Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: A multi-agent adaptive protocol for Femto-satellite applications

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, esp. en Telemàtica

AUTORS: Raquel González, Joan Naudó

DIRECTOR: Joshua Tristancho Martínez

DATA: 15 d'Juliol de 2011

Título: A multi-agent adaptive protocol for Femto-satellite applications
Autores: Raquel González, Joan Naudo
Director: Joshua Tristancho Martínez
Fecha: 15 de Juliol de 2011

Resumen

Los Femto-satelites son una categoría muy prometedora de satélites con menos de 100 gramos de masa.

También, un Pico-rover es un robot autocontenido de menos de un kilogramo de masa que su movimiento se realiza rodando su carcasa exterior, dejando el entorno fuera.

La ventaja principal de este tipo de agentes pequeños son sus múltiples puntos de vista cuando están trabajando como enjambre o dentro de una constelación. La complejidad de este tipo de sensores en red, además de su bajo consumo y pequeño tamaño, requieren una buena estrategia de gestión que es lo que se quiere presentar en este trabajo.

El paradigma de la “gestión del agente” consiste en un solo punto de vista de alta calidad y muchos puntos de vista de baja calidad donde la selección del punto de vista de alta calidad la realiza la propia red pero decidido de forma externa a la red o siguiendo una ley básica. Esta solución optimiza el ancho de banda de la red. En vez de transmitir cada punto de vista a alta calidad, solo uno de ellos lo hace. Al mismo tiempo esta solución permite una distribución de tareas en la que solo hay un agente productor, un agente consumidor y el resto de agentes trabajan como nodos repetidores.

Este trabajo está dirigido, por un lado, al diseño un simple pero robusto protocolo adaptativo basado en este paradigma; por otro lado, la implementación de este protocolo usando una plataforma de bajas características como lo es la arquitectura del microcontrolador 8051.

Palabras clave: Multi-agent, Adaptive protocol, Femto-satellite, Pico-Rover, SCP, UDP, IP, Network sensor

Title: A multi-agent adaptive protocol for Femto-satellite applications
Authors: Raquel González, Joan Naudó
Director: Joshua Tristancho Martínez
Date: July, 15th 2011

Overview

Femto-satellites are a very promising category of satellites that weigh less than 100 grams.

Also, a Pico-Rover it is a self-contained robot that weighs less than 1 kilogram and its motion works by rolling the external enclosure that keeps out any environment threats.

The main advantage of this kind of small agents is the multi-point of view when they work as swarm or taking part of a larger constellation. The complexity of these kinds of network sensors, in addition to the low power requirements and low size, requires a good strategy of management that we want to present in this work.

The paradigm on management-on-agent consists of a single high quality point of view and multiple low quality points of view where the selection of the point of view is done inside the network but decided externally to the network or done by a basic law. This approach optimizes the bandwidth used by the net. Instead of streaming every high quality point of view we only stream one of them. At the same time, this approach allows a task distribution on the network where there is only one producer agent, one consumer agent while the rest of agents work as relay nodes.

This work is addressed, on one side, to the design of a simple but robust and adaptive protocol based on this paradigm; on the other hand, an implementation using a low performance platform like the 8051 microcontroller architecture is required.

Keywords: Multi-agent, Adaptive protocol, Femto-satellite, Pico-Rover, SCP, UDP, IP, Network sensor

A nuestros padres.

INDEX

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

The current work presents a protocol that accomplishes the Management-on-agent paradigm where the information management passes from the application layer to the network layer. That way, it is possible to optimize the use of the net bandwidth for a given sensor network. The aim of this protocol is to allow, on the network layer, to download periodically basic information of every node to the Consumer node (Or sink node), giving the maximum bandwidth to the Producer node. For this reason, a discovery mechanism is defined that optimizes the routing from the Producer to the Consumer node.

Applications for this new protocol are based on sensor networks. In this work, two examples will be presented that have opposite behaviour: A constellation of Femto-satellites that are deployed together in the same orbit but keep separating as time goes by. The role of Producer is conditioned by the constellation position respect to a celestial phenomenon. The other example is a swarm of Pico-Rovers that walk over the Moon's surface and cannot separate further than 50 meters to avoid connectivity loss. The distribution is random and each rover tries to reach the furthest separation allowed.

The Femto-satellite is in the less than 100 grams category, and the Pico-Rover is in the less than 1 kilogram category.

