite proposals for simulators and emulators to achieve this goal, none of them integrates the dynamics of satellites with their communications.

This thesis presents two proposals and the implementation of one of them to extend the capabilities of the communication channel of the simulator developed by the NanoSat Lab of the UPC. Additionally, following the lines of the i2Cat Space Communications research group, a test bench is also presented to demonstrate the feasibility of a satellite contact emulator. As it will be demonstrated, developing such an emulator has not been possible.

Finally, it is concluded that both parts of this thesis may converge. Thus, combining the possibilities offered by the current simulator with those available in the emulator under development.

Resum

En el transcurs dels últims anys s'ha observat un canvi en el paradigma de les missions espacials. Aquestes ja no compten només d'un únic satèl·lit que hagi d'assolir tots els objectius d'una missió. Es reparteix la responsabilitat entre diferents dispositius els quals són capaços de comunicar-se i compartir recursos per tal d'assolir tals objectius. D'aquest canvi de paradigma neix la necessitat de poder avaluar el comportament de les comunicacions dins d'un FSS. Tot i l'existència de simuladors i propostes d'emuladors per assolir tal objectiu, cap d'ells integra les dinàmiques dels satèl·lits amb les seves comunicacions.

En aquesta tesi, es presenten dues propostes i la implementació d'una d'elles per tal d'ampliar les capacitats del canal de comunicacions del simulador desenvolupat pel NanoSat Lab de la UPC. A més a més, seguint les línies del grup de recerca de Comunicacions Espacials d'i2Cat també es presenta un test-bed per tal de poder demostrar la viabilitat d'un emulador de contactes de satèl·lits. Doncs, com es demostrarà, el desenvolupant d'un com a tal no ha estat possible.

Finalment, es concloura que ambdues parts d'aquesta tesi puguin convergir en algun moment combinant així les possibilitat que ofereix l'actual simulador amb les d'un emulador que està en desenvolpament.

Resumen

En el transcurso de los últimos años se ha observado un cambio en el paradigma en las misiones espaciales. Éstas ya no cuentan sólo con un único satélite que tenga que alcanzar todos los objetivos de una misión. Se reparte la responsabilidad entre diferentes dispositivos que son capaces de comunicarse y compartir recursos para alcanzar tales objetivos. De ese cambio de paradigma nace la necesidad de poder evaluar el comportamiento de las comunicaciones dentro de un FSS. A pesar de la existencia de simuladores y propuestas de emuladores para alcanzar tal objetivo, ninguno de ellos integra las dinámicas de los satélites con sus comunicaciones.

En esta tesis, se presentan dos propuestas y la implementación de una de ellas para ampliar las capacidades del canal de comunicaciones del simulador desarrollado por NanoSat Lab de la UPC. Además, siguiendo las líneas del grupo de investigación de Comunicaciones Espaciales de i2Cat también se presenta un test-bed para demostrar la viabilidad de un emulador de contactos de satélites. Pues, como se va a demostrar, el desarrollador de uno como tal no ha sido posible.

Por último, se concluirá que ambas partes de esta tesis puedan converger en algún momento combinando así las posibilidades que ofrece el actual simulador con las de un emulador que está en desarrollo.

Acknowledgements

The author would like to thank Professor Adriano Camps and PhD Joan A. Ruiz-de-Azua for the opportunity to participate in this project. He would also like to thank them for their help and support. A special mention goes to the Space Communications research team and the NanoSat Lab team for their support when developing this thesis.

Finally, the author's heart holds to the advice received from his late father: **Any great discovery is the result of teamwork**. Although no longer among us, I hope you can see this small contribution to research. I dedicate this work to you.

Revision history and approval record

Revision	Date	Purpose
0	15/05/2022	Document creation
1	20/05/2022	Document revision
2	12/06/2022	Document revision
3	16/06/2022	Document revision
4	21/06/2022	Document delivery

DOCUMENT DISTRIBUTION LIST

Name	E-mail
Dolz Puig, Arnau	arnau.dolz@estudiantat.upc.edu
Ruíz de Azúa Ortega, Joan Adrià	joan.ruizdeazua@i2cat.net
Camps Carmona, Adriano José	adriano.jose.camps@upc.edu

Written by:		Reviewed and approved by:			
Date	21/06/2022	**Date**	21/06/2022	**Date**	21/06/2022
Name	Dolz Puig, Arnau	**Name**	Ruiz de Azúa Ortega, Joan Adrià	**Name**	Camps Carmona, Adriano José
Position	Project Author	**Position**	Project Supervisor	**Position**	Project Supervisor

1 Introduction

Since the firsts CubeSats were launched in June 2003, the number of these satellites orbiting the Earth has exponentially grown. The rise of small satellite missions has made the space sector change its mind regarding mission approaches. Instead of using a stand-alone satellite, several small satellites can be used to create a constellation to achieve the same goals. Such satellite constellations can offer several advantages compared to stand-alone satellite missions. Some of them are the reduction of revisit time and the enlargement in the coverage they can offer. Those advantages are relevant due to the impact that they can cause on the current Earth observation methods and satellite communications. For example, a great activity is being conducted to integrate 5G in satellite communications [1], [2].

When a mission is designed based on a constellation of satellites, it is thought to meet specific objectives. It means that in some cases, the satellites that form a constellation must be able to share their resources to meet the mission goals. Such constellations are known as "Distributed Satellite Systems" (DSS). Some DSS propose to deploy a satellite network by means of the Inter-Satellite Links (ISLs). ISLs are those connections established between two satellites to provide communication means. When such ISLs depend on routing protocols, then the network is defined as the Internet of Satellites (IoSat), as can be seen, in [3] and [4]. This change in the paradigm of space missions has created the need to assess the feasibility of such networks, not just evaluate the platforms. It has also created the need to check and create new communication protocols to achieve reliable communication networks to make IoSat a reality. As well as the necessary tools to emulate and simulate the behaviour of the nodes in such networks.

As exposed in Section 2, there is not a wide range of offers to emulate or simulate ISL. The existing tools are slightly accessible (the main reason is almost always their price) nor specifically dedicated to the emulation and simulation. From this need, in 2018, aiming to achieve such a novel purpose (simulating ISL communications protocols) in [5] a simulation tool was presented. This simulation engine is known as "Distributed Satellite Systems Simulator" (DSS-SIM). The DSS-SIM is a tool based on NS-3. It was developed to put in one solution both satellite dynamics and communications. It allows users to implement custom subsystems and resource models for spacecraft. It also provides a versatile and highly extensible tool where users can tailor discrete physical models representing spacecraft components and instruments. The DSS-SIM is currently being extended by the Space Communications research group at i2Cat. Such an extension is being conducted in the framework of the Catalan space programme, the "*NewSpace Strategy of Catalonia*".

In terms of ISL emulation, and as shown in Section 2.2, there are no tools available. As a consequence of this lack of emulation tools from the Space Communications research group at i2Cat, the creation of a Satellite Contact Emulator is proposed. Its main goal is to find a solution to the lack of ISL emulation tools. Nevertheless, the contribution made in this thesis cannot reach such an achievement. A few things need to be mentioned. The first one is the fact that the development of an emulator is done from scratch. Meaning that the work to do will not only concern the emulation of the communications channel. It will also consider the design of the design of the entire emulator itself. Additionally, note that the Emulator is also developed by colleague Antonio Romero as explained in Section 1.1.

To conclude, this thesis contributes to the simulation tool developed at the NanoSat Lab by integrating two new features on the communications channel. Furthermore, it aims to contribute to the development of the prototype of the Satellite Contact Emulator.

1.1 Objectives

The space sector is currently making more equipment that simulates and emulates ISLs. This thesis contributes to fulfilling this need by expanding the communications channel features of the DSS-SIM. Also, it is expected to participate in the creation of a Satellite Contact Emulator.

This project expects to achieve:

- Contributions to the Simulation of ISL
  - The implementation of a model representing atmospheric effects on the communications for the DSS-SIM.
  - The definition of a transceiver simulates the collision of packets, the antenna parameters (such as gain and radiation pattern), the noise floor and the bandwidth for the DSS-SIM.
  - The implementation of a spectrum-aware communications channel model for the DSS-SIM.
- Contributions to the Emulation of ISL:
  - An end-to-end testbed using Software Defined Radio (SDR) devices to emulate ISLs dynamics and channel.
  - A communications channel model that must perform the attenuation, Doppler effect and the transmitted signals' delay.

At this point, it is essential to mention that the implementation of an orbit propagator that retrieves the position of the satellites during the emulation and allows to compute the necessary parameters to perform communication emulation is needed. As a part of his final Degree's thesis "*Contributions to satellite subsystem models and communications link emulation*" [6], Antonio Romero has contributed to the Satellite Contact Emulator. He has developed software that runs into a central computer that orchestrates real-time emulation. Mainly, his work is centred on the computation of channel effects and their propagation to the SDR. It has allowed the implementation of the communications channel model presented in this thesis.

The requirements and specifications are presented in Appendix D.1 (Tables D.1 and D.2). Note that requirements reflect the original design. During the project's development, there have been changes due to technology limitations that have affected the requirements and specifications. This had an impact on the final objectives of this thesis. Deviations from the initial plan are explained later in this document. The explanations regarding management and work-plan deviations can be found in Appendix A. The technical reasons that justify those deviations from the initial objectives are found in Sections 3, and 5.

1.2 Thesis Outline

This document is structured to expose the work done during the development of this thesis. The document is structured as follows:

2) State of the Art: A review of the current literature and research on the topic is done. Additionally, the demands and offers of simulation and emulation tools that can be found in the space communications market and industry are analysed.

3) Methodology and Project Development: The development phases and processes of the project are explained here.

4) Experiments and Results: The results obtained with the developed solutions are presented.

5) Conclusions and Future Work: The conclusive messages of the work and future developments are presented.

Appendices to complement the document are added at the end. This decision was made aiming to present the project's development and its results in the clearest form and to respect the size of the document. The appendix is structured as follows:

A) Management: The work plan is presented here. On top of that, deviations to the original work plan are commented and analysed.

B) Budget: The budget of the project developed in this thesis is presented.

C) Environmental impact: The environmental impact analysis of the project is exposed in this section.

D) Plots and tables: Plots and tables used and created to present the results of this thesis are found in this thesis.

2 State of the art

Different developments have been conducted in the last years to achieve equipment that emulates and simulates satellite communications. For ISL, they are still preliminary initiatives. Each subsection reviews the related technologies.

2.1 Simulation of Inter-Satellite Links

Nowadays, the available tools to simulate ISL are based on hybrid solutions. Such solutions are a combination of the software used to obtain the satellite dynamics (e.g. satellite positions) and network simulators to represent the network behaviour. Some of the most notorious network simulators are QualNet, OMNeT++ and NS. Note that all of them are developed for terrestrial communications. An approach is using licensed software (e.g. STK). As mentioned in [5], most of these software tools are not specifically dedicated to the simulation of ISLs. They are hybrid solutions. Such solutions combine simulation software to obtain the satellite dynamics and network simulators. Another example of this licensed software is the "*Satellite Communications Toolbox*" [7]. It is a MATLAB extension that computes satellite communications.

Some attempts at a fully integrated system can be found. Such as the "*Satellite Communication System Simulator*" (SatSim) [8]. It is a Python-based multi-satellite network simulator. Another, is the "'*Satellite Network Simulator 3* (SNS3) [9], which extends the NS-3 simulator. Those projects demonstrated the sustainability of a fully integrated system. However, both failed in the attempt to be reusable platforms where the user could customise spacecraft resources and platforms [3]. They also failed in flexible scenarios. As they were thought only to simulate satellite communications.

To overcome the problems found in other proposals, a simulation tool is presented in [5], the Distributed Satellite System Simulator (DSS-SIM). The DSS-SIM is an NS-3 based solution. It allows the users to implement custom subsystems and resource models for spacecraft. Letting the simulation of the impact of ISLs over spacecraft resources, enabling the implementation of high-level spacecraft interactions. It provides a versatile and highly extensible tool where users can tailor discrete physical models representing spacecraft components and instruments. Moreover, the DSS-SIM simulates communication protocols. As it is a modular software design, its development can be done in separate modules. Therefore, two new modules of the communication layers are implemented in this dissertation. Now it is time to look at the current technologies of the atmospheric attenuation models and the spectrum channel modules from NS-3.

Before going into deeper details, it is important to introduce NS-3. NS-3 is a discrete-event network simulator for Internet systems. It is an open-source software maintained by a worldwide community [10]. Some classes are remarkable for the scope of this thesis. They are the following ones: (1) PropagationLossModel, (2) Channel, (3) NetDevice, (4) MultiModelSpectrumChannel, (5) SpectrumChannel, (6) SpectrumPhy, (7) SpectrumModel, and (8) SpectrumValue. From the PropagationLossModel class, two specific functions are importantly remarkable. The *SetNext()* and *GetNext()*. Those functions allow the simulator to add propagation models to the simulation.

Regarding the *atmospheric attenuation modelling*, the atmospheric attenuation models presented in [11], [12], [13], and Benoit's method [14] have been considered. The cloud model, the Thomson modelling of the clouds [15], [16], and [17] are used to obtain the parameters that characterise a cloud. It is important to mention that STK uses [15] to model the clouds in the atmospheric analysis.

Concerning the *Transceiver*, not many developments related to transceivers have been found in simulation. Nevertheless, some classes from NS-3 aim to supply some parameters to simulate such a component. Those classes are NetDevice and SpectrumPhy. As it has already been said, a NetDevice is not a physical layer class, unlike the SpectrumPhy. However, some parameters can be added to the implementations of such a class.

About the *Spectrum model*, there are little to no applications that simulate ISL using the NS-3 class Spectrum Channel. However, the Lena Project [18] is a simulation tool for NB-IoT environments at physical and protocol layers that uses the NS3 multi-model-spectrum class. It creates a model representing the physical communication layer, allowing frequency division to simulate NB-IoT protocols.

To conclude, by using [11] and [15], the modelling of the atmospheric effects is based on the ongoing available technologies. Regarding developing the Spectrum model for the channel, the use of the NS-3 MultiModelSpectrumChannel will allow many other features to be implemented as future work.

2.2 Emulation of Inter-Satellite Links

Regarding the technologies that concern the emulation of ISLs, there are even fewer than in the simulation. In the academic area, there have been few attempts to create a testbed with SDRs ([19] and [20]). However, both testbeds are improvements over specific hardware.

In the commercial sector, some communications channel emulators can be found. However, their capabilities miss features such as:

- *Lack of connectors*: This means that when we want to test a network of satellites, it would not be feasible to use only one emulator. Most of them are meant to be used as channel emulators for the communication between two points (e.g. up-link and down-link communications), not for Inter-Satellite Links.
- *Narrow frequency range*: This fact makes those emulators very restrictive when testing different communication protocols and modulations. Especially if they are not working in the same frequency range.
- *Price*: Most channel emulators are modelled using hardware. This fact makes them costly devices, which leads to bare accessibility. See the comparison of prices and capabilities of different channel emulators in Table D.4, page 51.

One company that has developed a solution that can be escalated in terms of a satellite network allowing ISL and Satellite to Earth communications. Keysight Technologies propose such a solution with its product *"PROPSIM F64 5G Channel Emulation Solution - F8800A"* [21]. It is a hybrid solution of hardware and software that achieves full-stack end-to-end RF performance testing. Unfortunately, no quotation for this product has been obtained.

It must be mentioned that in 2004 the NASA Space Communication Emulation Facility team developed a facility that should approximately emulate the conditions in space that impact space communication [22]. However, the only communications to contemplate at that moment were the up-link and down-link contacts. As seen in [22], this emulator was running on a Linux operating system and employed both Java and C++ programming codes. Also, NASA, because of the increasing data requirements for space communications at that moment, developed the "*Space Link Extension (SLE) Emulation* *for High-Throughput Network Communication*" [23]. However, none of them considers ISLs emulation.

In 2019, at European Conference on Networks and Communications (EuCNC) in Valencia, another proposal to emulate satellite contacts was demonstrated [24]. It is essential to say that initially, it was meant to emulate communication between two different nodes on the Earth using 5G satellite communications. The approach made in [24] to develop the emulator is very similar to the one presented in Section 3.4.

From the European Space Agency (ESA), two project proposals are commented. They were launched by the ESA ARTES programme, calling for creating a satellite communications emulator. The first one was made in 2017 to develop a "*Real Time Satellite Network Emulator*" [25]. Airbus D&S SAS coordinates it as the main contractor and Magister Solutions Ltd and VTT Technical Research Centre of Finland (VTT) as subcontractors. Its results were presented in September 2021. It consists of three main parts:

- STK to generate the orbits and obtain the parameters.
- NS-3 to do the network emulation.
- SCNE Orchestrator to monitor the emulation.

Note that the Emulator structure developed by Airbus has some characteristics that the one presented in this thesis implements. Its main features are:

- The capability to design satellite constellations using STK.
- The capability to study ISL routing algorithms.
- The capability to study different protocol layers.
- It can configure different application scenarios. Such as traffic models and communication pairs.
- The capability to add new air interfaces and network layer protocols and algorithms for research purposes.

The second proposal was made in 2021. It consists of creating an "*Emulator of Satellite-Terrestrial 5G Radio Channels Hybrid Channel Emulator for Combined Satellite-Terrestrial 5G Channels*" [26]. Square Peg Communications Inc. coordinates this project as a prime contractor and WORK Microwave GmbH as a subcontractor. Note that this proposal, as the first one, has some similar specifications to those presented for the emulator developed in this thesis.

After reviewing the projects proposed by the ESA, it can be concluded that at this day, there is particular interest in the development of tools to emulate ISL.

The last things to be reviewed are the technologies used to develop the emulator. The Satellite Contact Emulator consists of 3 main hardware components. Software-Defined Radios (SDRs) to emulate the RF behaviour, a switch to interconnect the SDRs, and a central computer to orchestrate the entire process.

According to [27] a SDR is a radio communication piece of hardware that uses software for the modulation and demodulation of RF signals. Such a system can perform significant signal processing in a reconfigurable piece of digital electronics. Note that such hardware is not easy to create. This is the reason for such pieces of hardware to have high prices. The range of prices goes from hundreds of euros to some thousands of euros. The difference in price many times relays on the specifications of the devices.

This means that by using SDRs at the time to develop the emulator, an increase in flexibility is achieved. This flexibility means that the signal processing can be configured to achieve the desired objectives. There are several ways to implement the SDRs. The most user-friendly is using GNU-Radio. According to [28], GNU-Radio is an open-source toolkit for implementing software radios. However, another way is to set up the SDRs. It consists of typing a script in a compatible language. The most common ones are C, C++, and Python. However, to use this script, the user should do a cross-compilation of the software before uploading it to the board. Furthermore, when generating that piece of software, manufacturers provide libraries to make it easy for the users to implement the devices. Some of the most important and widely extended are:

- **Libiio** [29]: Libiio is a library firstly developed by Analog Devices to ease the development of software interfacing Linux Industrial I/O (IIO) devices. The library abstracts the low-level details of the hardware and provides a simple yet complete programming interface that can be used for advanced projects.
- **Nuand BladeRF** [30]: This library supports the Nuand bladeRF and bladeRF Micro USB 3.0 Software Defined Radio (SDR). It is intended to support developers looking to (1) Build both generic SDR software and domain-specific applications atop of the bladeRF and (2) Evaluate and experiment with the hardware, using routines that provide low-level access to device settings and registers.

Note that the main difference among different libraries is the developers that are the manufacturers of the SDRs. Those libraries are meant to be an easy tool to work with such devices. Nevertheless, note that the SDRs could be programmed from scratch by using low-level programming languages and discovering the boards' workflow.

The next piece of hardware to be analysed is the switch. The technologies found among commercial switches are the *managed* and the *unmanaged* ones. The first ones require configuration. Despite being manageable, such complexity is not needed for this dissertation. The second kind does not require any configuration. They are ready to be plugged in and fully functional.

To conclude, it has been seen that there is no wide range of possibilities in the space sector when looking for a satellite communication emulator. Such a lack of emulation tools leads to the solution presented in Section 3.4. Note that as presented in such section, the features to be developed are up-to-date with the proposals from the ESA's ARTES programme.

3 Methodology and project development

This chapter is divided into four different sections. Sections 3.1, 3.2, and 3.3 present the methodology for designing and implementing the classes that extend the DSS-SIM. Meanwhile, in Section 3.4, the methods undertaken to develop the design and implement the Satellite Contact Emulator are given.

The methodology followed to develop both projects that compose this thesis is:

1. Research and review current technologies and current similar implementations.
2. Design the software to be implemented.
3. Discuss and agree on the design with the thesis advisors.
4. Implement the design.
5. Create and execute a test to verify its functionality.

Networking module context

The DSS-SIM is a tool that addresses the lack of a fully integrated simulation solution. It allows the simulation of the communications network and autonomous satellite coordination mechanisms [5]. It is important to mention that the DSS-SIM employs the C++ programming language. For this reason, the software is developed in such a programming language. In this Section, a software-level description is provided. Figure 1 presents a block diagram of the classes of the simulator.

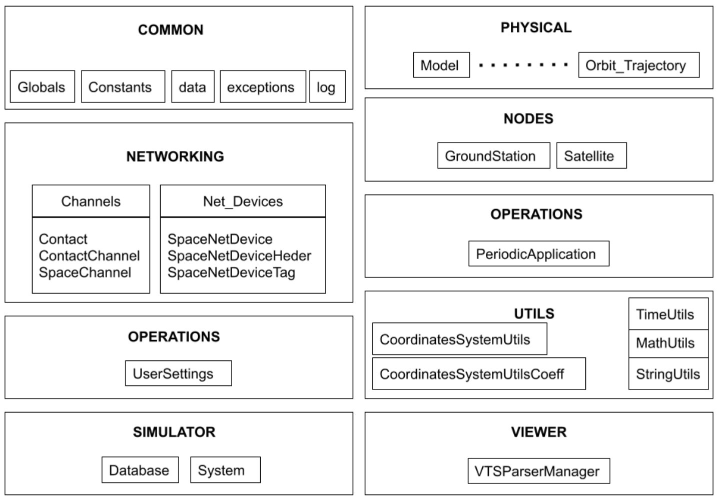


Figure 1: High-level architecture diagram of the DSS-SIM.

This dissertation pretends to extend the "*Networking*" module of the DSS-SIM. Firstly, context on the NS-3 classes extended by the DSS-SIM is needed. Such classes are the following ones:

- *PropagationLossModel* [31]: This class is an interface that allows the implementation of classes such as FriisPropagationLossModel (already implemented and included in NS-3) that simulates the attenuation in the communications.