Acknowledgements

We are grateful to our parents and our friends for supporting us during this work.

We are very grateful to all the WikiSat members who helped and supporting us with the realization of the TFC, specially Joshua Tristancho, Victor Kravchenco, Esteve Bardolet, Sonia Perez, Javier Perez, Roberto Rodríguez, Enric Fernandez and Lara Navarro.

Finally we want to thank our sponsors and collaborators showed in Figure 1:

Figure 1 Wikisat partners

ACRONYMS, ABBREVIATIONS AND DEFINITIONS

MOA	Management-on-the-agent
MAS	Multi Agent System
GS	Ground Station
TTC	Telemetry, Tracking and Command
MCU	Main Control Unit
IMU	Internal Measurement Unit
HD	High Definition
SoC	System on a Chip
RAM	Random-access memory
ACK	Acknowledgement
LOS	Line of sight
NLOS	Non Line of sight
OSI	Open System Interconnection
SNR	Signal to noise ratio
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CA	Consumer Agent
PA	Producer Agent
RA	Relay Agent
RTT	Round Trip Time
ETX	Expected Transmission Count
UML	Unified Modelling Language
LEO	Low Earth Orbit
FOV	Field of View
HQPV	High Quality Point of View
NACK	Negative Acknowledgement
QoS	Quality of Service

CHAPTER 1. Management-on-agent paradigm

This chapter, the Management-on-agent paradigm or MOA is introduced. All the architecture we present is based on it.

1.1 Management-on-agent paradigm

For a given agent network, the paradigm of Management-on-agent (MOA) sets that:

The responsibility of streaming management of an agent network passes to the agent and not to the network control in order to optimize the bandwidth of the network.

For example, if the agents are a set of wireless cameras, the paradigm of MOA consists of a single HQPV and multiples HQPV where the selection for the selected point of view is done externally to the network or done by a basic law. This approach optimizes the bandwidth use because instead of streaming every HQPV, only one of them streams thus the rest of points of view in HQPV allow the control task. At the same time, this approach allows a task distribution in the network where there is only one Producer Agent (PA), only one Consumer Agent (CA) while the rest of nodes work as a Relay Agent (RA). All the Agents should have the same properties. The term "point of view" here is used not only in terms of a camera but also in terms of to known information about how, when and where the point of view has been taken.

Hence, to implement this paradigm, the subsequent protocol is proposed. This implementation should know some rules in order to define which agent has the Producer role and which agent has the Consumer role thus the rest of agents work as a Relay role.

As soon the protocol is defined, a number of basic functionalities are selected in order to define the best algorithm for the late implementation. The algorithm is abstracted from the framework but the implementation is done. Depending on the hardware of the communication layer, implementation can vary in complexity. Let us see some examples.

Traditional approach for Recording Studios was to stream every single HQPV to a master control panel where, sooner or later, the selection action is done. The final result is a sequence of these HQPV. With this traditional approach, there is a huge waste of resources only affordable if transmission lines are wired. If the transmission line is a wireless network, this traditional approach is not viable. The MOA paradigm proposes to break with this approach and to make more efficient use of the resources and bandwidth available. For this reason, a new protocol is required.

Other examples of application for the proposed protocol are a Femto-satellite constellation that the configuration is somehow expanding but distribution is always the same. Also, a swarm of Pico-Rovers that can roll over the Moon but

line of sight limits the link to 50 meters; distribution can change. These examples will be explained in deep in other chapters.

1.2 Multi Agent System (MAS)

In order to have a multipoint view feature, it is necessary to implement a Multi-agent system (MAS). This system is composed of multiple interacting intelligent agents, all of them equally in such a way that they can change the role. The MAS system allows solving problems that cannot be solved by an individual agent-based system.