- *Channel* [32]: This class simulates a communication channel. The DSS-SIM extended it to the SpaceChannel.
- *NetDevice* [33]: The NetDevice is the network layer to the device interface. The DSS-SIM uses it to include some physical layer parameters.

The *Networking* module models the communications of the simulator. It contains the SpaceChannel and the SpaceNetDevice classes. They extend the NS-3 classes ns3::Channel and ns3::NetDevice, respectively. Each node of the simulator has one SpaceNetDevice attached. Note that the SpaceNetDevice is understood as the transceiver of a node. On the other hand, the SpaceChannel simulates a communications channel among ISLs. Figure 2 shows the structure of the SpaceChannel in a block diagram.

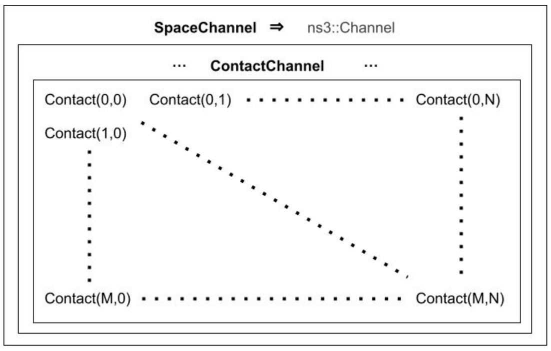


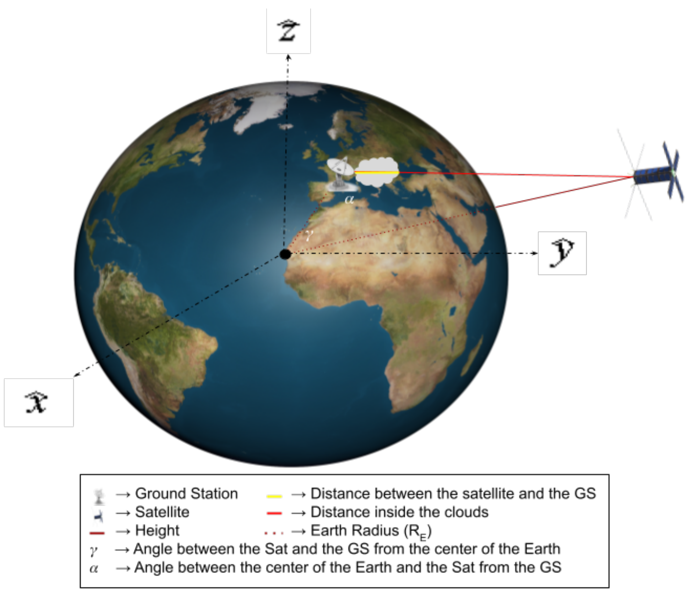
Figure 2: SpaceChannel structure.

The SpaceChannel consists of different ContactChannels, as shown in Figure 2. Each ContactChannel the contacts that work at the same data rate and frequency. Each Contact is defined as a physical and unidirectional link between two nodes. A simplified UML diagram of the networking module is given in Figure D.1 to understand the entire software structure of this module.

The *Networking* module allows the packet transmission. For this process, Figure D.2 provides a workflow diagram. The workflow structure can be defined as follows: (1) An event of transmission is scheduled in the simulation. The SpaceNetDevice calls the function *send* to start the transmission. (2) The *send* function calls the *propagatePacket* function from SpaceChannel class. Such a function looks at the different lists of ContactChannels, looking for the ContactChannel to which the sender belongs. (3) After finding the ContactChannel, it looks for the available contacts in the ContactChannel. Two different functions define such availability: *isFeasible* and *hasLineOfSight* from Contact. If any contact is available, the function *propagatePacket* allows the SpaceNetDevice to send the packet. (4) Then, the propagation is done.

3.1 Simulation of atmospheric effects on satellite communications

After introducing the *Networking* module, the design of the atmospheric effects class is presented. Firstly, the scheme of the challenge to be solved is provided in Figure 3. This problem involves two main characteristics: The first one is the definition of the attenuation of RF signal due to the Rayleigh Back-Scattering [11]; The second one is the geometric problem of finding the distance that the signal travels among the clouds (marked in yellow in Figure 3).



z
y
x
α
γ
→ Ground Station
→ Satellite
→ Height
→ Distance between the satellite and the GS
→ Distance inside the clouds
→ Earth Radius (R_E)
γ → Angle between the Sat and the GS from the center of the Earth
α → Angle between the center of the Earth and the Sat from the GS

Figure 3: Scenario of contact among a satellite and a GS where clouds must add an attenuation.

Atmospheric effects simulation on the communications channel is performed using [11], [15], [17] and [16]. In [11] the attenuation caused by the clouds on an RF signal is defined. Such a formula (Equation (1)) express the attenuation in dB per kilometre and relates the signal's frequency with the clouds' temperature.

$$\gamma_C(f,T) = K_l(f,T)M, \tag{1}$$

where:

- γ_C: Specific attenuation in [dB/Km].
- $K_l(f,T)$: Cloud liquid water specific attenuation coefficient in [(dB/Km)/(g/m^3)].
- M: Liquid water density in the cloud or fog [g/m^3].
- f: Frequency [GHz].
- T: Temperature of the cloud liquid water [K].

The attenuation model presented in [11] is only valid for those frequencies from 10[GHz] up to 200[GHz]. For this model, in particular, note that only those contacts among a satellite and a ground station with visibility greater than 10º are valid. This angle, which seems to be an arbitrary decision, resulted from the discussion with the thesis advisors. After defining the attenuation modelling, the other parameters used for such a model are defined.

- **Cloud liquid water-specific attenuation coefficient** $[K_l(f,T)]$: This parameter is obtained from [14]. The specific attenuation is defined by Equation (2) as follows:

$$K_l = \begin{cases} Water\ clouds\ and\ fog: & f^{a_1} e^{a_2(1+a_3T)}, \\ Ice\ clouds: & f^{b_1} e^{b_2(1+b_3T+b_4T^2)}, \end{cases} \tag{2}$$

where:

- $K_l(f,T)$: Cloud liquid water specific attenuation coefficient in [(dB/Km)/(g/m³)].
- f: Frequency [GHz].
- T: Temperature of the cloud liquid water [K].
- $a_1, a_2, a_3, b_1, b_2, b_3, b_4$: Benoit's constants defined in Table D.5.

- **Water liquid density of the cloud** M: It is also known as the Water Liquid Content of a cloud (WLC). It is defined after conducting some research on the topic. The merge of [15], [17], [16] is the result of such research. Table D.6 shows different kinds of clouds relating their thickness to their WLC.
- **Frequency**: The frequency is obtained from the SpaceNetDevice. It must be the frequency used for both nodes of the contact.
- **Temperature of the cloud**: This parameter shall be different depending on the scenarios. The user sets such a parameter.
- **Minimum frequency**: As this model is only valid for those frequencies from 10 up to 200 GHz, a minimum frequency is set. It sets a lower boundary. If the frequency is under 10 GHz, the model is not performed.
- **Distance**: The distance that the signal travels inside the cloud as shown in Figure 3.

The DSS-SIM provides the position of both nodes of the contact in ECI coordinates [34]. After obtaining the positions, trigonometry is used to relate vectors that unite the GS with the centre of the Earth and the GS with the Satellite. After this, the angle formed with such vectors is obtained. After obtaining the angle, the distance defined between the Satellite and the GS, it is trivial to obtain the distance that the signal travels among the cloud. Notice that the cloud is modelled as a rectangle. The formulas needed to obtain the angle and the distance are defined as follows:

$$\alpha = asen\left((R_{\text{Earth}} + h) \cdot \frac{sen(\gamma)}{d_{\text{body1-body2}}}\right) \tag{3}$$

$$d = acos(\alpha) \cdot h_{\text{clouds}} \tag{4}$$

After solving the challenge presented in Figure 3, and having a specific attenuation well-defined [11], the attenuation can be obtained by using Equations (4) and (1) as shown in Equation (5).

$$Att_{\text{dB}} = \gamma_c d \tag{5}$$

Once the methodology for the attenuation model has been found, it is time to implement a piece of software that meets the theoretical model found in Equation 5. This means that the software must implement the expressions 1, 3, 4 and 5. Also, it shall meet the development requirements of the simulator, which means that it must be compatible with NS-3 as long as the other propagation model used (Friis propagation model) is inherited from NS-3. To do so, Figures D.3 and D.4 two UML diagrams that implement the attenuation model to the DSS-SIM are shown. The main difference between both diagrams is how the Contact class computes the received power.

Analysing the behaviour of both proposals, it is clear that the final result must be the same. However, the process of adding attenuation to the signal will be different. For Figure D.3, the attenuation is computed independently in the PropagationLossAgregator Class. While in the case of Figure D.4 the attenuation is computed by the Implementation of the ns3::PropagationLossModel in CloudsPropagationModel. The computation of the attenuation in this second case of both Friis and Clouds propagation Models is possible due to the function ns3::AddNext. This function allows the concatenation of different propagation loss models. In Table 1, a comparison of some aspects taken into account at the time of choosing which path to follow at the time implement is shown. This table helps the reader understand both proposals' advantages and disadvantages.

Characteristic	Aggregator	NS-3 Integration
Easy to code	**X**	
Easy to understand		**X**
Best fitting with NS-3		**X**
DSS-SIM coding headlines		**X**
Best performance		**X**

Table 1: Differences between both proposals to integrate the clouds model.

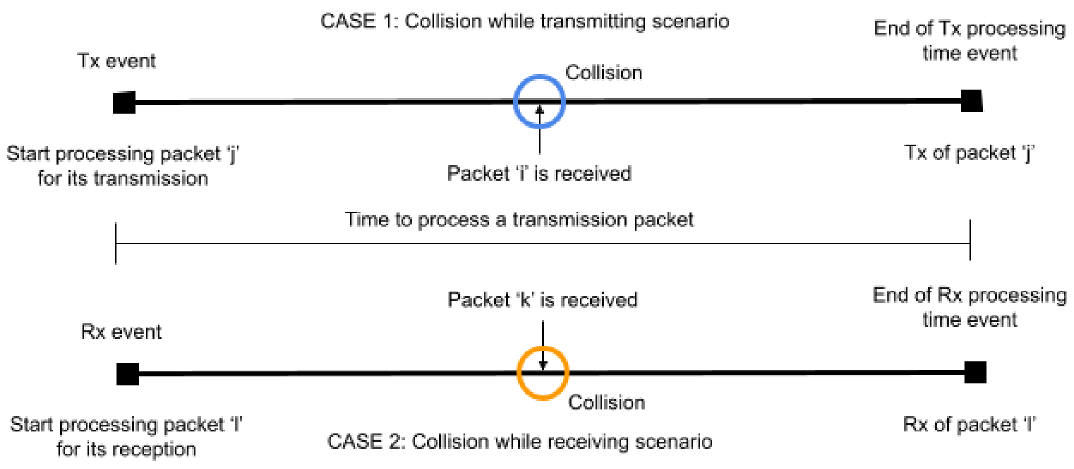
As seen in Table 1, even though the easy option would be the model presented in Figure D.3, the best option is the second one. The option presented in Figure D.4, has been the one implemented in this project due to its flexibility at the time to integrate new propagation losses models that the NS-3 community can develop. In Figure D.5, the UML focused on the attenuations is provided. Note that ns3::PropagationLossModel will be attached to the Contact class. Note that both will lead to the entire software design for implementing the atmospheric effects simulation model. The results of this implementation will be presented in the following chapter.

3.2 Transceiver simulation in ISL

Currently, there is no class in the DSS-SIM representing a transceiver. Despite this, the SpaceNetDevice contains some of the transceiver parameters. For this reason, a new class is created to represent a Transceiver. This class represents the hardware component and extends the current SpaceNetDevice parameters (e.g. consumption). The main functionalities of this class must be: (1) simulate collisions, (2) model the antenna parameters, (3) model the noise floor, (4) define bandwidth, (5) define a binary decider as a function of the SNR_{min}.

Collisions model

The collision nature is analysed to represent the packet collisions in the DSS-SIM. Figure 4 presents a scenario where a collision is done. Note that when regarding wireless communication, the transceiver is always half-duplex. A half-duplex system is presented in both cases of Figure 4.



CASE 1: Collision while transmitting scenario
Tx event
End of Tx processing time event
Collision
Start processing packet 'j' for its transmission
Packet 'i' is received
Tx of packet 'j'
Time to process a transmission packet
Packet 'k' is received
Rx event
End of Rx processing time event
Collision
Start processing packet 'l' for its reception
CASE 2: Collision while receiving scenario
Rx of packet 'l'

Figure 4: Temporal scheme of the collisions scenario scheme.

The algorithm proposal to achieve the behaviour shown in Figure 4 is presented in Figure D.6. Such an algorithm presents that if a collision happens, both packets are discarded. Now the cases presented in Figure 4 are presented.

CASE 1: A node wants to transmit a packet. When this node is processing the packet, another node tries to send a packet to it. Then a collision happens.
CASE 2: A node is receiving a packet. When this node is processing the received packet, another node tries to send another packet to it. When this occurs, a collision happens.

In both cases presented in Figure 4, an event of transmission is scheduled in the simulation. Once this event happens, the initial time is captured. After a period (that represents the processing time), the transmission ends and the time is recorded. With those times, the duration of the transmission is defined. Once the duration is known, the algorithm looks in the ContactChannel where the contact that sets the event belongs. If only any other contacts were trying to run an event of transmission during that time, both transmissions are invalid. Consequently, all packets are lost. The expression representing the collision function in Boolean is shown in Equation 6.

$$Collision = \begin{cases} 1 & Scheduling\ span < \ \Delta t \\ 0 & Otherwise \end{cases} \tag{6}$$

Antenna model

To represent the antenna parameters and the bandwidth, variables to represent such parameters will be created. Regarding the antenna parameters, the most important ones are (1) gain, (2) direction of the maximum lobe, and (3) beam width at −3dB and between the zeros.

It is necessary to mention that the attitude module in the simulator is missing by now. Such a lack makes this implementation challenging to test as it complements it. Note that the class that will use them is Contact at the time to compute the received power.

To accomplish the features expressed previously, Figures D.8 and D.7 present a simplified UML diagram of the Transceiver class. Three conclusions can be obtained:

- Collisions will be simulated in the SpaceChannel. The reason for this is that this class contains the ContactChannels (needed to search for the corresponding ContactChannel). Because of this, it is inherited from the SpaceNetDevice class that contains the transceiver.
- All parameters defined in the transceiver will be used in the contact to redefine the noise floor. By doing this, the noise floor contemplates the effects of such parameters instead of just assuming a specific sensibility (especially the noise introduced by the transceiver components and the antenna). Once this is done, the new binary decider defined by *isFeasible()* is:

  $$\gamma_{th} \geq \frac{P_{tx} G_{src} G_{rx}}{N_o} \left(\frac{\lambda}{4\pi d} \right)^2, \tag{7}$$

  where:

  - γ_{th}: Threshold
  - P_{tx}: Transmitted power
  - G_{tx}, G_{rx}: Transmission and reception gain (respectively)
  - d: The distance between two nodes of a contact
  - N_o: Noise defined as:

    $$N_o = KTB, \tag{8}$$

    * K: Boltzman's constant (1.3810^{-23} [J/K])
    * T: Temperature of the enviroment in Kelvin [K]
    * B: Bandwidth

  The function *isFeasible()* changes to achieve the behaviour described in Equation 7. By doing this, the binary decider dependent on the signal-to-noise ratio would be obtained. This means that as shown in D.7 the binary decider would be in the Contact Class.
- The bandwidth is not only defined to compute the noise factor. It can be used to improve the current behaviour of the simulation even more. After defining the bandwidth, it is needed to consider that from the DSS-SIM, the velocity vectors of the satellites and nodes, in general, can be obtained. With all these taken into account, and using Equation 9 from [35], the Doppler frequency is obtained. With this frequency added to the carrier, a loss in the transmission can distinguish. This fact simulates a rough problem that can be studied for ISL.

  $$f_D = \frac{\overrightarrow{V}_T \widehat{n}_{Tx-target}}{\lambda} \frac{\overrightarrow{V}_R \widehat{n}_{Rx-target}}{\lambda} \tag{9}$$

With all the information provided previously, the features expected to be designed to accomplish the objectives are now defined. The design of the software is found in Figures D.8 and D.7. Equation 6 must be implemented in the SpaceChannel to simulate the collisions. All the parameters that define a transceiver are now defined in a new class. That will cause the improvement of the binary decider can be implemented by using Equation 7 and even an implementation of the Doppler effect to set new losses in the transmission.

3.3 Spectrum aware channel simulation in ISL

To simulate spectrum-aware protocols, such as NB-IoT protocols, the need for a channel with spectrum characteristics raises on the DSS-SIM. To simulate the protocols, it uses the SpaceNetDevice, which is an access level layer, not a physical one (despite having some attributes of a physical layer). Nevertheless, NS-3 has implemented a class allowing the simulation of spectrum-aware communications. The classes of interest are:

- *SpectrumChannel* [36]: It defines the interface for spectrum-aware channel implementations. This class is not currently used in NS-3.
- *SpectrumPhy* [37]: Abstract base class for Spectrum-aware PHY layers. It can be understood as a NetDevice but focused on the physical layer. This class is not currently used in NS-3.
- *MultiModelSpectrumChannel* [38]: It is a SpectrumChannel implementation that can handle the presence of SpectrumPhy instances, which can use different spectrum models (E.g. different SpectrumModel). This class is not currently used in NS-3.
- *SpectrumModel* [39]: It represents a set of frequency values implementing the domain of the functions in the Function Space defined by SpectrumValue. This class is not currently used in NS-3.
- *SpectrumValue* [40]: Set of values corresponding to a given SpectrumModel. This class is not currently used in NS-3.

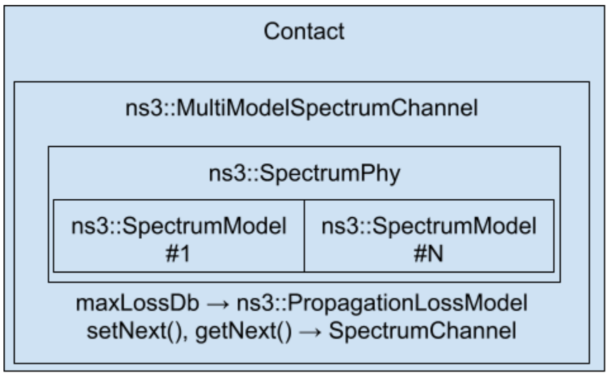

Contact
ns3::MultiModelSpectrumChannel
ns3::SpectrumPhy
ns3::SpectrumModel #1
ns3::SpectrumModel #N
maxLossDb → ns3::PropagationLossModel
setNext(), getNext() → SpectrumChannel

Figure 5: Scheme of a contact with spectrum parameters.

Figure D.9 shows how the spectrum-aware channel implementation. The NS-3 class MultiModelSpectrumChannel represents a contact, as shown in Figure 5. Each node (SpaceNetDevice) of the contact will need a ns3::SpectrumPhy. This will allow the SpaceNetDevice to use to MultiModelSpectrumChannel to propagate the packets. Notice that the MultiModelSpectrumChannel allows the use of different spectrum-aware implementations. This means that each protocol that uses a spectrum-aware channel. They can be redefined in the simulation to test them. It is important to note how this functionality has been included in the current simulator design. Despite being an easy-to-understand diagram, the simulation has adopted it does not mean that it will not increase the resources consumed by the simulator. It must be said that if for a specific simulation, the main target is not the spectrum-aware channels, this communications can be inactivated, allowing the simulation to work as it has until now by setting the fast-propagation mode. This will allow the implementation of different protocols, such as NB-IoT protocols, to be tested. Using its necessary mobility models, they must be set at the simulation definition. Also, the NetDevices should be defined to adapt them to the SpaceNetDevice.

By looking at the definition of those classes (see NS-3 documentation [10], [36], [37], [38], [39], [40]) the physical parameters such as Antenna parameters are found in these classes. Note that other parameters such as noise, and collisions shall be implemented as described in Section 3.2 in parallel to the integration of the spectrum channel. Those parameters must be included as SpaceNetDevice parameters (as being transceiver parameters) while the computations of the effects shall be done in Contact.

To conclude, the implementation of a spectrum-aware channel increases the cost of the simulation. Nevertheless, doing this must imply the implementation of many other protocols and modulation to the features that the simulation can perform. This cost-effective implementation is justified by the simulator's last goal, which is to test and simulate the communication protocols for ISLs. Also, by doing this, the creation of a transceiver is negligible., As most of the parameters are included in the ns3::SpecrtumPhy class. Those that miss in the SpectrumPhy are found in the SpaceNetDevice class. Both classes can define a transceiver in physical and medium access layers.

3.4 Contribution to the communications channel for a Satellite Contact Emulator

Design of a Satellite Contact Emulator

The emulator's design was firstly considered for prioritising flexibility and modularity. Such a design is based on interconnected SDRs with IP/Ethernet LAN. The architecture is presented in Figure 6.

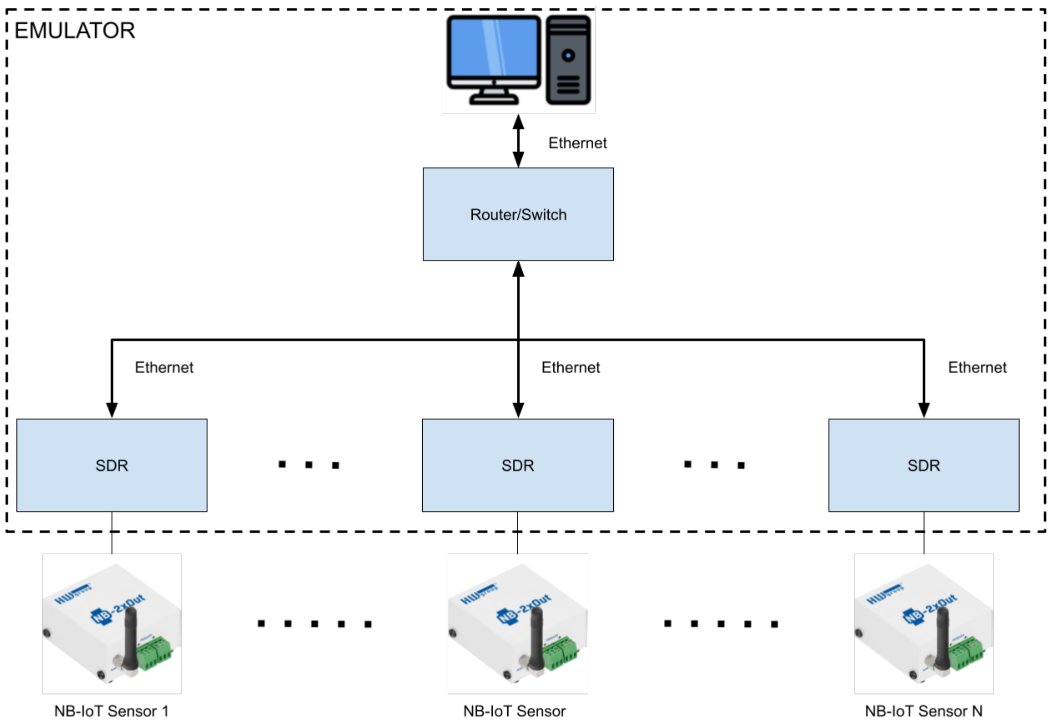


Figure 6: Initial scheme of the satellite contact emulator.