The MAS used in these kinds of networks are supposed to have the following characteristics:

- Autonomy
- Local views
- Decentralization

The prototype agent works within the directives of the following scheme:

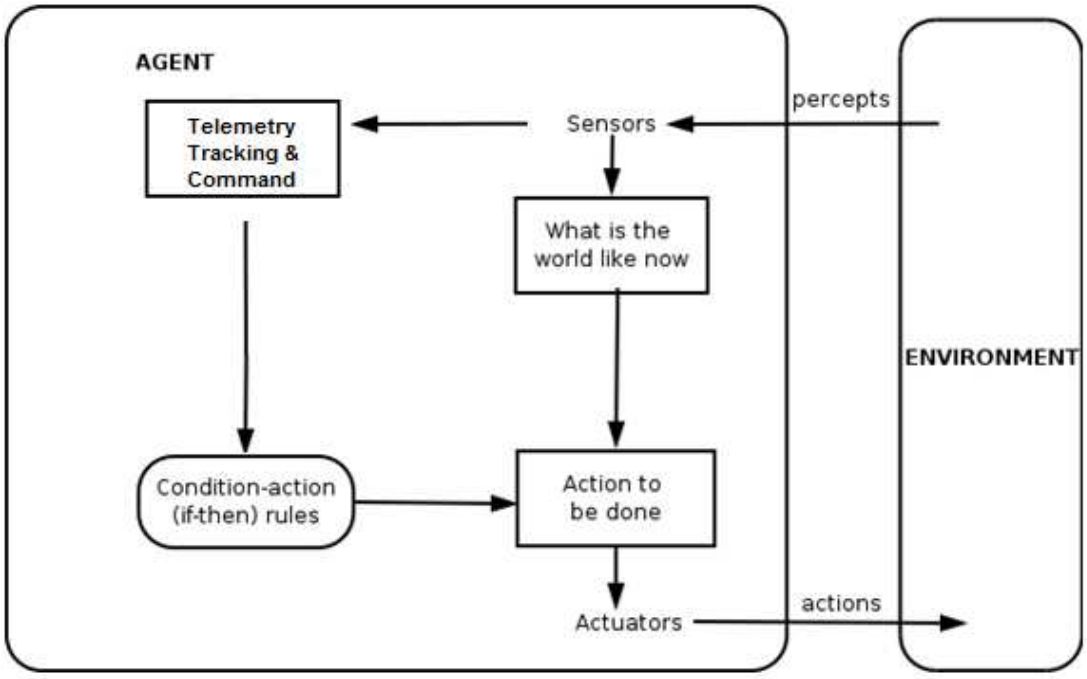


Figure 2 The agent process

Figure 2 summarizes any agent processes, defining the agent as a node with a reduced intelligence. The special feature of this system is the coordination and cooperation of the all nodes.
The following features define each aspect of the agent inner workings:

TTC: Telemetry, Tracking and Command. This function is in charge to send the position to other agent or ground station and also it is able to receive new commands. The MAS needs this data in order to take right decisions.

Sensors: They produce the data streams that will be later sent to another relay node or to the ground station.

Actuators: Physical response management. For example, changes the direction of any antenna or sensor in order to keep the agent in optimal position.

Some examples of agents are: Pico-Rover, Femto-satellite and Camera inside a show.

- **Pico-Rover:** This is an agent that is able to rover over the Moon surface in order to explore the surface and send video back to the Earth.
- **Femto-satellite:** This is an agent that records a video of the Earth surface and sends the information to the ground.
- **Camera show:** This is an agent that has a high quality camera to record a single point of view inside a scenario.

1.3 Case study of MAS

Figure 3 shows three case studies for a Direct link case, A Relayed link case and a Multi path case.

A direct link case is when the CA is closer enough to the PA, not needing any other agent at all; which is not true because the aim of the MAS is to provide real-time information for each of the agent's point of view to the CA. In this case we can see that the transmission is faster and no delay is produced. TTC should be sent from every node to the CA.

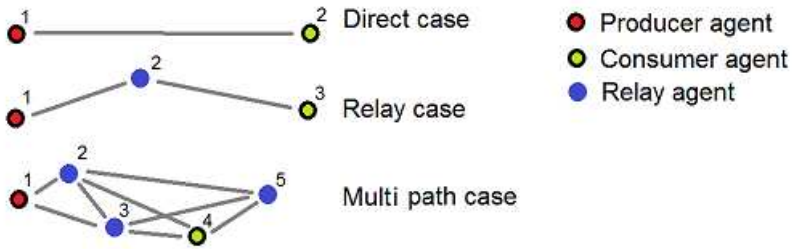


Figure 3 Direct case, Relay case and Multi path case

A relay case is when the CA cannot reach the PA directly but through other agent. In this case we can see that the use a relay node will add some delay to the full communication time.

When multi-path case is considered, delay depends proportionally to the number of jumps needed. Infinite loops must be avoided.

1.3.1 Observations for all cases

Observing the previous cases, some basic ideas can be taken into account:

- TTC information of all nodes should be sent whatever the configuration
- The smaller jump number is the better because distances are small
- When multi-path the lower jump number is the preferred
- Need to avoid infinite loops
- Need to ignore repeated information
- There is a point that the sensor network cannot sent all the information

1.3.2 Conclusions for Direct, Relayed and Multipath cases

There is no sense of the paradigm of MOA for the Direct case but when any RA exists, an optimization in the sensor network can be done. The MOA paradigm is implemented in the previous Direct case in such a way that most of the time, packets from the PA are received by the CA and time to time, packets from other RA is received by the CA. When the boundaries of the sensor network is constant, maximum efficiency is achieved in the use of the bandwidth but for changing sensor networks, the changing boundaries make less efficient the use of the bandwidth because many packets are sent to adapt the sensor network.

CHAPTER 2. State of art Femto-sats and Pico-Rovers

Before starting to work on the design of the new protocol, we will first introduce and analyse which agents will be used and its own purposes. In this chapter, we will present the two main kinds of agent, the Femto-satellites and the Pico-Rovers.

2.1 Introduction to Femto-satellites

Femto-satellites are a new kind of satellites that weigh less than 100 and are not much bigger than a cell phone. In no time, these satellites will probably displace the conventional satellites (more than a ton of weight and much higher manufacturing cost).

These satellites are organized as swarms and could revolutionize many applications in telecommunications, military, entertainment, science, weather and climate forecasting science fields at much lower fabrication and launch costs.

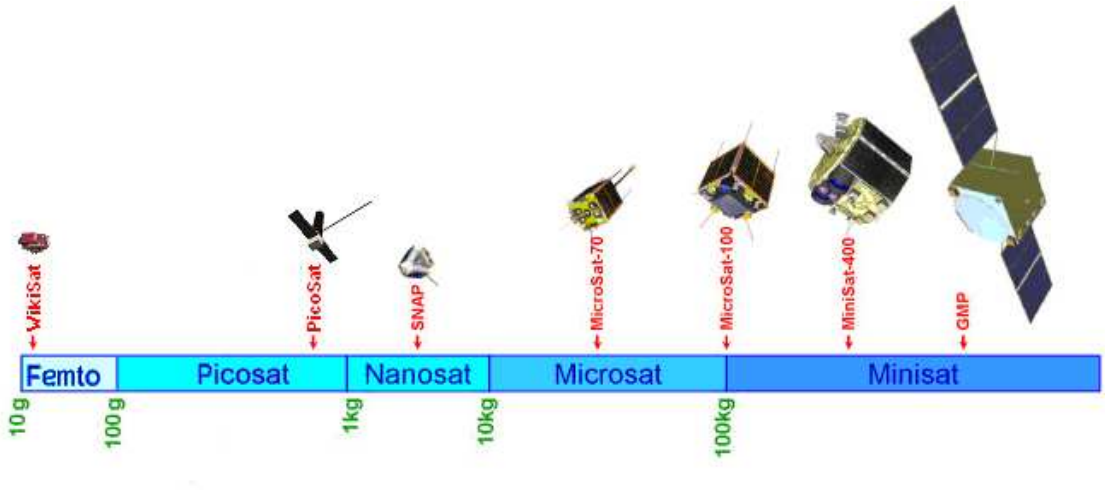


Figure 4 Scale of satellites

In the WikiSat group, we work in the implementation of this kind of satellites.

Some examples of Femto-satellites are:

- Comtech AeroAstro's Miniature Star Tracker: a Femto-satellite created to track stars with a high-angular optical.
- Krishna Kumar and his research team developed the first Femto-satellite constructed in a Canadian university lab [8]
- Helvajian and Janson have designed a complete Femto-satellite in 2002 [7]

2.1.1 The N-Prize contest

The N-Prize challenge is a competition to motivate the creativity, originality and inventiveness in the face of severe odds and impossible financial restrictions.

The rules of the contest are very simple; we have to put a satellite with a mass of between 9.99 and 19.99 grams, and to prove that it has completed at least 9 orbits around the Earth. The total cost of the mission cannot exceed in any case the £999.99.