This design of the emulator must contemplate the RF connection between the sensors and the SDRs. Additionally, it shall interconnect the SDRs through Ethernet. This connection shall provide real-time behaviour with UDP. This choice is made to avoid blocking the communications.

The hardware used in the design outlined above is chosen. The switch is selected among the options presented in Table D.7. The chosen switch is the GS108PEv3 model from NETGEAR (Marked in green in Table D.7). The reasons to choose it are (1) its price and (2) features, such as the transmission speed.

Regarding the SDRs, Table D.8 presents different parameters that are used to perform the comparison. Such variables are (1) the FPGA, (2) the transceiver, (3) the number of connectors, (4) the MSPS, (5) the BW, (6) the frequency range, and (7) the price.

The selected devices are marked in Figure D.8. Such pieces of equipment are the PlutoSDR and the BladeRF 2.0 micro xA9. The reason for choosing two of them is to allow more flexibility in the development. Since the frequency range and the bandwidth between them are widely different. Also, another reason to choose two of them was to compare the behaviour of two different SDRs that not only change in terms of features but also in price. Such flexibility in having different devices can conclude in choosing one piece of hardware for the final design. The reason to choose the PlutoSDR as the most cost-effective device is due to its price and specs. It offers a reasonable bandwidth, and its frequency range goes from 325 MHz to 3.8 GHz. For the case of the BladeRF board, the reason was its higher bandwidth and frequency range (compared to the PlutoSDR), also its FPGA. Despite not being the most cost-effective board from BladeRF, its FPGA performance was worth it.

After defining the used SDRs, it is time to analyse the libraries provided by hardware manufacturers. For the PlutoSDR, the library provided by Analog Devices is called Libiio. It is compatible with several languages such as C, C++ and Python. For this project, the programming language employed is C because the main basic applications and tutorials were made in C. For the case of bladeRf, the library is called libbladeRF. This compatibility is quite similar to the Libiio. Thus, the C programming language can also be used. Notice that at the time of coding the software for both SDRs, each library shall be used for its specified hardware (Libiio for the PlutoSDR and libbladeRF for the BladeRF 2.0 micro xA9).

Option 1 - Interconnected SDRs with IP/Ethernet LAN

To prove the feasibility of the design presented in Figure 6, a testbed is developed. Its structure must be as shown in Figure 7. Note that this testbed consists of a switch and two PlutoSDRs. This means that at the time to develop the necessary software, Libiio is used. Additionally, in Figure 8 the architecture of the testbed is presented. Whereas in Figure 9 the testbed setup is shown.

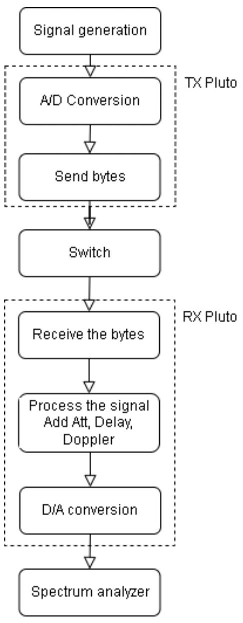


Figure 7: Testbed workflow.

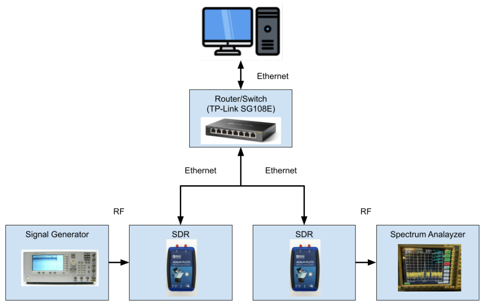


Figure 8: Test-bed with two SDR

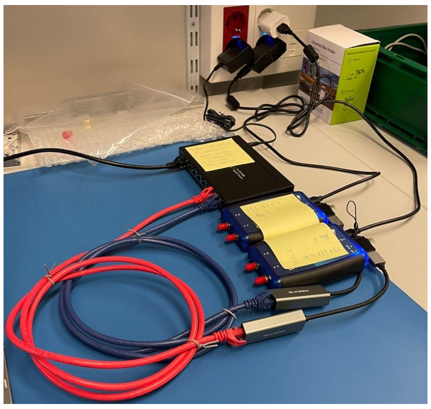


Figure 9: Testbed setup

The software developed to meet with the testbed proposed in Figure 7 implements: (1) The definition of the transmission and reception channels plus the definition of the SDR parameters, (2) the reception of RF signal and the use of the ADC to obtain the sampled signal, (3) the connectivity with other SDRs using UDP protocol via Ethernet, (4) The signal processing before transmitting the signal, (5) The updating of the parameters used by the SDRs to compute the channel parameters. It means connectivity with the central computer and (5) the conversion and transmission of the signal using the DAC and the transmission port.

To achieve (1), Libiio is used [41], [42]. The main parameters to set are the LO frequency, the sampling rate, the ports to be used, and the bandwidth. Also, the transmission and reception channels, and the IIO buffers, as their context are defined. Once the parameters are set, and the channels and the buffer are defined, the device is ready to start receiving and transmitting. A workflow diagram is given in Figure D.10 to understand such software behaviour

To receive RF signals and fill the IIO Buffer (2), the function IIOBufferRefill [43] is used.

A simple Client-Server structure is implemented to allow the devices to establish the connectivity described in (3). The SDR that shall read the signal from the signal generator will behave as the client. Whereas the SDR that shall transmit the signal to be red by the spectrum analyser is taking the role of the server. UDP is used to simulate the real-time data transmission. The C class networking has been used when coding such implementation to have the native send() and receive() functions.

Emulating the communications channel requires some signal processing (4). The channel parameters obtained from the central computer are used to emulate a communication channel behaviour. The algorithm to implement is shown in Figure 10. When simulating attenuation, the IQ samples are multiplied. However, to simulate the Doppler effect, the LO is used to tune the frequency at the corresponding one. Regarding the delay, a product to the resulting phasor is done. Note that a delay can not be implemented by using a function such as wait(), or the entire emulation would collapse due to delay.

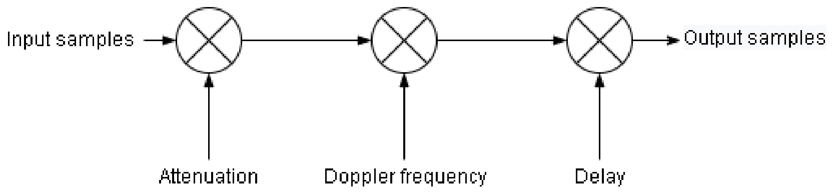


Figure 10: Signal processing diagram.

To implement the connectivity with the central computer to refresh the channel parameter (5), a Client-Server structure is used. The “client” role is given to the SDRs and the “server” role to the central computer. Unlike the implementation presented in (2), TCP is used to ensure the refreshing of the parameters.

Allowing the transmission of an RF signal from the SDR the IIOBufferPush [44] function is used. Such function reads the IIO Buffer and transmits the data through the RF port.

After implementing the software, some tests were performed to verify its functionality. They are presented in section 4.2. Once they were performed, there were some issues to be solved.

The first one was the occurrence of a current peak. After some more testing to make that pike disappear, it was found that to obtain a flat noise floor, 40 dB of attenuation needed to be added after the transmission port of the SDR. However, adding those 40 dB means they have to be compensated at some other point. That gain can be compensated by using the reception gain parameter of the SDR. Nevertheless, when the test was performed again, the results were incoherent. The output signal was much more attenuated than expected. Another test was performed to study the relationship between the attenuation and the input power. The results of those tests concluded that the best gain to be added in the Rx of the SDR was 50 dB. When performing this test, it was found that the IQ values saturated at a certain point. As the range of powers used went from -30 dBm to -90 dBm, the ADC was studied. It was found that both the ADC and DAC only used 12 bits for each sample [45]. In Figure 11 the IQ codification performed by the converters is illustrated.

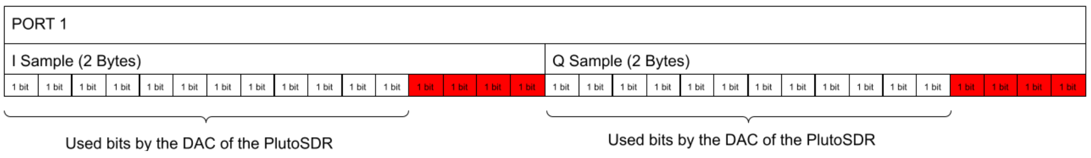

PORT 1
I Sample (2 Bytes)
Q Sample (2 Bytes)
1 bit
Used bits by the DAC of the PlutoSDR
Used bits by the DAC of the PlutoSDR

Figure 11: IQ codification structure.

Once the incoherence related to the SDR behaviour was solved, the test presented in Figure 7 was performed. The first result obtained was **unsatisfactory**. Only some spurious pikes appeared. In the beginning, it was thought to be interference from other signals. However, after tuning the frequency, the peaks were still there. After observing the limits of those peaks, the sweep time of the spectrum analyser was increased up to 5 s. The result was the transmitted signal sampled. A conclusion is obtained: The Ethernet network that connects the SDRs, the buffering time, and the data rate of the devices is generating such a delay that only some parts of the signal are being transmitted.

Option 2 - Design based on a single SDR

A new design is proposed to evaluate the communications channel emulation and the connectivity with the central computer. It consists of a single SDR. Such a design must demonstrate the feasibility of the Satellite Contact Emulator. The switch is removed from the setup to avoid delays related to the network. By eliminating the connectivity among the SDR, the remaining device must be able to perform the reception, processing and transmission while refreshing the channel parameters from the central computer. In Figure 12 the workflow of this testbed is shown. Additionally, Figure 13 provides the testbed architecture, while in Figure 14 the testbed is shown.

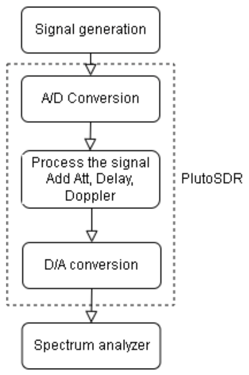


Figure 12: Second testbed workflow.

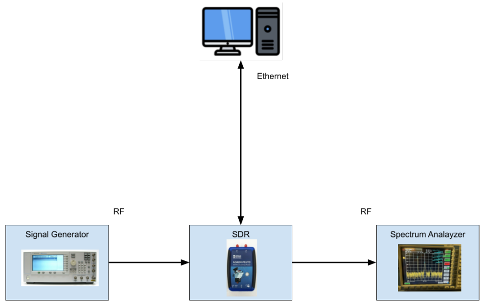


Figure 13: Testbed with one SDR

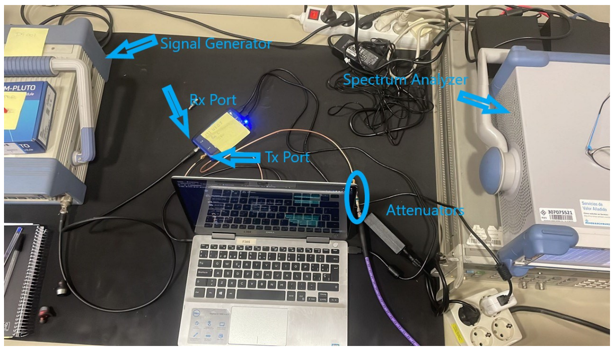


Figure 14: Testbed setup

To achieve the testbed proposed in Figure 12 the software must consider: (1) The definition of channels and the buffers, (2) The use of the ADC to fill the IIO buffer, (3) The signal processing to simulate the channel, (4) The connectivity with the central computer to refresh the channel parameters, and (5) The use of the DAC to read the IIO buffer and transmit the data through the RF port.

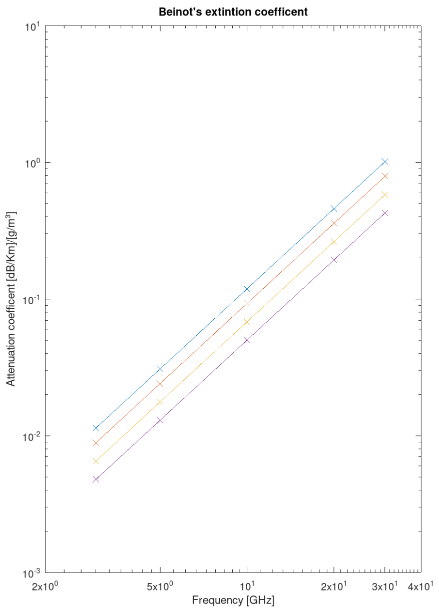
The software implementation for this testbed is very similar to the one proposed for the testbed presented in Figure 8. The main differences are the definition of the channels and the buffers (1), as they are now defined in the same SDR. As explained in the previous testbed proposal, the other aspects will remain the same. Except for the connectivity among the SDR, it is not considered in this testbed.

4 Experiments and results

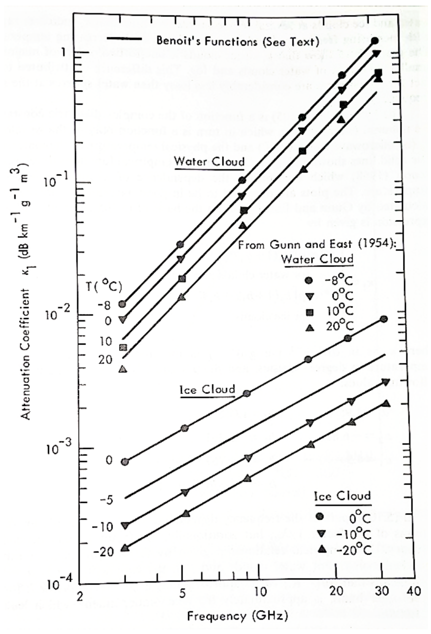
4.1 Results obtained by testing the atmospheric model

A testing tool is used to test and validate the results obtained from the software development. In this case, passing a series of G-Test [46] was mandatory to accept the implementation of the code to be merged in the DSS-SIM. These tests are unit ones and verify each function of the implemented classes.

The first thing done to validate the results was to pick the model described by Ulaby in [14] and represent the same plot as presented in Figure 15b. Below, in Figure 15a, the plot obtained is presented. To validate such a plot, it is compared to the one provided by Ulaby. The test is based on the results obtained from the model obtained using Octave. Such software obtains more accurate values than the implementation done in C++. If both results (C++ and Octave) are pretty similar (Accepting an error among them about 10^{-6} times lower), the tests will be considered to be passed successfully.



(a) Implemented model.



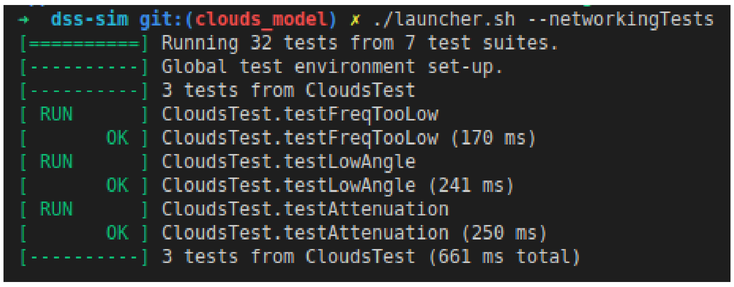
(b) Ulaby's model.

Figure 15: Ulaby's model vs implemented model.

To validate the results obtained when implementing the clouds model, three main tests have been done. In Figure 16 the screenshot of the passed test is shown. The tests had consisted on:

- Checking the working frequency of the source. Different contacts were created with different working frequencies on their SpaceNetDevices. If the frequency was lower than 10GHz (passed value), the model should not be performed. All tests were passed ***successfully***.
- Check the conditions of the contact to check the function that assures the model's validity. For this particular case, different planet points are chosen and translated into the point of a LEO orbit. Those points were the West coast of Australia and a point of the ocean near New Zealand, obtained by an orbit propagator that propagated the orbit of the International Space Station (ISS). Meanwhile, the GS for all the tests is located in Barcelona. It was expected not to have visibility among the satellite and the GS. So no model should be applied as far as no transmission must occur. All test were passed ***successfully***.
- Checking the values of the attenuation obtained by the model implemented. The attenuation algorithm is recreated to compute the attenuation in MATLAB and

plot the attenuation coefficients' values. The idea was to check the values obtained by the Matlab script with the reference of the model [14]. So the results obtained in Figure 15a were meant to be equal or very similar to the ones presented in [14] in Figure 15b. As can be seen, both figures are quite similar. After obtaining those results, the same code used in C++ was re-adapted to compute the distance the signal meant to travel. Moreover, with those values, its product was compared to the output of the C++ code. The tests were performed for different frequencies and different temperatures. AL test were passed ***successfully***.



```
→ dss-sim git:(clouds_model) ✗ ./launcher.sh --networkingTests
[==========] Running 32 tests from 7 test suites.
[----------] Global test environment set-up.
[----------] 3 tests from CloudsTest
[ RUN      ] CloudsTest.testFreqTooLow
[       OK ] CloudsTest.testFreqTooLow (170 ms)
[ RUN      ] CloudsTest.testLowAngle
[       OK ] CloudsTest.testLowAngle (241 ms)
[ RUN      ] CloudsTest.testAttenuation
[       OK ] CloudsTest.testAttenuation (250 ms)
[----------] 3 tests from CloudsTest (661 ms total)
```

Figure 16: Screenshot of the passed tests.

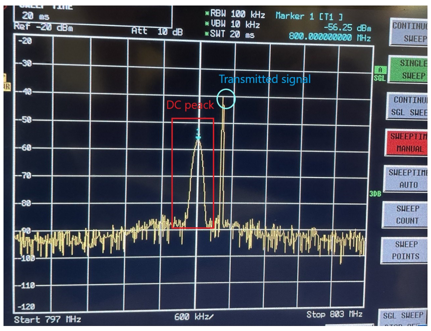
4.2 Results obtained by testing the emulator

To obtain valid results in the emulator's development, several tests have been done. The most important and representative ones are the following ones:

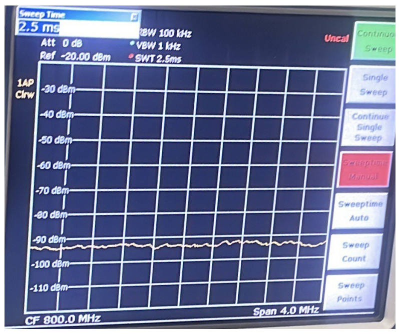
1. Verification of the flat noise floor.
2. Verification of the noise introduced by the PlutoSDR.
3. Verification of the Doppler effect by obtaining a shifted signal.
4. Verification of the attenuation by obtaining an attenuated signal.
5. Verification of the Doppler effect plus the attenuation simultaneously by obtaining a signal shifted and attenuated.
6. Verification of the bandwidth effect when the carrier deviates from its original frequency.

Elimination of the peak

To obtain a flat noise floor and eliminate the current peak, a PlutoSDR is simultaneously connected to a signal generator and the spectrum analyser (1). A current peak is seen without introducing any attenuation, as shown in Figure 17a. The first attempt to attenuate the signal is by using the attenuator on the Tx port of the PlutoSDR. However, nothing happens when attenuating the signal. The next attempt consists of adding attenuators to the transmission port. When a signal is transmitted, now the peak is reduced. After some iterations tuning the value of the attenuator, it is found that the value for the peak to disappear is 40 dB. The result is shown in Figure 17b



(a) Capture with 0 dB attenuation.



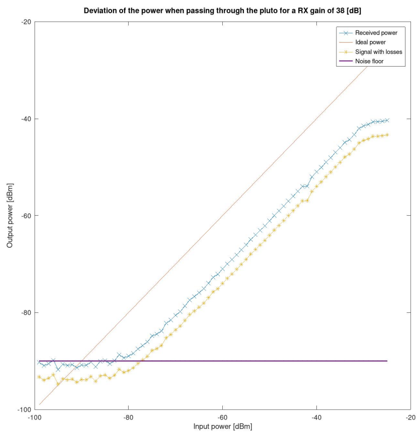
(b) Capture with 40 dB attenuation.

Figure 17: Comparison between attenuating the output port or not.

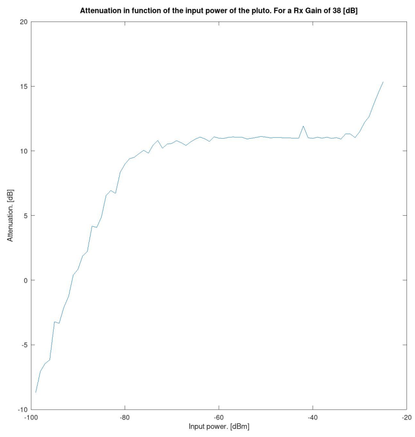
Noise introduced by the pluto

It is noted that the input power values when tuning it do not match the output power. It either corresponds to the input value -40 dB. So another test is conducted to obtain the optimal value of the Tx gain value and the noise introduced by the SDR (2). The test consisted of changing the gain on the reception port and tuning the input power to note the received power in the spectrum analyser. By doing this, the maximum gain is obtained experimentally.

By knowing that 40 dB of attenuation had been added, a priory, the gain introduced should be 40 dB. However, when reading the output in the spectrum analyser, the received power was not -60 dBm; it was -80 dBm. Such values mean that the PlutoSDR introduced 20 dB of attenuation. A test is performed to find the best gain, even though it is expected to be around 60 dB. Figures 18, 19, 20 and 21, presents the relationship between the gain, and the attenuation.