There are two prizes of £9,999.99 each for the first entrant to complete the challenge using a non-reusable launch system and for the first entrant to complete the challenge using a partially or wholly reusable launch system. Lots of teachers, students and collaborators are working in a Femto-satellite called WikiSat and his rocket, the Wiki-Launcher to participate in the N-prize[1].

2.1.2 *WikiSat: The Femto-satellite*

The WikiSat space program consists of to implement a low cost satellite for the N-Prize.

The satellite WikiSat is a less than 20 grams Femto-satellite that will be the brain of our mission, is responsible of ignite the rocket, its control, to ignite the Stage2, etc... With this we save job and weight because we don't need a single control system for the rocket, our WikiSat is capable of doing these functions also.

Then the WikiSat is an essential part of the group and everything that makes the group has directly or indirectly results with the Femto-satellite from the Satellite Heart (AWIP board [11]) and the antenna [12], until its vehicle into orbit [10].

We have worked to allow that all Femto-satellites can establish network and return information once they are in orbit, working like a constellation. There are four versions some of them are showed in Figure 5.

[1] http://www.n-prize.com/

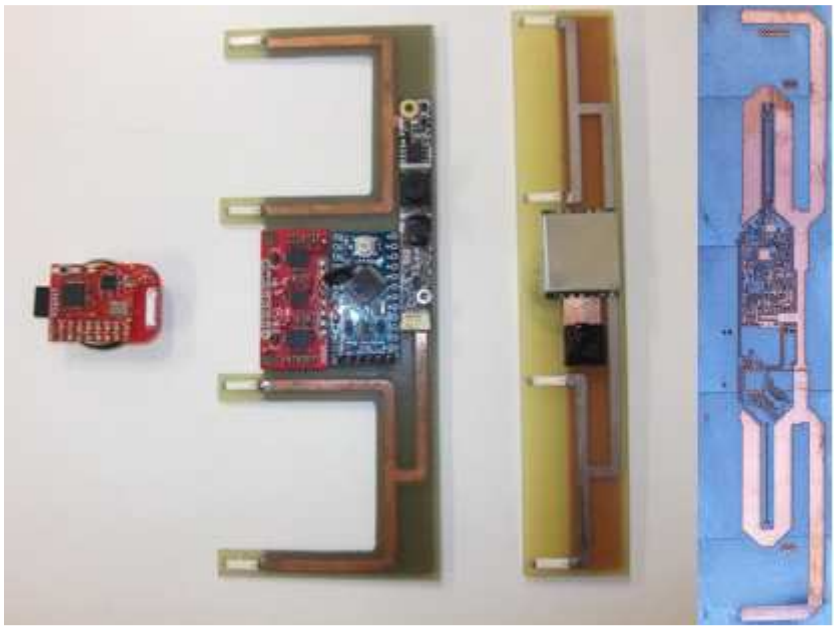


Figure 5 WikSat prototypes (1, 2, 3 & 4)

2.2 Introduction to Pico-Rovers

A rover is a space exploration vehicle designed to move across the surface of a planet or natural satellites. There are several kinds of Rovers. One of them is a Pico-Rover. The Pico-Rover is a pico lunar rover which is formed of a less than 1 kg self-contained autonomous ball. We can find different Pico-Rovers nowadays, but all of them share the same approach. Some examples are the following:

- Rotundus is a project of GroundBot Company. Rotundus use the same approach to achieve the movement as the Pico-Rover. The difference is that this ball has two lateral cameras which allow seeing different points of view at the same time.
- Sphero is a project of Orbotix Company. It's the first Pico-Rover controlled with a smartphone (Iphone or Android software).

2.2.1 The X-Prize contest

The Google Lunar X PRIZE is a competition also created by Google Company. This challenge pretends to ignite a new era of lunar exploration. The main challenge of this competition consists in safely landing a robot on the surface of the Moon, make it travel 500 meters over the lunar surface and send high quality images and data back to Earth. Teams must be at least 90% privately funded, though commercially reasonable sales to government customers are allowed. There is a of $30 million prize for the first team than is able to complete the challenge[2] before 2014.

[2] http://www.xprize.org/

2.2.2 Team FREDNET: The Pico-Rover

Pico-Rover is a project from Team FREDNET[3], a group that participates in the Google X-Prize.

Pico-Rover is sphere robot with less than 250 grams and 12 centimetres diameter. The Pico-Rover is equipped with a motor and a HD camera with a micro-computer that allows it to establish current position, receive information, process data and send it back.