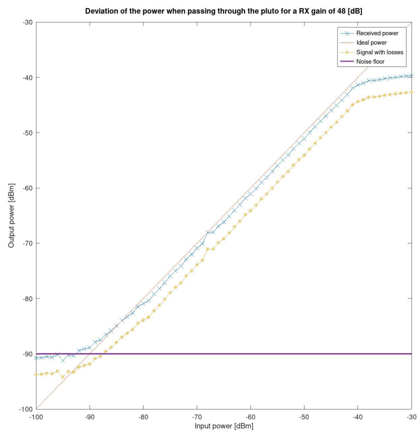


(a) Rx gain equal to 38 dB: Output power vs input power.

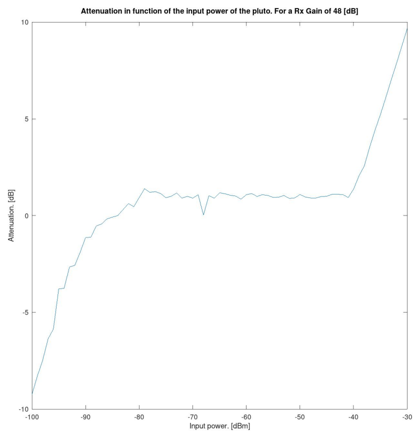


(b) Rx gain equal to 38 dB: Attenuation vs input power.

Figure 18: Results of the working margin for an Rx gain of 38dB.

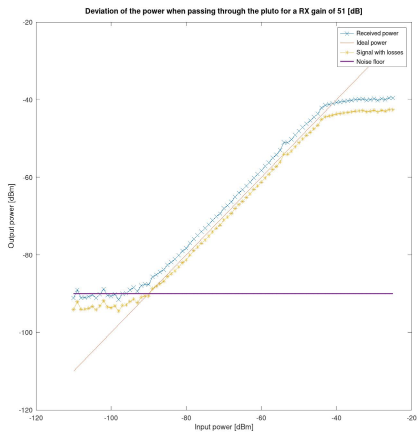


(a) Rx gain equal to 48 dB: Output power vs input power.

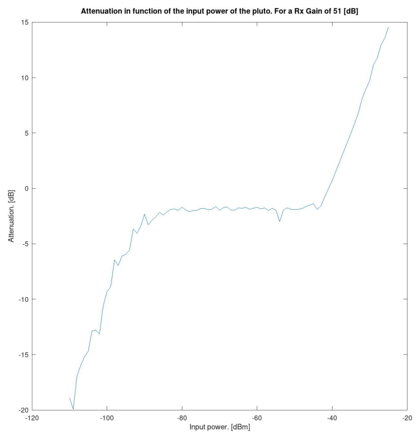


(b) Rx gain equal to 48 dB: Attenuation vs input power.

Figure 19: Results of the working margin for an Rx gain of 48dB.

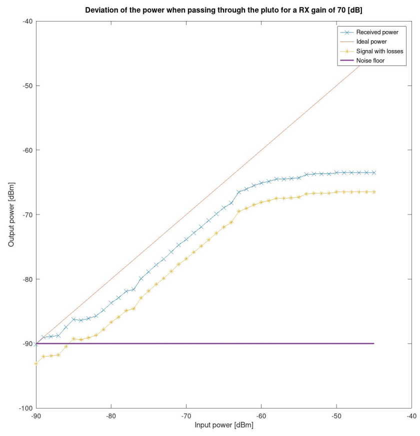


(a) Rx gain equal to 51 dB: Output power vs input power.

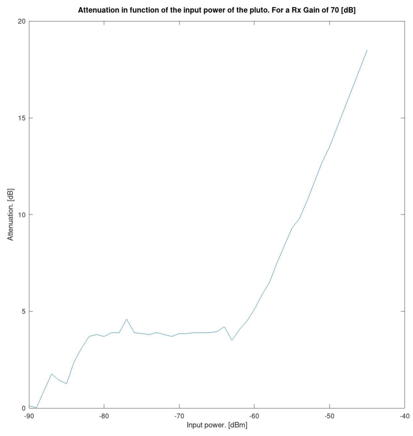


(b) Rx gain equal to 51 dB: Attenuation vs input power.

Figure 20: Results of the working margin for an Rx gain of 51dB.



(a) Rx gain equal to 70 dB: Output power vs input power.



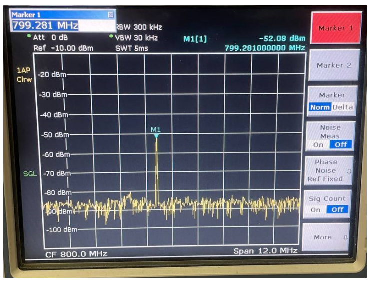
(b) Rx gain equal to 70 dB: Attenuation vs input power.

Figure 21: Results of the working margin for an Rx gain of 70dB.

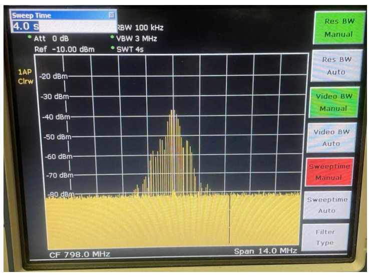
As it can be seen in Figures 19 and 20, the best choice for the gain in reception is a value between 48 and 51 dB. Nevertheless, for the sake of simplicity, $G_{Rx} = 50$ dB is chosen. To ensure a linear relationship. By observing those figures, it can be said that the operating range in terms of power goes from -90 dBm to -40 dBm. Such values mean that the input signal at the SDR ports shall be attenuated from its original value to at least -40 dBm. At the same time, any signal under -90 dBm will be understood as noise. This limitation is not only introduced by the hardware as it has already been said in Section 3.4 both the ADC and DAC work with 12 bits. Each in-phase and quadrature word is encoded with that number of bits. The codification of each word has a direct impact on the operating range. As it has experimented, the values saturate at around -35 dBm. From this value, the attenuation can not be greater.

Option 1 - Testing

At the time to test the attenuation (4) on the testbed proposal from Figure 8 the output was not satisfactory. The results obtained are shown in Figure 22. It was not attenuation that failed. It was the capture of the signal. As seen in 22a, only one signal peak was captured. As shown in Figure 22b, the entire signal is captured. However, the signal is not well captured. By analysing this behaviour, it can be concluded that the delay introduced for both the Ethernet network and the buffers of the SDRs is too high.


Marker 1
799.281 MHz
RBW 300 kHz
Att 0 dB
VBW 30 kHz
Ref -10.00 dBm
SWT 5ms
M1[1] -52.08 dBm
799.281000000 MHz
CF 800.0 MHz
Span 12.0 MHz

(a) Capture with a sweeptime of 5 ms


Sweep Time
4.0 s
RBW 100 kHz
Att 0 dB
VBW 3 MHz
Ref -10.00 dBm
SWT 4s
CF 798.0 MHz
Span 14.0 MHz

(b) Capture with a sweeptime of 4 s

Figure 22: Received signals using the test-bed presented in Figure 8

The conclusion obtained from Figure 22 is that the Ethernet network and the buffers introduced such a delay on the testbed that the signal is not correctly transmitted. Note that the signal cannot be captured unless the sweep time is increased.

Option 2 - Testing

From the conclusion obtained in Figure 22, the testbed proposed in Figure 13 was developed to test the channel emulation.

The results obtained by testing the **attenuation** (4) in this second testbed are presented in Figure 23. Here, applying an attenuation to the received signal is obtained. The reference signal equals the received signal without attenuation (left) and the attenuated signal by 3 dB (right). Note that this test is passed ***sucessfully***.

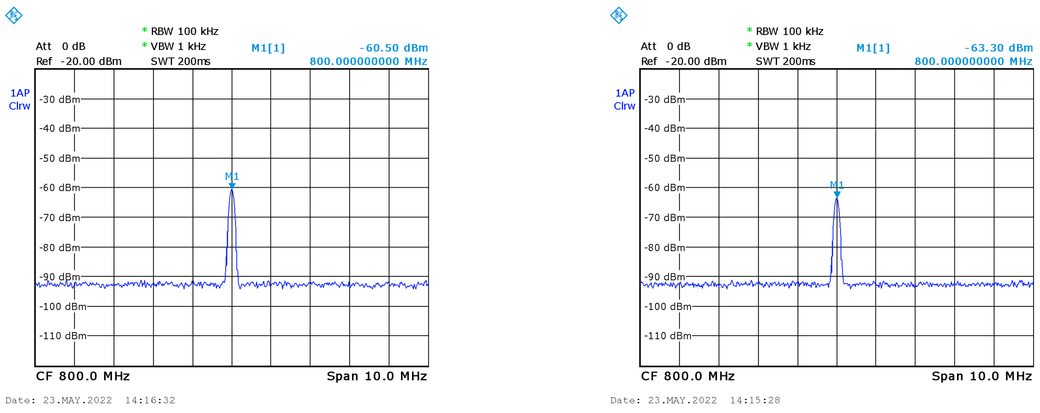


Figure 23: Attenuation between 2 signals at the same frequency.

Before testing the Doppler effect itself (3), it is important to see how the **bandwidth** affects the signals (6). A bandwidth of 2 MHz was defined in the SDR. Then three signals were introduced separately. The first signal was set at 800MHz. The next two corresponded to the signal at 801 MHz and 799 MHz. The same test was performed twice by reducing the input signal by 3 dB to re-check the attenuation introduced. See Figures 24 and 25 to see the display of the spectrum analyzer in the case of the different power inputs (not attenuated signal in Fig. 24 and a signal attenuated in Fig. 25).

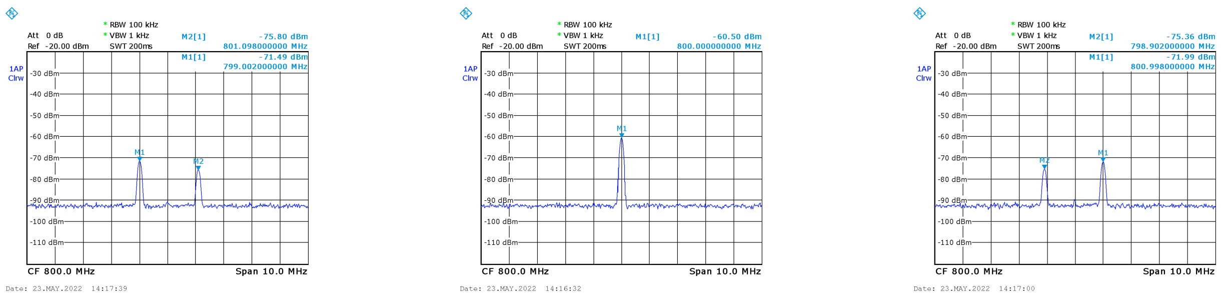

M2[1] -75.80 dBm 801.098000000 MHz
M1[1] -71.49 dBm 799.002000000 MHz
M1[1] -60.50 dBm 800.000000000 MHz
M2[1] -75.36 dBm 798.902000000 MHz
M1[1] -71.99 dBm 800.998000000 MHz
M1
M2
CF 800.0 MHz
Span 10.0 MHz
Date: 23.MAY.2022 14:17:39
Date: 23.MAY.2022 14:16:32
Date: 23.MAY.2022 14:17:00

Figure 24: Bandwidth test results with an input signal not attenuated.

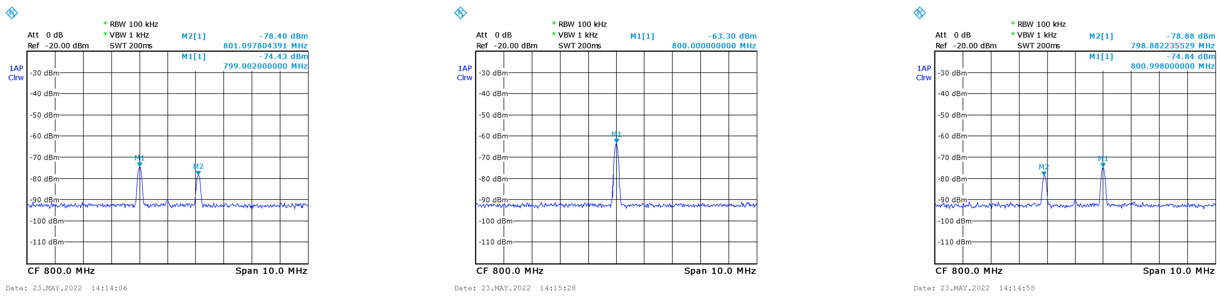


Figure 25: Bandwidth test results with an input signal attenuated 3 dB.

By analysing them, it can be said that the Bandwidth is properly set. The input signal was a sinusoidal signal, and its Fourier transform should be understood as two pulses. Once centred in the carrier frequency and its mirrored signal. The power spectral density of both pulses should be equal to one that was not affected by the bandwidth. Although the naked eye can lie if both pulses were integrated, the result of such an integration would be the same as the one obtained by integrating the reference pulse. Notice that the results were ***satisfactory***.

The next feature to test on the emulation was to develop a **Doppler effect** (3) implementation in the SDR. As explained in 3.4 the Doppler effect was represented by a change on the local oscillator of the board. Nevertheless, it is crucial to note that thin can cause inferences due to the electronic devices performing signal processing. In Figure 26 the tests' results are provided to compare the different signals when the frequency changes. Also, Figure 27 provides the results when the Doppler effect is applied at the same time as attenuation is affecting.

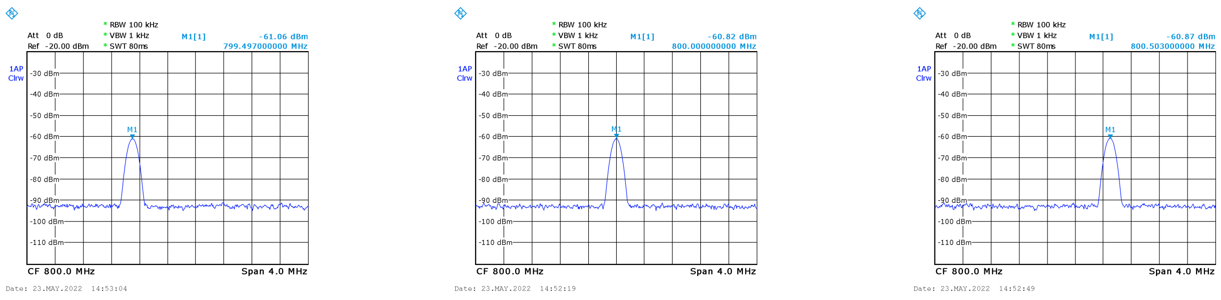


Figure 26: Doppler effect test results.

Figure 26 concludes that the Doppler effect has been successfully achieved after presenting the results. Not only the Doppler itself.

Finally, the combination of the **Doppler effect** plus the **attenuation** (5) is tested. Figure 27 presents the results of such test. As can be seen in such figure, the results obtained were ***successfully***.

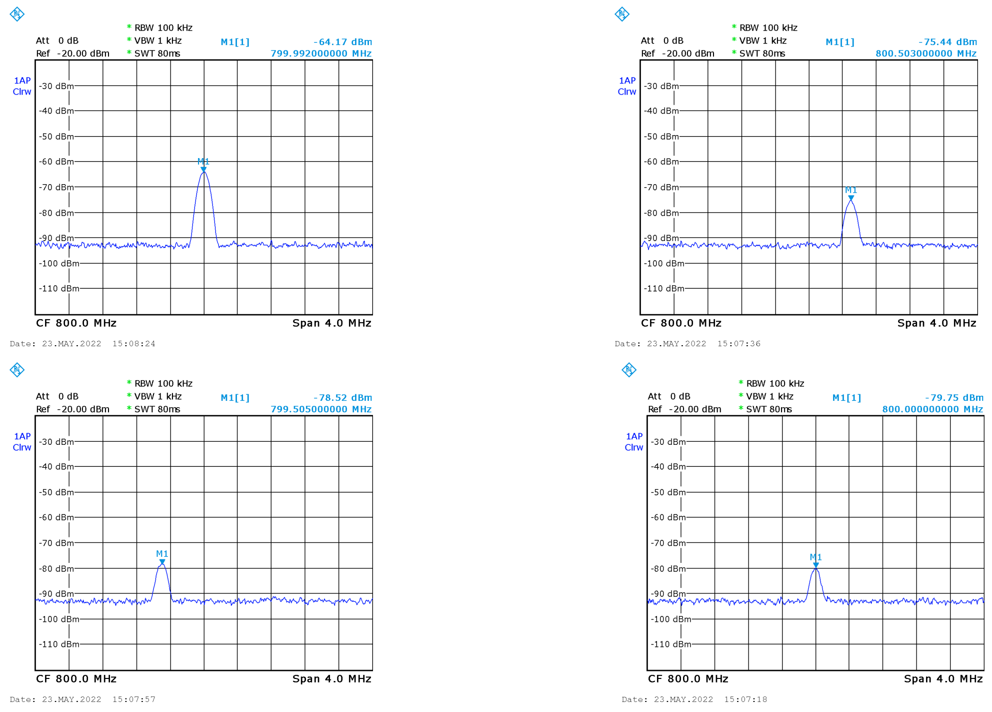


Figure 27: Doppler plus attenuation test results.

5 Conclusions and future work

The main conclusion that can be obtained is that this project was too ambitious. It was expected to achieve many results and implementations in a little time. Nevertheless, it is part of the learning procedure to re-adapt the strategy to achieve the objectives. Even by prioritizing those that are more critical than others by evaluating their complexity and impact on the project. Also, it is important to mention that although this project was meant to contribute to the simulation and emulation of ISL, it has opened the window to possible future work. On the other hand, more specific conclusions on the developed parts of this project are presented. However, the first thing to be introduced is the relationship between this thesis's objectives and achievements.

Objective	**Achieved**	**Not achieved**
Atmospheric effects model	**X**	
Transceiver		**X**
Spectrum-aware channel		**X**
Testbed of a satellite contact emulator	**X**	

Table 2: Objectives achievement.

Regarding the achievements of the contribution to the DSS-SIM, in Table D.9 a more detailed description is provided. However, below, a brief analysis of them is given.

Conclusions can be obtained from developing a model to retrieve the attenuation caused by the atmospheric effects. The first one is that all tests were passed successfully. Nevertheless, only the attenuation caused by clouds has been implemented. Not the scintillation effects nor other effects had been taken into account. This stands as a future work, as well as to compute the attenuation caused by the atmosphere between two satellites. Finally, as this model is now implemented, is concluded that developing future atmospheric models will be easier.

Regarding the design presented in both Sections 3.2 and 3.3, it can be concluded that the design to be used to implement the transceiver shall not be the one proposed in the Section 3.2. It must be more profitable in terms of performance to implement those parameters that represent a transceiver in the implementation of the spectrum-aware channel. It is essential to mention that those implementations could not be tested due to a lack of time. Such lack of time is because, during the project's development, the ns3::MultiModelSpectrumModel was found and studied to compare its features and how it could benefit the DSS-SIM. As a result, this thesis proposes and describes the design of a spectrum-aware channel. However, it stands as a future work proposal to develop such an implementation.

In terms of emulation, a detailed analysis of the requirements fulfilment is found in Table D.10. Despite not achieving the original architecture proposal because of the Ethernet delays, a proof of concept was developed. It has been successfully at the time to emulate a communications channel. Which means that it stands as future work to develop the escalation of the emulator. Also, such escalation proposal is demonstrated to be feasible as seen in [24]. Note that the project described in [24] was thought to have 30 months. Regarding the duration of the mentioned project, the developed testbed has been quite an achievement. Taking into account the issues found during the development of the first testbed proposal described in 3.4.

Note that despite only achieving testing half of the objectives proposed for this thesis due to a lack of time. It can be said that this contribution to the emulation and simulation to ISL has: (1) Achieved the implementation of atmospheric effects in the DSS-SIM will also make easier future propagation models implementations. (2) Achieved the design of a spectrum channel into the DSS-SIM. Such design allows for the implementation of this class which will add several features to the simulator. (3) Achieved to prove the feasibility of a cost-effective Satellite Contact Emulator (even more cost-effective than the solution proposed in [24]) by designing a test-bed to emulate a communication channel.

To conclude, it is crucial to understand the implications that both projects have. Besides the DSS-SIM and the Satellite Contact Emulator are robust simulation and emulation tools separately, from [25] the combination has been demonstrated to be feasible. Moreover, a robust testing tool can be obtained. Not just in terms of testing network protocols by simulating them, they can use accurate signals to be passed through the communications channel implemented in the emulator to test its feasibility in real hardware. Moreover, it can lead to a tool capable of testing the entire communications system of mission that concerns a FSS by ensuring the ISLs and the Up-link and Down-Link contacts in real time before its deployment.

References

[1] Giovanni Giambene, Sastri Kota, and Prashant Pillai. Satellite-5g integration: A network perspective. *IEEE Network*, 32(5):25–31, 2018.

[2] THALES ALENIA SPACE. Integrating satcom and 5g: challenges and solutions. https://www.sat5g-project.eu/wp-content/uploads/2020/03/SaT5G-WHITE-PAPER-ABOUT-RESEARCH-PILLARS-Release-version-final-version.pdf.

[3] J.A. Ruiz De Azúa Ortega. *Contribution to the development of autonomous satellite communications networks : the internet of satellites.* Tesi doctoral, UPC, Departament d'Enginyeria Telemàtica, http://hdl.handle.net/2117/346653, 2020.

[4] Joan A. Ruiz de Azúa, Anna Calveras, and Adriano Camps. Internet of Satellites (IoSat): Analysis of Network Models and Routing Protocol Requirements. *IEEE Access*, 6:20390–20411, 2018.

[5] Joan A. Ruiz-de Azúa, Carles Araguz, Anna Calveras, Eduard Alarcón, and Adriano Camps. "Towards an integral model-based simulator for autonomous earth observation satellite networks". In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 7403–7406, 2018.

[6] Antonio Romero. Contributions to satellite subsystem models and communications link emulation. Bachelor's thesis, UPC, 2022.

[7] MATLAB. Satellite communications toolbox. https://www.mathworks.com/products/satellite-communications.html.

[8] University of Luxembourg. Satellite communication system simulator. https://wwwen.uni.lu/snt/research/sigcom/sw_simulators/satsim.

[9] J. Puttonen, S. Rantanen, F. Laakso, and J. Kurjenniemi. Satellite network simulator 3. In *Workshop on Simulation for European Space Programmes (SESP)*, volume 24, page 26, 2015.

[10] NS-3 Community. NS-3 Documentation. https://www.nsnam.org/doxygen/.

[11] ITU Radiocommunication Assembly. Recommendation ITU-R P.840-8 (08/2019) Attenuation due to clouds and fog. https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.840-8-201908-I!!PDF-E.pdf.

[12] ITU Radiocommunication Assembly. Recommendation ITU-R P.618-13 (12/2017) Propagation data and prediction methods required for the design of Earth-space telecommunication systems. https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.618-13-201712-I!!PDF-E.pdf.