Pico-Rover is based in the roly-poly concept. The counterweight trends to maintain the rounded shape always straight. For the same reason, if this counterweight is translated, it allows the Pico-Rover to move in the same direction and even climb slopes.

Moreover, Pico-Rover contains a shield to protect it from the radiation, and extreme temperatures. This shield is made of fiberglass. This component matches the stated specifications. Also, Pico-Rover is provided with “spikes in order to give it some kind of traction to the ground, so it can avoid any issues while moving on sand or dust like surfaces.

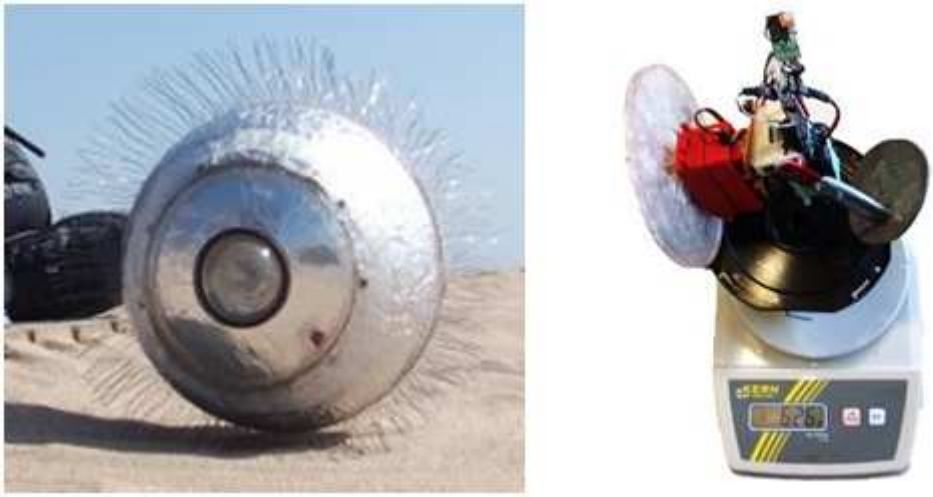


Figure 6 a) Pico-Rover on the beach and b) Less than 1 kilogram component

2.3 Proposed hardware platform

We will use a hardware platform that can be used regardless the agent, be it a Pico-Rover or a Femto-satellite (or even some other future devices).

This platform features the following sections [13]:

[3] http://www.frednet.com/

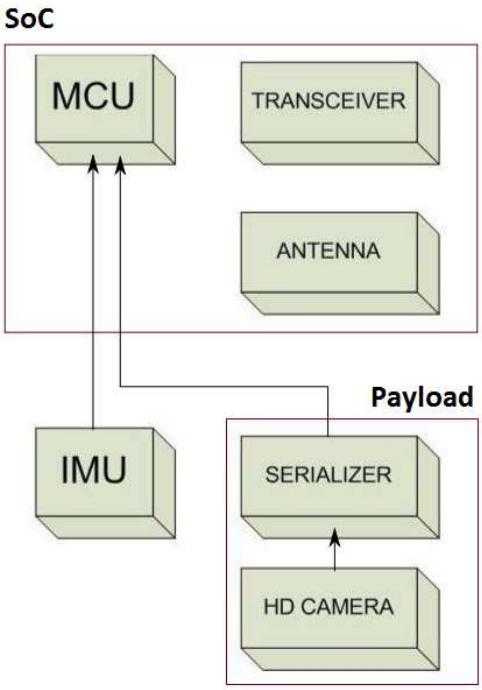


Figure 7 Hardware platform

- System-on-Chip (SoC): It contains all the necessary components of a full performing computer in a single integrated chip. It includes the following components:

    - MCU (Main Control Unit): It features a 8051 processor, 16MHz clock frequency, 32 kb of RAM and 2 kb SRAM.
    - Transceiver: Charged with transmission and data reception. It will use the 2'4 GHz bandwidth at a 250kps rate.
    - Antenna: Constituted of a broadside array of ceramic antennas.

- IMU (Internal Measurement Unit): It captures all the data relevant to the movement of the platform and its internal environment. Three axis mounted accelerometers provide a 16 bits resolution data, along with 3 gyroscopes and temperatures sensors.