[13] ITU Radiocommunication Assembly. Recommendation ITU-R P.676-12 (08/2019) Attenuation by atmospheric gases and related effects. https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.840-8-201908-I!!PDF-E.pdf.

[14] Fawwaz T. Ulaby, Richard K. Moore, and Adrian K. Fung. *Microwave Remote Sensing Active and Passive.* Artech House, Norwood, Massachusetts, 1981.

[15] Anne Thomson. Simulating the adiabatic ascent of atmospheric air parcels using the cloud chamber. https://en.wikipedia.org/wiki/Liquid_water_content.

[16] Anom Prasetio and Aulia Nasution Bambang L. Widjiantoro. Overview of ground-based generator towers as cloud seeding facilities to optimize water resources in the Larona Basin. https://www.researchgate.net/publication/331785464_Overview_of_ground-based_generator_towers_as_cloud_seeding_facilities_to_optimize_water_resources_in_the_Larona_Basin.

[17] Weather Forecast Office. Cloud classification and characteristics. https://www.weather.gov/lmk/cloud_classification.

[18] Others Nicola Baldo, Marco Miozzo. NS-3 LENA NB. https://github.com/tudo-cni/ns3-lena-nb/blob/master/doc/source/lte-design.rst.

[19] J. C. Merlano-Duncan, J. Querol, L. Martinez-Marrero, J. Krivochiza, A. Camps, S. Chatzinotas, and B. Ottersten. Sdr implementation of a testbed for synchronization of coherent distributed remote sensing systems. In *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 6588–6591, 2020.

[20] J. Querol, J. C. Merlano-Duncan, L. Martinez-Marrero, J. Krivochiza, S. Kumar, N Maturo, A. Camps, S. Chatzinotas, and B. Ottersten. A cubesat-ready phase synchronization digital payload for coherent distributed remote sensing missions. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 7888–7891, 2021.

[21] Keysight Technologies. Propsim f64 5g channel emulation solution - f8800a. https://www.keysight.com/us/en/assets/7018-07057/brochures/5992-2962.pdf.

[22] C. A. Hill. Space communications emulation facility, 2004. https://ntrs.nasa.gov/citations/20050186847.

[23] A. Hill Chante. Space link extension (sle) emulation for high-throughput network communication. https://ntrs.nasa.gov/api/citations/20140017058/downloads/20140017058.pdf.

[24] M. Djurica Tahir Ali Raza, Harri Saarnisaari and K. Briggs. Testbed data centre equipment installed, connected and functionally tested and validated / validation of 5g control and user plane harmonisation – validation setups and test results. https://www.sat5g-project.eu/wp-content/uploads/2019/04/SaT5GD5.5_final_v1.0.pdf.

[25] Airbus D&S SAS. Real-Time Satellite Network Emulator. https://artes.esa.int/projects/realtime-satellite-network-emulator.

[26] European Space Agency. Emulator of Satellite-Terrestrial 5G Radio Channels HYBRID CHANNEL EMULATOR FOR COMBINED SATELLITE-

TERRESTRIAL 5G CHANNELS. https://artes.esa.int/projects/emulator-satelliteterrestrial-5g-radio-channels.

[27] Vijay K. Garg. Chapter 23 - Fourth Generation Systems and New Wireless Technologies. In Vijay K. Garg, editor, *Wireless Communications & Networking*, The Morgan Kaufmann Series in Networking, pages 23–1–23–22. Morgan Kaufmann, Burlington, 2007. https://www.sciencedirect.com/science/article/pii/B9780123735805500570.

[28] GNU-Radio project. What is GNU Radio?, 2022. https://www.gnuradio.org/about/.

[29] Michael Hennerich. What is Libiio?, 2022. https://wiki.analog.com/resources/tools-software/linux-software/libiio.

[30] Nuand LLC. BladeRF documentation, 2019. https://nuand.com/bladeRF-doc/libbladeRF/v2.2.1/.

[31] NS-3 Community. ns3::propagationlossmodel. https://www.nsnam.org/doxygen/classns3_1_1_propagation_loss_model.html.

[32] NS-3 Community. ns3::channel. https://www.nsnam.org/doxygen/classns3_1_1_channel.html.

[33] NS-3 Community. ns3::netdevice. https://www.nsnam.org/doxygen/classns3_1_1_net_device.html.

[34] Wikipedia Foundation Inc. Earth-centered inertial. https://en.wikipedia.org/wiki/Earth-centered_inertial.

[35] William Emery and Adriano Camps. *Introduction to Satellite Remote Sensing Atmosphere, Ocean, Cryosphere and Land Applications*. Elsevier, 2017.

[36] NS-3 Community. ns3::spectrumchannel. https://www.nsnam.org/doxygen/classns3_1_1_spectrum_channel.html.

[37] NS-3 Community. ns3::spectrumphy. https://www.nsnam.org/doxygen/classns3_1_1_spectrum_phy.html.

[38] NS-3 Community. ns3::multimodelspectrumchannel. https://www.nsnam.org/doxygen/classns3_1_1_multi_model_spectrum_channel.html#details.

[39] NS-3 Community. ns3::spectrummodel. https://www.nsnam.org/doxygen/classns3_1_1_spectrum_model.html.

[40] NS-3 Community. ns3::spectrumvalue. https://www.nsnam.org/doxygen/classns3_1_1_spectrum_value.html.

[41] LibIIO Contributors. Libiio api. https://analogdevicesinc.github.io/libiio/master/libiio/index.html.

[42] Terry Carpenter. Ad9361 high performance, highly integrated rf agile transceiver™ linux device driver. https://wiki.analog.com/resources/tools-software/linux-drivers/iio-transceiver/ad9361.

[43] LibIIO Contributors. Iio buffer refill. https://analogdevicesinc.github.io/libiio/master/libiio/group__Buffer.html#gac999e5244b5a2cbbca5ecaef8303a4ff.

[44] LibIIO Contributors. Iio buffer push. https://analogdevicesinc.github.io/libiio/master/libiio/group__Buffer.html#gae7033c625d128667a56cf482aa3149bd.

[45] Travis Collins. Ad-fmcomms2/3/4/5 basic iq datafiles. https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz/software/basic_iq_datafiles#binary_format.

[46] Google. Googletest user's guide. https://google.github.io/googletest/.

[47] University of Michigan. Life cycle of a computer. https://sustainablecomputing.umich.edu/knowledge/life-cycle.php.

[48] Nowtricity. Co2 emissions for kwh in spain. https://www.nowtricity.com/country/spain/.

Appendices

A Management

In this appendix, the planning and its deviations are presented. It is structured in two sections. The first one will present the original Gantt diagram plus its work packages. The second one will show the deviation with the modified Gantt diagram and the incidences.

A.1 Original Planning

At the beginning of this project, the objectives to achieve were marked. Once the objectives were defined, it was time to organize the planning in order to achieve such objectives within the given time. The objectives to achieve for this project are:

- Implementing atmospheric effects on the DSS-SIM
- Define a transceiver that simulates collisions of packets, the antenna parameters (such as the gain and the radiation pattern), the noise floor and the bandwidth in the DSS-SIM.
- Implementing a spectrum-aware communications channel model for the DSS-SIM.
- Contribute to creating an end-to-end testbed using SDRs to emulate ISLs dynamics and channel.
- Implement a communications channel model to perform the attenuation, the Doppler effect and the delay.

To achieve such objectives, the work breakdown structure is provided in Figure A.1.

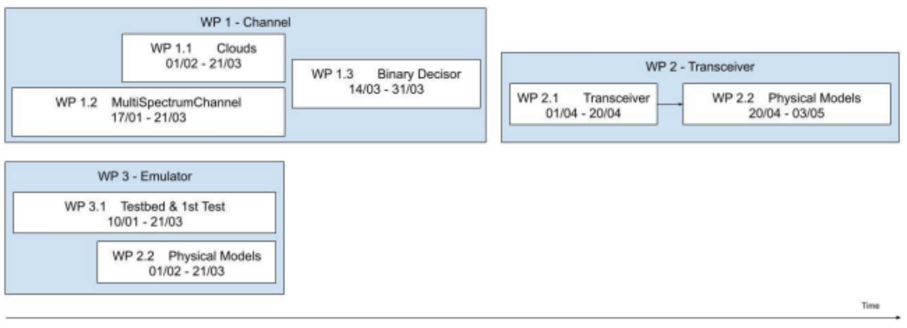
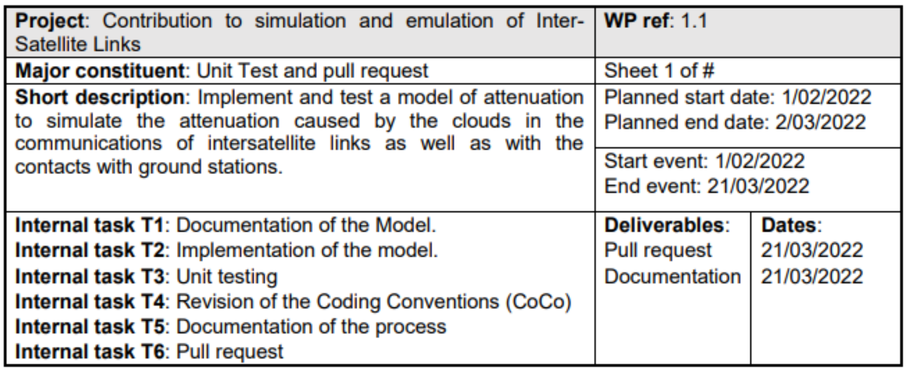

WP 1 - Channel
WP 1.1 Clouds 01/02 - 21/03
WP 1.3 Binary Decisor 14/03 - 31/03
WP 1.2 MultiSpectrumChannel 17/01 - 21/03
WP 2 - Transceiver
WP 2.1 Transceiver 01/04 - 20/04
WP 2.2 Physical Models 20/04 - 03/05
WP 3 - Emulator
WP 3.1 Testbed & 1st Test 10/01 - 21/03
WP 2.2 Physical Models 01/02 - 21/03
Time

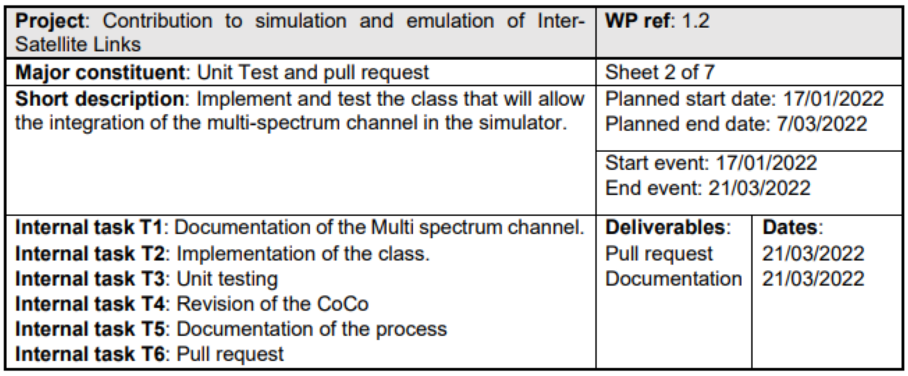
Figure A.1: Work Breakdown Structure Diagram.

The work packages are shown below:



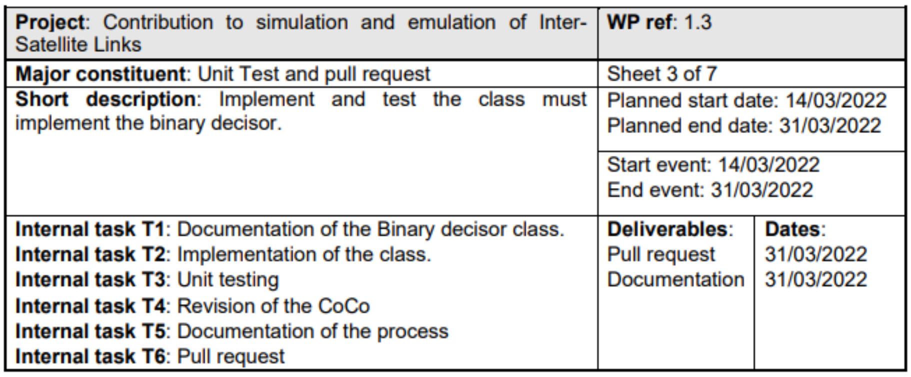
Project: Contribution to simulation and emulation of Inter-Satellite Links	**WP ref**: 1.1	
Major constituent: Unit Test and pull request	Sheet 1 of #	
Short description: Implement and test a model of attenuation to simulate the attenuation caused by the clouds in the communications of intersatellite links as well as with the contacts with ground stations.	Planned start date: 1/02/2022 Planned end date: 2/03/2022 Start event: 1/02/2022 End event: 21/03/2022	
Internal task T1: Documentation of the Model. **Internal task T2**: Implementation of the model. **Internal task T3**: Unit testing **Internal task T4**: Revision of the Coding Conventions (CoCo) **Internal task T5**: Documentation of the process **Internal task T6**: Pull request	**Deliverables**: Pull request Documentation	**Dates**: 21/03/2022 21/03/2022

Figure A.2: Work Package 1.1.



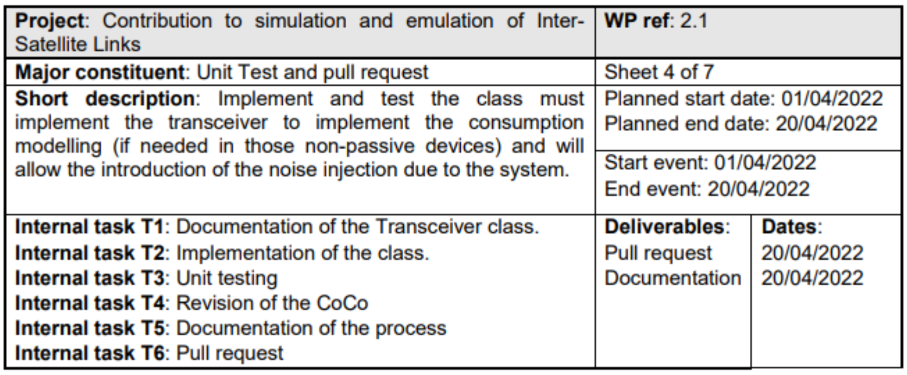
Project: Contribution to simulation and emulation of Inter-Satellite Links	**WP ref**: 1.2	
Major constituent: Unit Test and pull request	Sheet 2 of 7	
Short description: Implement and test the class that will allow the integration of the multi-spectrum channel in the simulator.	Planned start date: 17/01/2022 Planned end date: 7/03/2022 Start event: 17/01/2022 End event: 21/03/2022	
Internal task T1: Documentation of the Multi spectrum channel. **Internal task T2**: Implementation of the class. **Internal task T3**: Unit testing **Internal task T4**: Revision of the CoCo **Internal task T5**: Documentation of the process **Internal task T6**: Pull request	**Deliverables**: Pull request Documentation	**Dates**: 21/03/2022 21/03/2022

Figure A.3: Work Package 1.2.



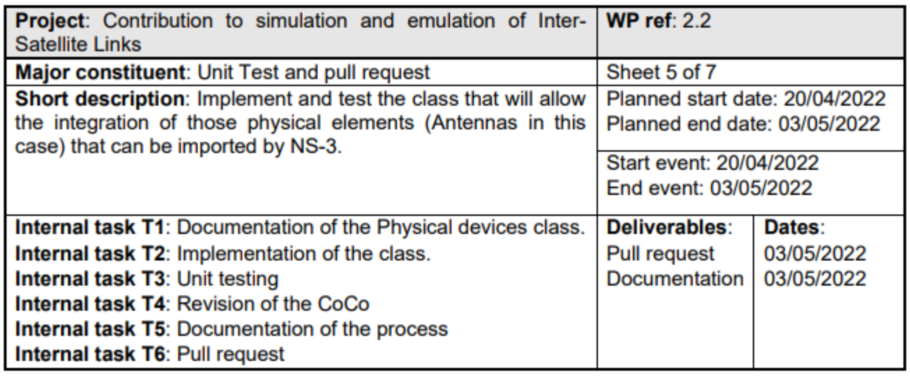
Project: Contribution to simulation and emulation of Inter-Satellite Links	**WP ref**: 1.3	
Major constituent: Unit Test and pull request	Sheet 3 of 7	
Short description: Implement and test the class must implement the binary decisor.	Planned start date: 14/03/2022 Planned end date: 31/03/2022 Start event: 14/03/2022 End event: 31/03/2022	
Internal task T1: Documentation of the Binary decisor class. **Internal task T2**: Implementation of the class. **Internal task T3**: Unit testing **Internal task T4**: Revision of the CoCo **Internal task T5**: Documentation of the process **Internal task T6**: Pull request	**Deliverables**: Pull request Documentation	**Dates**: 31/03/2022 31/03/2022

Figure A.4: Work Package 1.3.



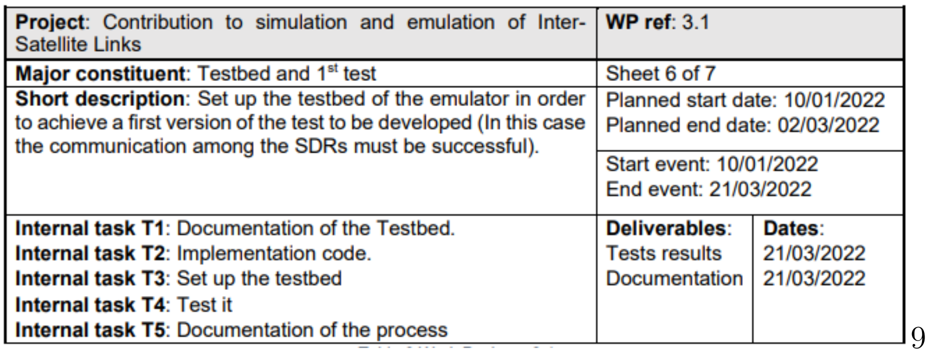
Project: Contribution to simulation and emulation of Inter-Satellite Links	**WP ref**: 2.1	
Major constituent: Unit Test and pull request	Sheet 4 of 7	
Short description: Implement and test the class must implement the transceiver to implement the consumption modelling (if needed in those non-passive devices) and will allow the introduction of the noise injection due to the system.	Planned start date: 01/04/2022 Planned end date: 20/04/2022 Start event: 01/04/2022 End event: 20/04/2022	
Internal task T1: Documentation of the Transceiver class. **Internal task T2**: Implementation of the class. **Internal task T3**: Unit testing **Internal task T4**: Revision of the CoCo **Internal task T5**: Documentation of the process **Internal task T6**: Pull request	**Deliverables**: Pull request Documentation	**Dates**: 20/04/2022 20/04/2022

Figure A.5: Work Package 2.1.



Project: Contribution to simulation and emulation of Inter-Satellite Links	**WP ref**: 2.2	
Major constituent: Unit Test and pull request	Sheet 5 of 7	
Short description: Implement and test the class that will allow the integration of those physical elements (Antennas in this case) that can be imported by NS-3.	Planned start date: 20/04/2022 Planned end date: 03/05/2022	
	Start event: 20/04/2022 End event: 03/05/2022	
Internal task T1: Documentation of the Physical devices class. **Internal task T2**: Implementation of the class. **Internal task T3**: Unit testing **Internal task T4**: Revision of the CoCo **Internal task T5**: Documentation of the process **Internal task T6**: Pull request	**Deliverables**: Pull request Documentation	**Dates**: 03/05/2022 03/05/2022

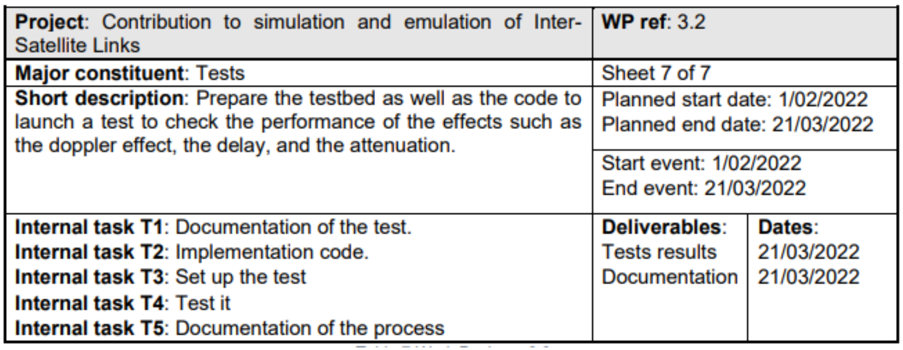
Figure A.6: Work Package 2.2.



Project: Contribution to simulation and emulation of Inter-Satellite Links	**WP ref**: 3.1	
Major constituent: Testbed and 1st test	Sheet 6 of 7	
Short description: Set up the testbed of the emulator in order to achieve a first version of the test to be developed (In this case the communication among the SDRs must be successful).	Planned start date: 10/01/2022 Planned end date: 02/03/2022	
	Start event: 10/01/2022 End event: 21/03/2022	
Internal task T1: Documentation of the Testbed. **Internal task T2**: Implementation code. **Internal task T3**: Set up the testbed **Internal task T4**: Test it **Internal task T5**: Documentation of the process	**Deliverables**: Tests results Documentation	**Dates**: 21/03/2022 21/03/2022

9

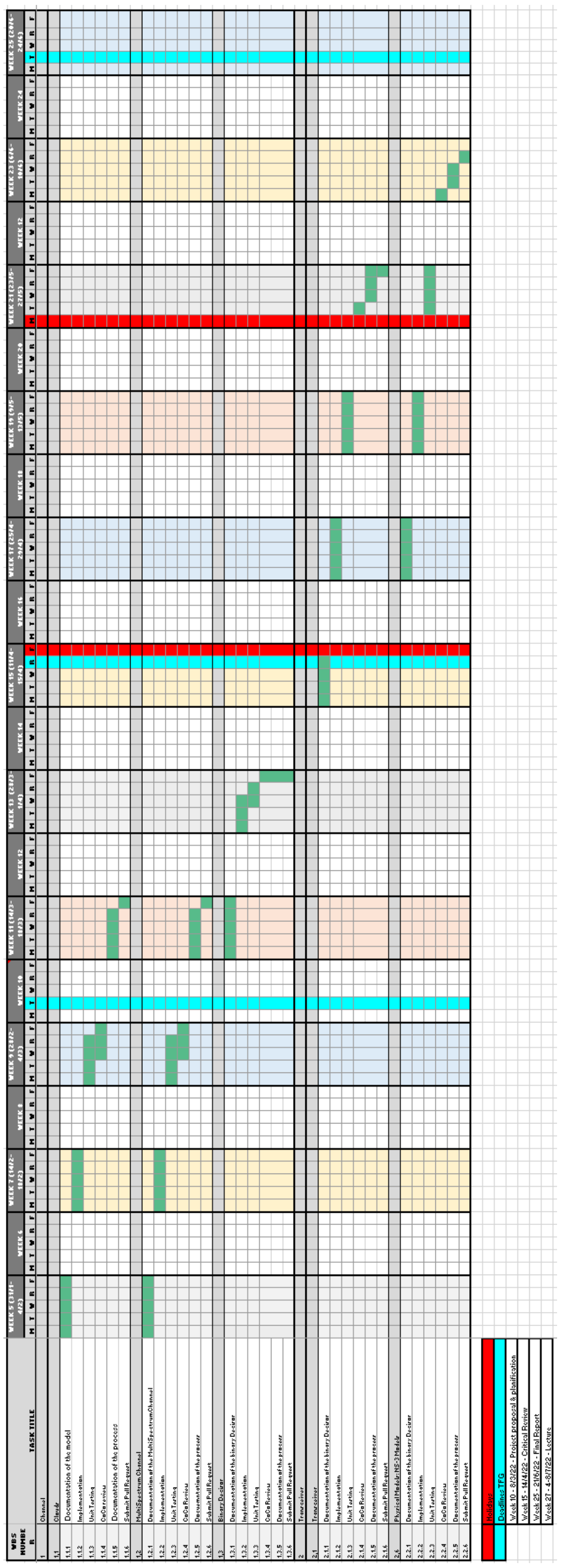
Figure A.7: Work Package 3.1.



Project: Contribution to simulation and emulation of Inter-Satellite Links	**WP ref**: 3.2	
Major constituent: Tests	Sheet 7 of 7	
Short description: Prepare the testbed as well as the code to launch a test to check the performance of the effects such as the doppler effect, the delay, and the attenuation.	Planned start date: 1/02/2022 Planned end date: 21/03/2022	
	Start event: 1/02/2022 End event: 21/03/2022	
Internal task T1: Documentation of the test. **Internal task T2**: Implementation code. **Internal task T3**: Set up the test **Internal task T4**: Test it **Internal task T5**: Documentation of the process	**Deliverables**: Tests results Documentation	**Dates**: 21/03/2022 21/03/2022

Figure A.8: Work Package 3.2.

After having presented the work packages, it is time to show the Gantt diagram that will allow the comprehension of the planning more visually. On the following page, the original Gantt diagram is provided.


(a) DSS-SIM Initial Gantt Diagram

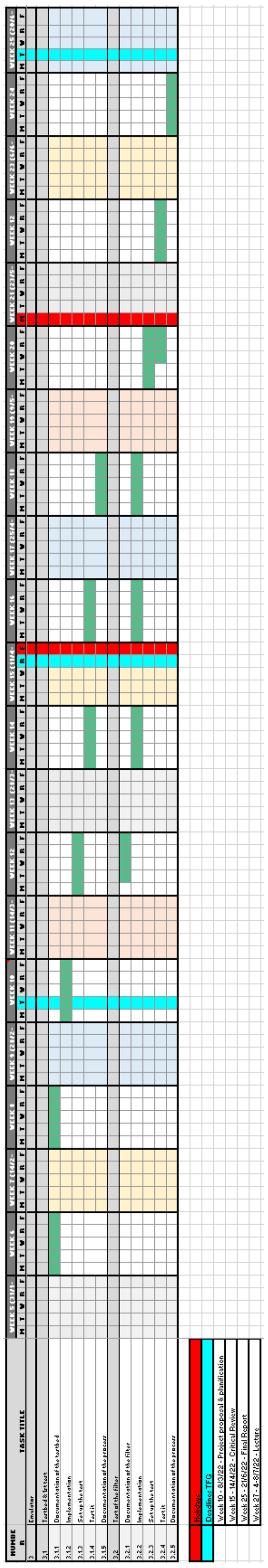

(b) Emulator Initial Gantt Diagram

Figure A.9: Gantt Diagram of the Thesis.

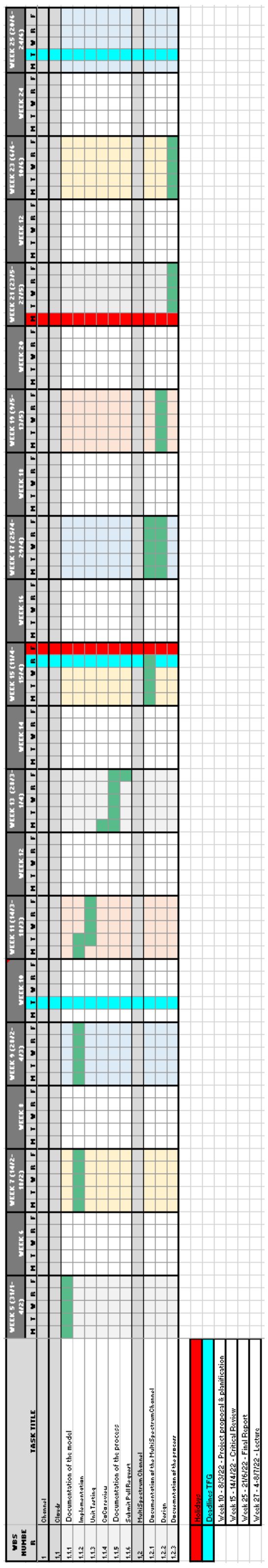
A.2 Deviations

Now, the analysis of the deviations will be made. First of all there are differences between Figure A.9 and Figure A.10. Those differences are due to technical issues that had appeared during the project. Now some of them are going to be pointed out in order to justify such delay.

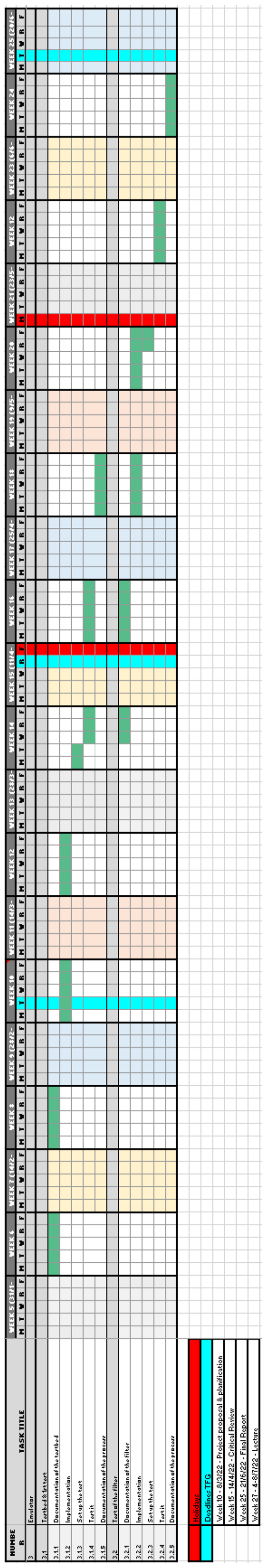
- Find out that ns3::MultiModelSpectrumChannel can handle physical attributes corresponding to the transceiver. It means that the reason for implementing a transceiver has no more sense.
- The appearance of an unexpected peak related to the LO that could not be removed without some testing.
- The delay observed due to the Ethernet that did not allow us to see the whole signal and led to wrong conclusions at the beginning.

Note that these issues made the project change some objectives. The most notorious ones are changes implementing the design of the MultiModelSpectrum and the skip of the transceiver class that corresponded to WP2. Regarding the Emulator, the most notorious change has been not obtaining a testbed that should be a prototype of a satellite contact emulator. Instead, a testbed with just one SDR was made to prove that the channel was being emulated correctly. This fact was due to the delays on the Ethernet network. From this, it can be concluded that work packages 1.3 (Figure A.4), 2.1 (Figure A.5), and 2.2 (Figure A.6) are skipped. Finally, the Gantt diagram will be shown in Figure A.10. As seen in Figure A.10 in both parts of this thesis, there has been a delay. Such delay has resulted from the technological problems found during the project. Due to the lack of time resulting from this delay, the objectives were re-defined to achieve them. The final objectives to achieve are:

- Implementing atmospheric effects on the DSS-SIM
- Design a spectrum-aware communications channel model for the DSS-SIM.
- Implement a communications channel model to perform the attenuation, the Doppler effect and the delay.
- Demonstrate the feasibility of the Satellite Contact Emulator by emulating a channel using a central computer and a SDR.



(a) DSS-SIM Final Gantt Diagram



(b) Emulator Final Gantt Diagram

Figure A.10: Gantt Diagram of the Thesis.

B Budget

To identify this thesis's costs, it is necessary to estimate each cost separately. Two main costs can be identified: the human resources cost, which implies the money dedicated to cover the salary, and the hardware and software-related costs.

Regarding the cost of employment for this thesis, the main point is the salary. As far as i2Cat uses a time-track system called "*Tempo*" that can be found in the management tool "*Jira*" the estimation of the cost in terms of salary can be easily distinguished. As far as this project was developed as a part of an internship, the salary per hour is 9€/h.

Hours	**Cost per hour(€)**	**Amount (€)**
540	9	4860
Contributions	-	1458
Total Amount (€)	6316	

Table B.1: Salary estimation.

From table B.1 the amount of the salary can be obtained. However, from this, a 30% (€1458) must be added to contemplate the costs for the company to employ a student. So the total amount to employ a student is €6318.

Now that the human resources costs have been computed, it is time to estimate the costs of the hardware and the software used to develop this thesis. In Tables B.2, and B.3 the hardware and software costs are provided. It is essential to note that the computer borrowed by the company was from 2016, and its current value has reached its residual value of it.

Hardware	**Price (€)**
PlutoSDR	183.52 x 2
Switch	95
Spectrum analyzer	250
Signal Generator	250
Total amount	962.04

Table B.2: Hardware costs estimation.

Both prices from the Spectrum Analyzer and the Signal generator rental are approximated. The usage of such equipment results from a cooperation agreement between the NanoSat Lab at UPC and i2Cat. Regarding the software, it is essential to note that the subscription to some software licences is priced monthly, and as far as the duration of this project has been about four months, they must be computed four times.

Software	Licence price (€)
Google services	5.7 * 4
Jire services	15 * 4
Visual Studio	0
Git services	0
Overleaf	0
Diagrams.io	0
Octave	0
Linux OS	0
Total amount	82.8

Table B.3: Software pricing estimation.

From Tables B.1, B.2, and B.3, Table B.4 is obtained. In this table the total estimation of the costs is provided.

Hours	Cost per hour(€)
Salary	6316
Hardware	962.04
Software	82.8
Total amount	7,360.84

Table B.4: Total budget estimation.

To conclude, the budget of this project, once all the costs, human, software and hardware have been considered, is €7,360.84. Note that such an amount would have been higher if the salary contemplated for this thesis would have been the salary corresponding to a junior engineer, which is higher than the one doing an internship.

C Environmental impact

Despite being a software-orientated thesis, this project used some hardware. Nevertheless, not only the hardware has a direct impact on the environment. The fact is that depending on the use of the software can also have an impact. However, this project has not led to using an internet-based application that can cause a high environmental impact due to the electricity consumption of all the infrastructure. This fact is one of the hidden facts about the use of technology nowadays.

Going into deeper detail on the environmental impact of this project, it is not easy to make an approach to such impact due to the lack of similar applications and its consumption and use time. However, an attempt is made to provide an idea about the impact that the project could cause on the environment.

First of all, a revision of the impact of the hardware used. The impact of a computer, a switch, and two SDRs will be provided regarding the emulator. According to the university of Michigan [47], the average annual consumption of a computer is 5,600 MJ (MegaJoules). However, only the 34% of life cycle energy consumption occurs in the use phase of a computer. The other 66% occurs during the mining, manufacturing, packaging, and transportation processes. This means that if a computer has an average lifespan of 5 years, only during this time would have consumed 28000 MJ. However, 54.353 MJ are generated to allow users to use their computers. This means that a computer generates 82000 MJ during its life. Now, the impact in g of CO2 is provided by the data from C02/KWh in Spain in 2020. Converting the 82000 MJ into KWh, it is obtained 22777.78 KWh. From [48] using 175g/KWh (of CO2), which was the average value from 2020, it is obtained that the total consumption of CO2 for a computer is 3.99 Tones of CO2. The same procedure for the switch and the SDRs. Since they are simple devices, it can be deduced that their consumption and energy at the time to manufacture and the package would be less it is assumed that it would be a 30% of a computer. From this in Table, a summary of the consumption is provided. Notice that it has been thought that the consumption only affects for four months. Below, in Table C.1 the CO2 generated from the hardware used in one part of this project is provided.

Device	**Manufacturing (MJ)**	**Consumption (MJ)**	**Total (KWh)**	**Total (CO2)**
Computer	54.353	933.3	15357.31	2687.53 Kg
SDR x 2	36235.3	622	10238.14	1791.67 Kg
Switch	18117.67	311	5119.07	895.84 Kg
Total amount of CO2: 5375 Kg				

Table C.1: Minimum Amount of CO2 generated for the emulator.

As seen in Table C.1 the minimum amount of CO2 for four months of use of the hardware is 5375 Kg of CO2. Moreover, this is if everything was made in Spain, which is not one of the most polluting countries in the world. So, considering that all the components had been manufactured in China, the amount of CO2 will be higher because their primary electricity source is carbon plants. Note that it has been said the minimum since the internet connection consumption has not been considered. Internet connection has been used to develop such a project. As well as the use of e-mail and teleconferences.

Also, it is essential to know that there are other social impacts related to the extraction of the minerals, such as wars in central Africa and pollution of rivers in some cities where the devices are manufactured.

To conclude, if the primary materials to manufacture the hardware devices of the emulator are from unreliable sources (sources where they can not confirm that they had been involved in wars, polluted cities, etc.) There is also an ethical implication. Regarding the environmental impact, it must be said that at least the minimum impact this project has had is 8062.5 Kg. Note that in Table C.2 a summary of the minimum total consumption of this project considering the simulator (that uses one computer) and the emulator (CO2 generated presented in Table C.1) is shown.

Project	**CO2 generated in Kg**
Emulator	5375
Simulator	2687.53
Minimum amount of CO2 generated: 8062.5 Kg	

Table C.2: Minimum Amount of CO2 generated for the project.

D Tables, Diagrams and Plots

Such an appendix is where tables, diagrams and plots that are too detailed for the document are placed. However, they had been included as far as are a result of the work done to obtain the final results and conclusions of this thesis.

D.1 Tables

Table D.1: Simulation Requirements and Specifications.

Simulation Requirements & Specifications			
Label	**Name**	**Requirement**	**Specification**
*SIM*_010	Customization	The user must be able to set the desired parameters on the simulation	To do so, the implementation shall contemplate the possibility to let the user customize the simulation
*SIMMODEL*_010	Atmospheric effects model	A class that represents the atmospheric effects shall be implemented	To design a model that simulates the atmospheric effects, the recommendations from the ITU-R will be used. To be more specific [11], [12] and [13] would be studied
*SIMMODEL*_020	NS-3 Compatibility	The design of the clouds must take into account the fact that the simulator currently uses the NS-3 class Friis Propagation Losses	To implement such class a way to aggregate the attenuation caused by such effects, must be designed
*SIMMODEL*_030	Probability model	To simulate atmospheric effects it must be taken into account that those effects are not equally among time nor the atmosphere	This means that a probability model must be found or done in order to distribute the atmospheric effects among the globe in a realistic way

Continued on next page

Label	Name	Requirement	Specification
$SIMTRANS\_010$	Transceiver model	The implementation of class that represents a model of a transceiver shall be implemented	Such class must include parameters available in a transceiver such as the noise generated by the hardware, the antenna gain, the processing time of a packet, BW etc.
$SIMTRANS\_020$	Noise floor	A noise floor shall be defined	This noise floor shall be defined in order to redefine the accuracy of the decision of the availability of a contact
$SIMTRANS\_030$	Collisions	A collisions model shall be defined and implemented	The implementation of a model will be based on the processing time of a packet defined by the transceiver
$SIMTRANS\_040$	Binary decider	The current decision of establishing a contact shall be revised	This decision will be modified in order to allow the possibility of the transceiver parameters to affect it
$SIMTRANS\_050$	Radiation pattern	The radiation pattern of an antenna shall be modelled	To model a radiation pattern without consuming to many resources of the simulator, only the direction of the main lobe, the beamwith at -3 dB and the beamwith among 0 will be used
$SIMTRANS\_060$	Gain	The gain of the antenna shall be defined	To improve the computation of the received power the gain of the antennas must be defined
$SIMTRANS\_070$	Polarization losses	The polarization losses must be defined.	The definitions of the polarization losses will be done by using Table D.3
$SIMTRANS\_080$	Direction	The direction where the antenna is pointing to shall be defined	To define the direction such parameter must be taken from the ADCS system

Continued on next page

Label	Name	Requirement	Specification
*SIMTRANS*_090	Consumption States	It shall be defined a model to model the consumption of a transceiver and its states	This model will be designed taking into account an average consumption value for the transceiver hardware or set by the user. To define the consumption states it is as easy as to use the event of transmission
*SIMTRANS*_100	NS-3 Adaptability	The antenna models shall be picked from the ones provided by the NS-3 community	To pick those models, the transceiver must be compatible with the NS-3 antenna models
*SIMSPECTRUM*_010	Spectrum-aware channel	The simulator shall include an spectrum-aware channel	To do so, the class Multi Model Spectrum Channel from NS-3 shall be used
*SIMSPECTRUM*_020	Physical layer	A class shall implement a physical layer link	For this case the class SpectrumPhy from NS-3 (Which is very similar to the NS-3 class NetDevice) will be used
*SIMSPECTRUM*_030	Physical link	A class must model the physical link specifications	The Multi Model Spectrum Channel from NS-3 will be used to perform the physical link

Table D.2: Emulation Requirements and Specifications.

Emulation Requirements & Specifications			
Label	**Name**	**Requirement**	**Specification**
*EM*_010	SDR Capabilities	Allow a wide range of frequencies, good BW and good Samplign rate	At least the SDR should have 2Tx and 2Rx ports. The other parameters should be as good as possible with a good relation on price vs board specifications
*EM*_020	Effects	It shall allow the emulation of propagation effects	Friis propagation loss model, Doppler effect and delay are the minimum effects to be emulated for this communications channel

Continued on next page

Label	Name	Requirement	Specification
*EM*_030	Communications Channel	It shall emulate a communications channel.	The effects described in *EM*_020 must represent the behaviour of the communications channel
*EM*_040	Settings	It shall allow the user to set the effects to be emulated	The user shall be able to choose among which effects on the communications channel would like to emulate
*EM*_050	Data recording	The data generated by the SDRs when processing the signal shall be stored	For some cases of interest it has to be possible to store the data processed
*EM*_060	Channel parameters	The channel parameters shall be obtained by each SDR from the central computer	Those parameters shall be obtained from a central computer and periodically updated
*EM*_070	Plotting	The data obtained from *EM*_050 must be available for plotting	The data recorded shall be able to represent the BER evolution and the SNR for those cases of interest
*EM*_080	Compatibility	The emulator shall be compatible with the DSS-SIM	The compatibility with the DSS-SIM is not only at the time to make some implementation easier, also for in a future work to merge both projects in order to obtain a better satellite contact emulator
*EM*_090	Input power range	The input power range should at least be 33[dBm]	The input power shall be able to be one according to the worst case scenario which would be a GS transmitting at a power of 33[dBm]
*EM*_100	SDRs communication	The communication among SDRs shall be implemented using a real-time protocol	The real-time protocol such as UDP must emulate a live communication among SDRs. So, in some cases some packets can be lost
*EM*_110	Escalating	The emulator shall be escalated to N connections	To do so, a switch that does not interfere with the propagation (low delay and more than 3 connections) shall be used. SO an unmanaged switch will be needed plus with at least 8 ports to test the availability.

Table D.3: Polarization Losses on a Tx-Rx Link.

Transmission Pol. vs Rx Pol. (in dB)	**Horizontal**	**Vertical**	**Right hand***	**Left hand***
Horizontal	0	30	3	3
Vertical	30	0	3	3
Right hand*	3	3	0	30
Left hand*	3	3	30	0

* *Notice that both right and left handed polarization are by definition circular.*

Table D.4: Available hardware for testing satellite communications in the market.

Product	**Company**	**Price [$]**	**Functionalities**
ACE9600	dBm	-	Earth Terminal Test, Mobile Transceiver Test, Satellite Payload Test, Satellite System Integration Test Beds, UAV Test
SLE9250	dBm	-	Earth Terminal Test, Mobile Transceiver Test, Satellite Payload Test, Satellite System Integration Test Beds, UAV Test
SLE9072	dBm	-	Earth Terminal Test, Mobile Transceiver Test, Satellite Payload Test, Satellite System Integration Test Beds, UAV Test
SLE9125	dBm	-	Earth Terminal Test, Mobile Transceiver Test, Satellite Payload Test, Satellite System Integration Test Beds, UAV Test
F8800A *	Keysight Technologies	-	Sat-Sat, Sat-Earth Communications testing
F8820A *	Keysight Technologies	-	Sat-Sat, Sat-Earth Communications testing
PSLE-140-36D	Polaris wave	-	Satellite Modem Test, VSAT Test, Satellite Payload Test, UAV Test, Earth Terminal Test, Satellite System Integration Test Beds, Mobile Transceiver Test
PSLE-L-200D	Polaris wave	-	Satellite Modem Test, VSAT Test, Satellite Payload Test, UAV Test, Earth Terminal Test, Satellite System Integration Test Beds, Mobile Transceiver Test

Continued on next page

Product	Company	Price [$]	Functionalities
PSLE-L-10D	Polaris wave	-	Satellite Modem Test, VSAT Test, Satellite Payload Test, UAV Test, Earth Terminal Test, Satellite System Integration Test Beds, Mobile Transceiver Test
PSLE-L-100D	Polaris wave	-	Satellite Modem Test, VSAT Test, Satellite Payload Test, UAV Test, Earth Terminal Test, Satellite System Integration Test Beds, Mobile Transceiver Test
PSLE-70-10	Polaris wave	-	Satellite Modem Test, VSAT Test, Satellite Payload Test, UAV Test, Earth Terminal Test, Satellite System Integration Test Beds, Mobile Transceiver Test
PSLE-70-10ND	Polaris wave	-	Satellite Modem Test, VSAT Test, Satellite Payload Test, UAV Test, Earth Terminal Test, Satellite System Integration Test Beds, Mobile Transceiver Test
IZT C6000	IZT	-	Sat-Earth Communications testing
IZT C3040	IZT	-	Earth Terminal Test, Satellite Payload Test, Satellite System Integration Test Beds, Mobile Transceiver Test, UAV Test
QSS-131000 C3040	Atlantic Microwaves	60-111.461	Earth Terminal Test
PCE 600	ECA Group	-	Satellite Payload test, Satellite System Integration Test Beds
ECP-70	Elta	-	Mobile Transceiver Test, Satellite Payload Test
T400SG-SSE	KRATOS General Microwave	-	Earth-Sat communications testing

* *Depends on the S8825A Satellite and Aerospace Channel Emulation Toolset provided also by Keysight Technologies.*

Water constant	Value	Ice constant	Value
a_1	1.95	b_1	1.006
a_2	-6.866	b_2	-8.261
a_3	$4.5{\cdot}10^{-3}$	b_3	-1.767$\cdot10^{-3}$
–	–	b_4	-4.374$\cdot10^{-4}$

Table D.5: Benoit's constants.

Clouds Type	LWC [g/m³]	Thickness [Km]
Cirrus	0.03	3
Fog	0.05	0.7
Stratus	0.25 - 0.30	4.5
Cumulus	0.25 - 0.30	3
Stratocumulus	0.45	2
Cumulonimbus	1.0 - 3.0	10.5

Table D.6: Relationship between the type of clouds, their thickness and their WLC

Model	Brand	Price [$]
TL-SG108E	TP-Link	29.99
GS208E	NETGEAR	44.99
GS308T	NETGEAR	69.99
GS108PEv3	NETGEAR	99.99
T2500G-10TS	TP-Link	104.94
SF302-08MPP-K9-NA	CISCO	169.00
GS110-8-HW-US	CISCO	206.82

Table D.7: Comparison table among different switches.

Model	**Brand**	**FPGA**	**Transceiver**	**Connectors**	**MSPS**	**BW (MHz)**	**Frequency range (MHz)**	**Price [€]**
PlutoSDR	Analog Devices	Xilinx Zynq 7010	AD9363	2Tx-2Rx	61.44	20	325-3800	183.52
ADRV9361-Z	Analog Devices	Xilinx Zynq 7035	AD9361	2Tx-2Rx	800	56	70-6000	1095
USRP-2900	NI			1Tx-1Rx	61.44	56	70-6000	1269
USRP-2901	NI			2Tx-2Rx	61.44	56	70-6000	1855
USRP-B210	Ettus (NI)			2Tx-2Rx	61.44	56	70-6000	1460
USRP-2974	Ettus (NI)	Xilinx Kintex-7 XC7K410T	AD9146	2Tx-2Rx	200	160	13-6000	15038
USRP-2954	Ettus (NI)			2Tx-2Rx	200	160	13-6000	11227
bladeRF 2.0 micro xA4	Nuand	49KLE Cyclone-V		2Tx-2Rx	61.44	56	47-6000	1269
bladeRF 2.0 micro xA9	Nuand	301KLE Cyclone-V	-	2Tx-2Rx	61.44	56	47-6000	1855

Table D.8: Comparison table among different SDRs.

Table D.9: Simulation Requirements and Specifications conclusions.

Simulation Requirements & Specifications review				
Label	**Name**	**Studied**	**Designed**	**Verified**
SIM_010	Customization	**X**	**X**	**X**
$SIMMODEL\_010$	Atmospheric effects model	**X**	**X**	**X**
$SIMMODEL\_020$	NS-3 Compatibility	**X**	**X**	**X**
$SIMMODEL\_030$	Probability model	**X**		
$SIMTRANS\_010$	Transceiver model	**X**	**X**	
$SIMTRANS\_020$	Noise floor	**X**	**X**	
$SIMTRANS\_030$	Collisions	**X**	**X**	
$SIMTRANS\_040$	Binary decider	**X**	**X**	
$SIMTRANS\_050$	Radiation pattern	**X**	**X**	
$SIMTRANS\_060$	Gain	**X**	**X**	
$SIMTRANS\_070$	Polarization losses	**X**	**X**	
$SIMTRANS\_080$	Direction	**X**		
$SIMTRANS\_090$	Consumption States	**X**		
$SIMTRANS\_100$	NS-3 Adaptability	**X**		
$SIMSPECTRUM\_010$	Spectrum-aware channel	**X**	**X**	
$SIMSPECTRUM\_020$	Physical layer	**X**	**X**	
$SIMSPECTRUM\_030$	Physical link	**X**	**X**	

Table D.10: Emulation Requirements and Specifications conclusions.

Emulation Requirements & Specifications review				
Label	**Name**	**Studied**	**Designed**	**Verified**
EM_010	SDR Capabilities	**X**	**X**	**X**
EM_020	Effects	**X**	**X**	**X**
EM_030	Communications Channel	**X**	**X**	**X**
EM_040	Settings	**X**	**X**	**X**
EM_050	Data recording	**X**	**X**	**X**

Continued on next page

Label	Name	Studied	Designed	Verified
EM_060	Channel parameters	**X**	**X**	**X**
EM_070	Plotting			
EM_080	Compatibility	**X**	**X**	
EM_090	Input power range	**X**	**X**	**X**
EM_100	SDRs communication	**X**	**X**	**X**
EM_110	Escalating	**X**	**X**	**X**

D.2 Diagrams and Plots

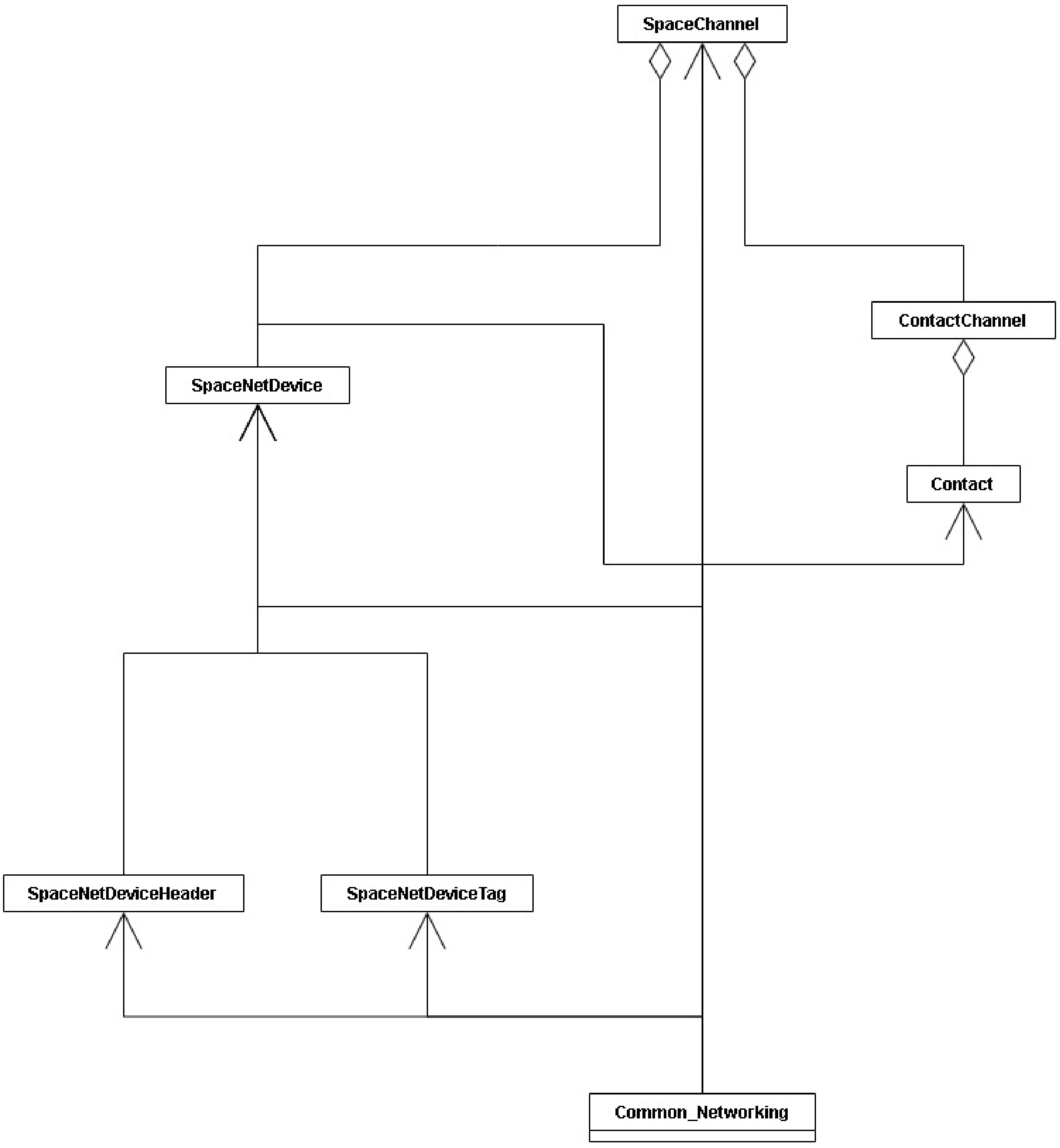


Figure D.1: SpaceChannel UML.

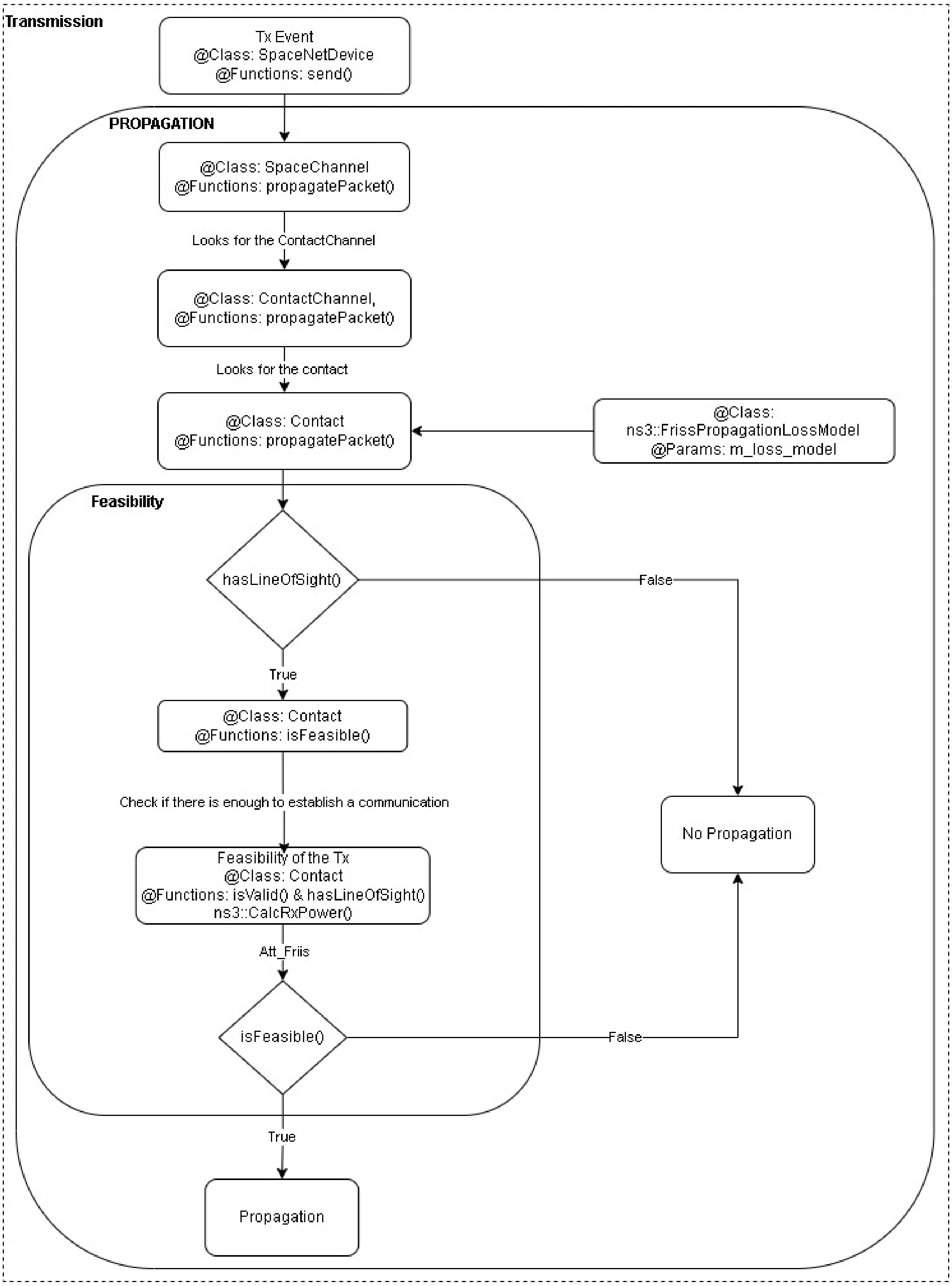


Figure D.2: Flowchart of the transmission process of a packet in the DSS-SIM

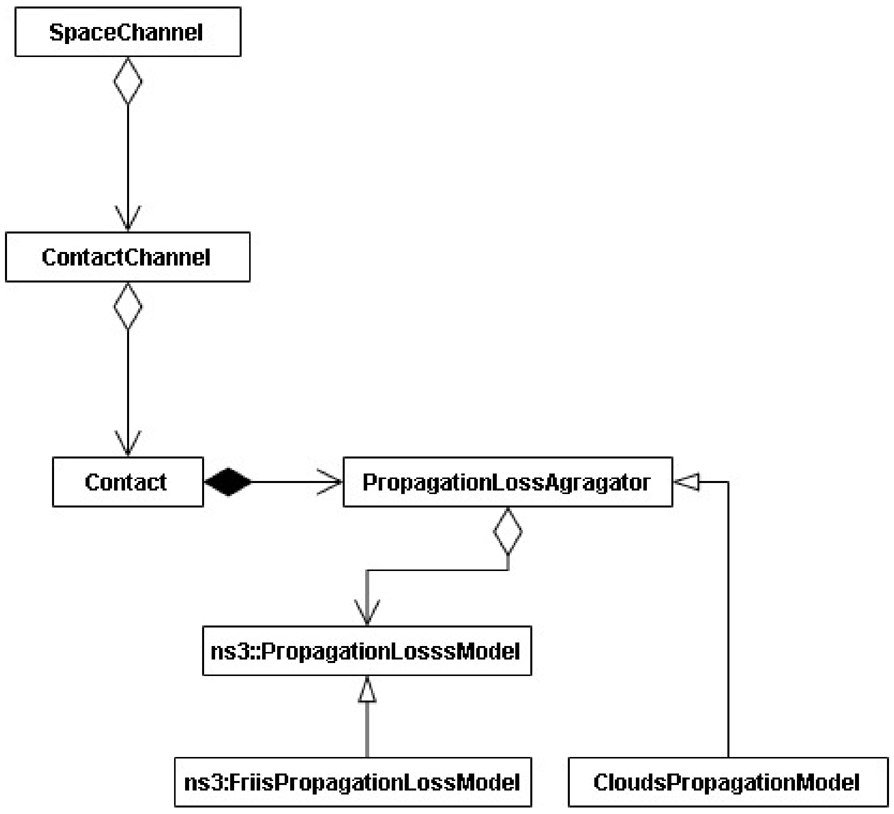


Figure D.3: UML diagram proposal for an attenuation aggregator.

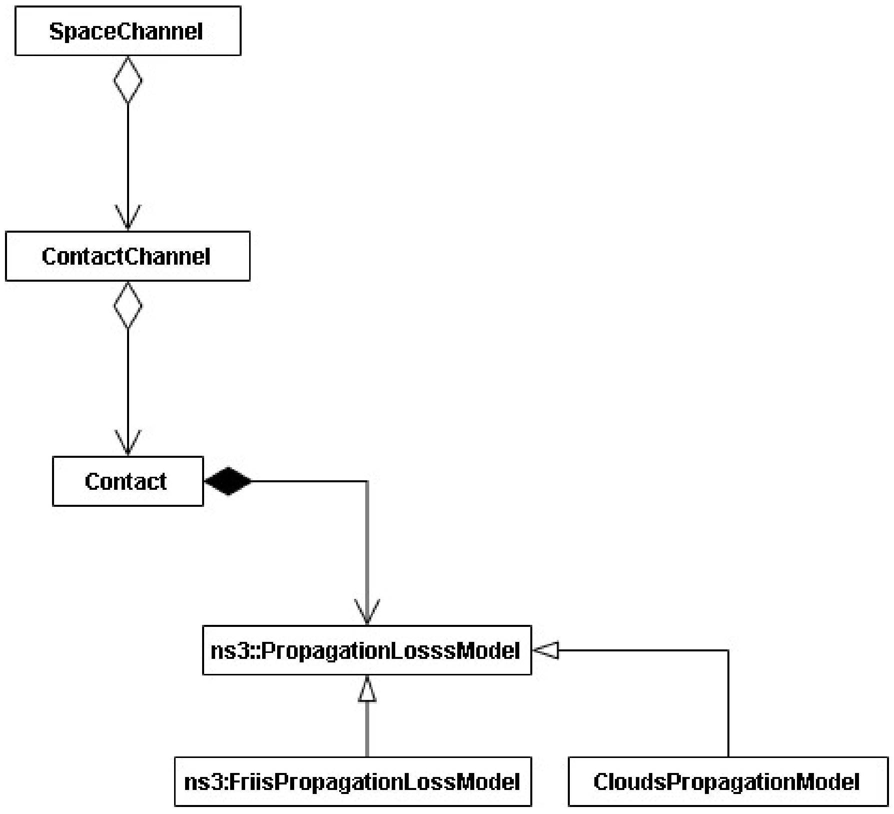


Figure D.4: UML diagram proposal for using ns3::PropagationLossModel and the proposed clouds model.

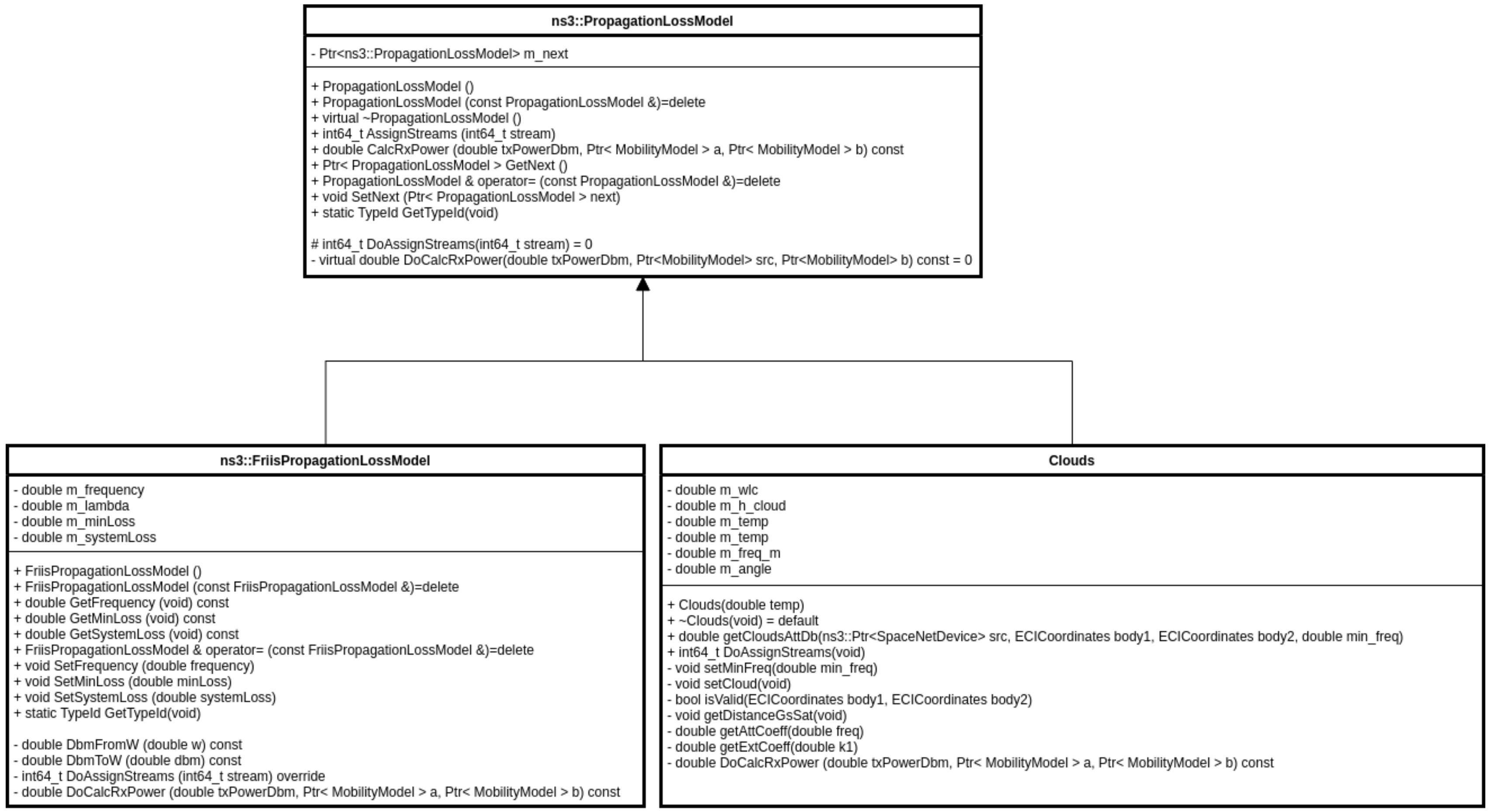

ns3::PropagationLossModel
- Ptr<ns3::PropagationLossModel> m_next
+ PropagationLossModel ()
+ PropagationLossModel (const PropagationLossModel &)=delete
+ virtual ~PropagationLossModel ()
+ int64_t AssignStreams (int64_t stream)
+ double CalcRxPower (double txPowerDbm, Ptr< MobilityModel > a, Ptr< MobilityModel > b) const
+ Ptr< PropagationLossModel > GetNext ()
+ PropagationLossModel & operator= (const PropagationLossModel &)=delete
+ void SetNext (Ptr< PropagationLossModel > next)
+ static TypeId GetTypeId(void)
# int64_t DoAssignStreams(int64_t stream) = 0
- virtual double DoCalcRxPower(double txPowerDbm, Ptr<MobilityModel> src, Ptr<MobilityModel> b) const = 0
ns3::FriisPropagationLossModel
- double m_frequency
- double m_lambda
- double m_minLoss
- double m_systemLoss
+ FriisPropagationLossModel ()
+ FriisPropagationLossModel (const FriisPropagationLossModel &)=delete
+ double GetFrequency (void) const
+ double GetMinLoss (void) const
+ double GetSystemLoss (void) const
+ FriisPropagationLossModel & operator= (const FriisPropagationLossModel &)=delete
+ void SetFrequency (double frequency)
+ void SetMinLoss (double minLoss)
+ void SetSystemLoss (double systemLoss)
+ static TypeId GetTypeId(void)
- double DbmFromW (double w) const
- double DbmToW (double dbm) const
- int64_t DoAssignStreams (int64_t stream) override
- double DoCalcRxPower (double txPowerDbm, Ptr< MobilityModel > a, Ptr< MobilityModel > b) const
Clouds
- double m_wlc
- double m_h_cloud
- double m_temp
- double m_temp
- double m_freq_m
- double m_angle
+ Clouds(double temp)
+ ~Clouds(void) = default
+ double getCloudsAttDb(ns3::Ptr<SpaceNetDevice> src, ECICoordinates body1, ECICoordinates body2, double min_freq)
+ int64_t DoAssignStreams(void)
- void setMinFreq(double min_freq)
- void setCloud(void)
- bool isValid(ECICoordinates body1, ECICoordinates body2)
- void getDistanceGsSat(void)
- double getAttCoeff(double freq)
- double getExtCoeff(double k1)
- double DoCalcRxPower (double txPowerDbm, Ptr< MobilityModel > a, Ptr< MobilityModel > b) const

Figure D.5: UML diagram proposal for using ns3::PropagationLossModel and the proposed clouds model.

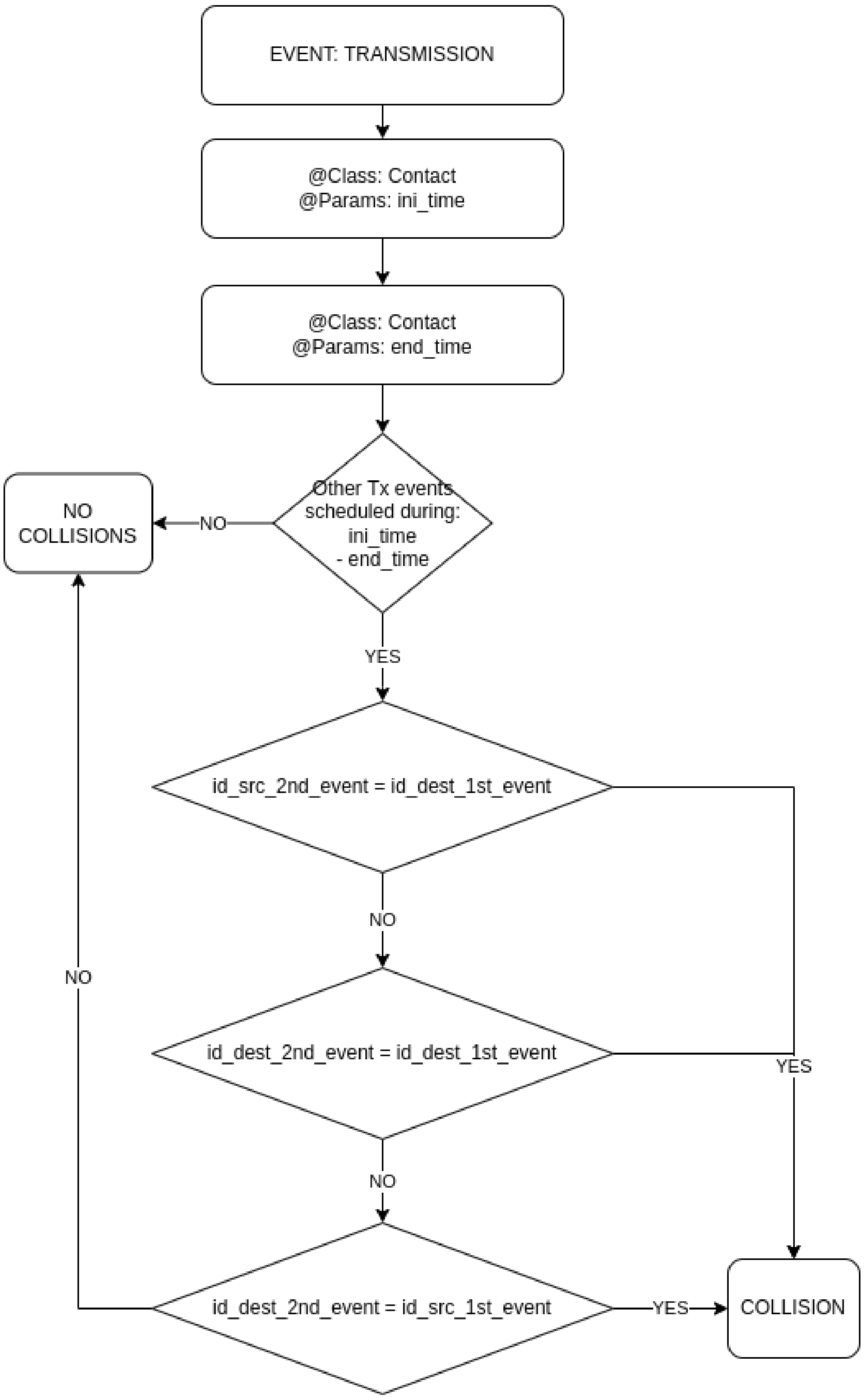
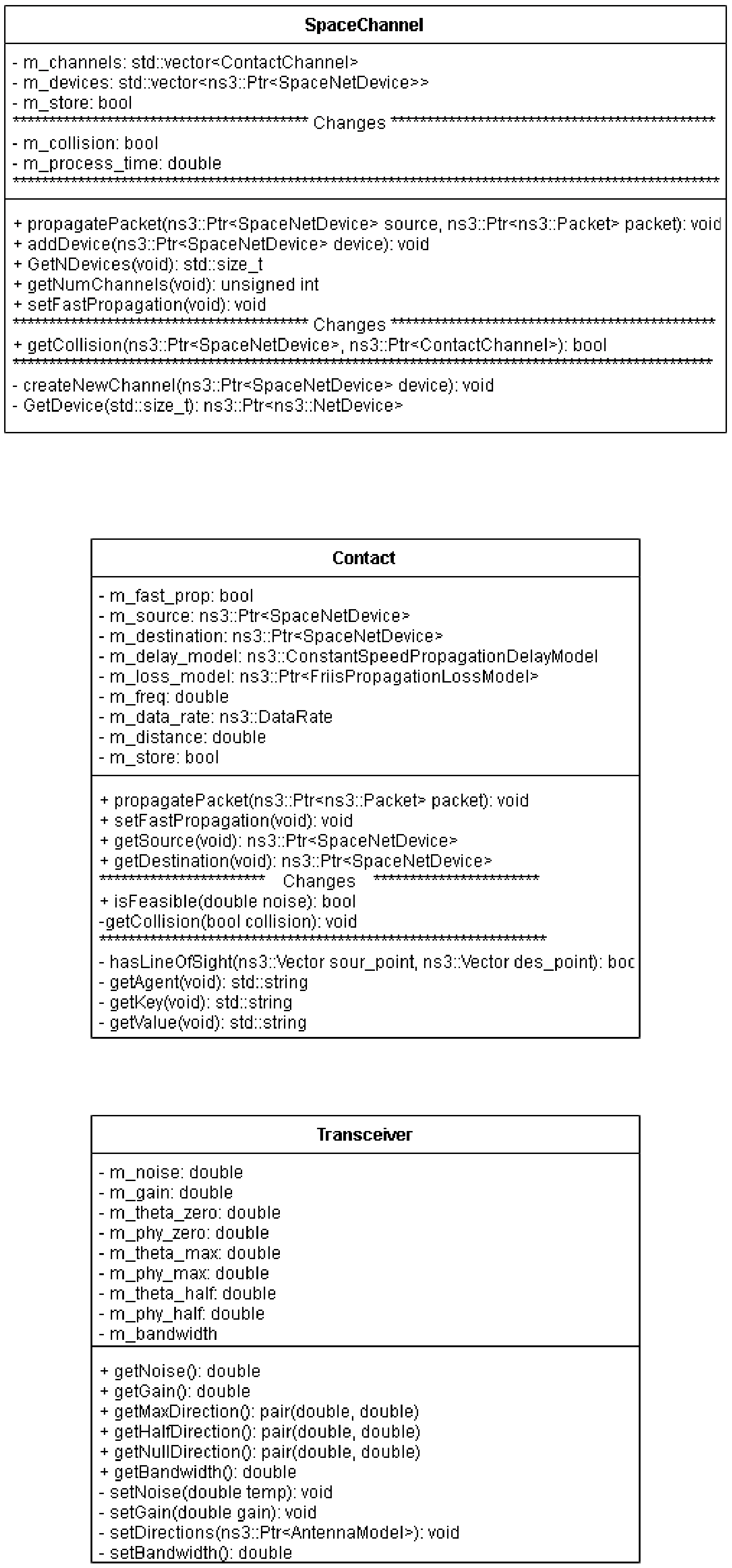

EVENT: TRANSMISSION
@Class: Contact
@Params: ini_time
@Class: Contact
@Params: end_time
Other Tx events scheduled during: ini_time - end_time
NO
NO COLLISIONS
YES
id_src_2nd_event = id_dest_1st_event
NO
id_dest_2nd_event = id_dest_1st_event
YES
NO
NO
id_dest_2nd_event = id_src_1st_event
YES
COLLISION

Figure D.6: Workflow diagram of the collisions case.



```
SpaceChannel
- m_channels: std::vector<ContactChannel>
- m_devices: std::vector<ns3::Ptr<SpaceNetDevice>>
- m_store: bool
******************************************** Changes ************************************************
- m_collision: bool
- m_process_time: double
***************************************************************************************************************

+ propagatePacket(ns3::Ptr<SpaceNetDevice> source, ns3::Ptr<ns3::Packet> packet): void
+ addDevice(ns3::Ptr<SpaceNetDevice> device): void
+ GetNDevices(void): std::size_t
+ getNumChannels(void): unsigned int
+ setFastPropagation(void): void
******************************************** Changes ************************************************
+ getCollision(ns3::Ptr<SpaceNetDevice>, ns3::Ptr<ContactChannel>): bool
***************************************************************************************************************
- createNewChannel(ns3::Ptr<SpaceNetDevice> device): void
- GetDevice(std::size_t): ns3::Ptr<ns3::NetDevice>
```

```
Contact
- m_fast_prop: bool
- m_source: ns3::Ptr<SpaceNetDevice>
- m_destination: ns3::Ptr<SpaceNetDevice>
- m_delay_model: ns3::ConstantSpeedPropagationDelayModel
- m_loss_model: ns3::Ptr<FriisPropagationLossModel>
- m_freq: double
- m_data_rate: ns3::DataRate
- m_distance: double
- m_store: bool

+ propagatePacket(ns3::Ptr<ns3::Packet> packet): void
+ setFastPropagation(void): void
+ getSource(void): ns3::Ptr<SpaceNetDevice>
+ getDestination(void): ns3::Ptr<SpaceNetDevice>
*************************    Changes    *************************
+ isFeasible(double noise): bool
-getCollision(bool collision): void
*********************************************************************
- hasLineOfSight(ns3::Vector sour_point, ns3::Vector des_point): boc
- getAgent(void): std::string
- getKey(void): std::string
- getValue(void): std::string
```

```
Transceiver
- m_noise: double
- m_gain: double
- m_theta_zero: double
- m_phy_zero: double
- m_theta_max: double
- m_phy_max: double
- m_theta_half: double
- m_phy_half: double
- m_bandwidth

+ getNoise(): double
+ getGain(): double
+ getMaxDirection(): pair(double, double)
+ getHalfDirection(): pair(double, double)
+ getNullDirection(): pair(double, double)
+ getBandwidth(): double
- setNoise(double temp): void
- setGain(double gain): void
- setDirections(ns3::Ptr<AntennaModel>): void
- setBandwidth(): double
```

Figure D.7: Transceiver attributes and changes in the classes.

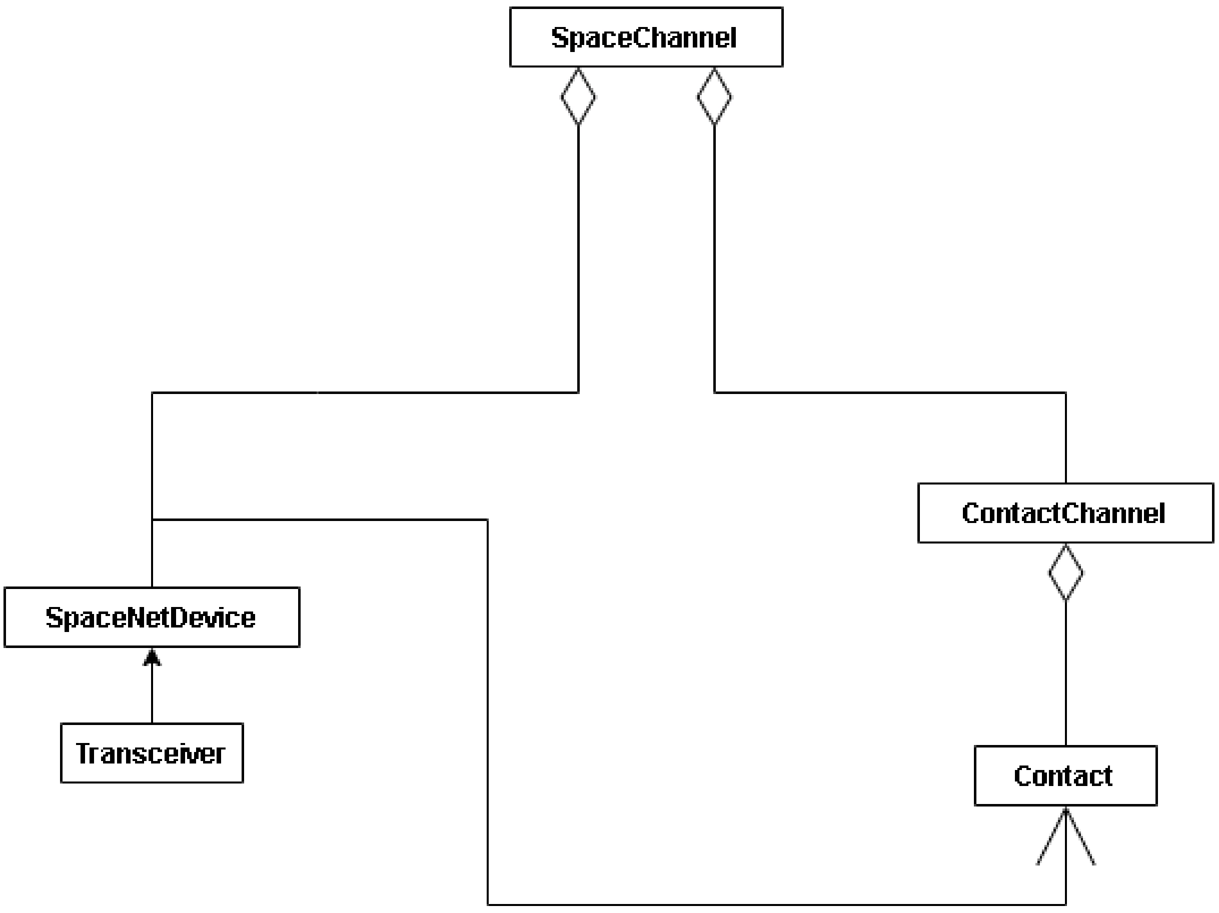


Figure D.8: Simplified UML diagram integrating the transceiver.

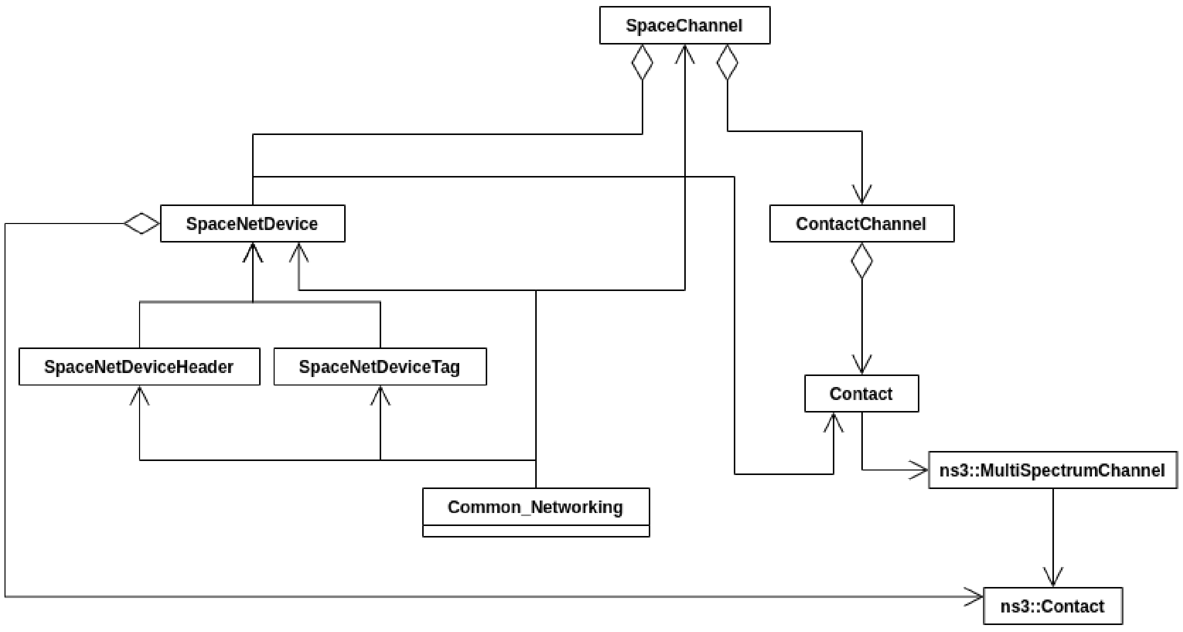


Figure D.9: MultiModelSpectrumChannel integration UML.

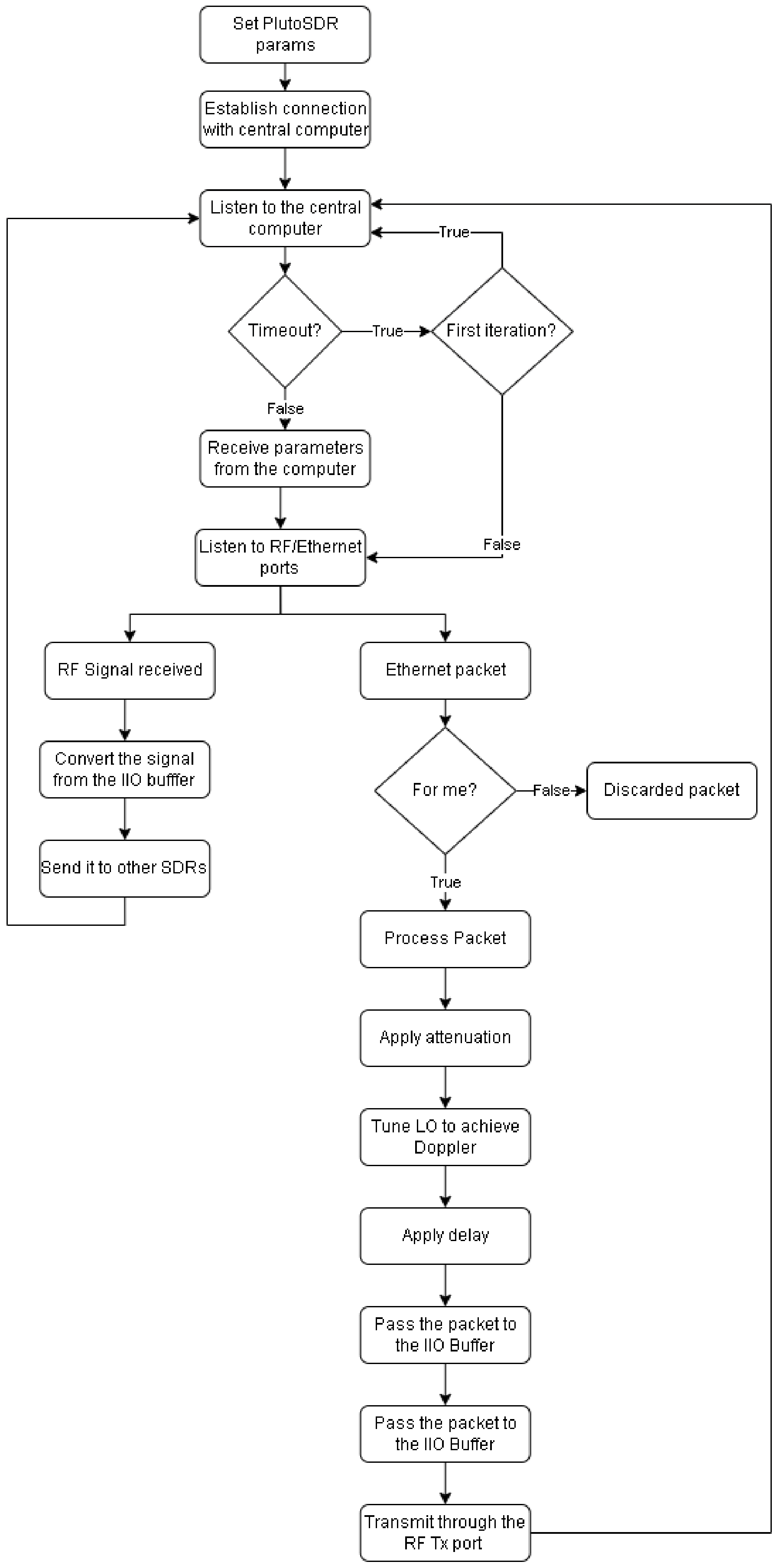

Set PlutoSDR params
Establish connection with central computer
Listen to the central computer
Timeout?
True
First iteration?
True
False
Receive parameters from the computer
Listen to RF/Ethernet ports
False
RF Signal received
Ethernet packet
Convert the signal from the IIO bufffer
For me?
False
Discarded packet
Send it to other SDRs
True
Process Packet
Apply attenuation
Tune LO to achieve Doppler
Apply delay
Pass the packet to the IIO Buffer
Pass the packet to the IIO Buffer
Transmit through the RF Tx port

Figure D.10: Proposed workflow structure for the testbed.