- Payload: The devices charged with the responsibility of getting a the payload are :

    - HD Camera: We will use the TCM8230MD, with 1.3 Mpix of resolution it provides high definition.
    - Serializer: It serialise the data provided by the camera. The exact model we will be working with is the SN74HC165.

CHAPTER 3. Proposed background

Once the different kinds of agent have been studied in detail, we will analyse the main scenario where the protocol will be implemented. It is vital to intimately understand the full environment if we are to design and optimal solution.
In this chapter, we will observe the network characteristics and its directives for its right performance.

3.1 Kind of Network

The network we are facing is an ad-hoc one. It means that is a kind of unmanaged wireless network where each node is able to forward any data to the rest.

This kind of network is very flexible and dynamic, in which the nodes vary its position depending on time and network needs, altering completely the network main topology.

The nodes working in these conditions have limited durability due to its reliance on batteries. This also factorises critically the power requirements in transmission, with the challenge of achieving a low-power network.

3.2 Proposed requirements

In order to implement the MOA paradigm it is necessary to establish a set of basic directives that must be followed during the entire communication process. These will be applied in the case studies explained later on chapters 5 and 6. They affect as much as the agents as the communication process itself. Now we will describe them.

3.2.1 Acknowledgement failure

For most part of the time the send/receive data will be basically a video streaming. This implies certain requirements that must be obliged in order to correctly transfer the data.

The data stream implies that there is a constant flow of data. The delays inside the network affect the continuity of the transmission, and must be as short as possible. This is why we will not confirm correct arrival of data, or receiving acknowledgement (ACK nor NACK), because it would add and unbearable delay to the net.

3.2.2 *Error detection*

Due to we use the radio wave channel, there will be a lot of noise during the data transmission, so we have to determine whether the data that has arrived contains any error.
A set of mechanism will be designed in order to accomplish this task.

3.2.3 Routing

As mentioned before, in an ad-hoc network, all nodes must be able to redirect data packets towards its final destination. To this end we will apply a dynamic routing protocol capable to trace the most optimum path in each case. This protocol will be explained in detail in the next chapter.

3.2.4 Agent roles

There are three major roles for the agents to follow: Consumer, Producer and Relay. In order to assign a role to each node, the following set of rules will be established:

- *Consumer node:*
  Only one node can be the CA, and this node always will be the node with a direct link of the ground station and it presents better characteristics. Then, this node will be responsible for transmitting the information to the ground station.
  If, at any time, the node does not satisfy these conditions, is no longer valid to fulfil this role and is no longer the CA. Then, we must supervise the MAS in order to select a new CA.

  If no agent is in direct link with the ground station, there is not a CA. This is important in situations where the agents spend more time in NLOS. Then, we need a solution for these cases. The proposed solution is to distribute the information between all the agents of MAS while there is no CA.

- Producer Node:
  As with the CA, only one node can be a PA in the MAS. Then, this node will be responsible for passing the information to other agents until it reaches the CA. This means that the PA always sends the information to the CA address.

  Any node can be selected as PA, and the decision comes entirely from the ground station.

- Relay Nodes:
  Once the nodes for the CA and the PA have been selected, the rest of the nodes will form the MAS, and will automatically assume the Relay Role. The RA sole mission is to route all the data stream to its final destination.

CHAPTER 4. Multi-agent adaptive protocol

Once we have analysed the scenario and the use of the agents, we can proceed to the design stage of the new protocol.

In this chapter the proposed protocol for the MOA paradigm and the basic functionalities are explained.

A protocol is a series of rules that computers use to manage their dialogue in information exchanges. These rules allow two separated hardware from different bands to communicate without problems.

On the other hand, functionality can be defined as a series of tools that can be used to implement the protocol.

4.1 OSI layered structure

In order to be able to design a protocol that covers our needs in the given situation, we will base our structure on the OSI standard and work from it.

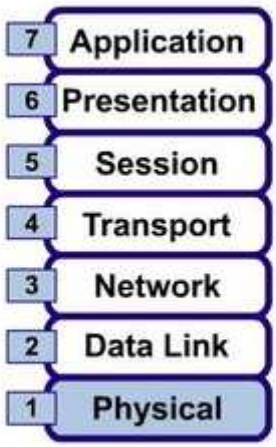


Figure 8 OSI layered structure

This model is divided in 7 different layers, where each layer performs a diferent function of the communication process. Each layer will have to:

- Communicate with the immediately adjacent layers, being able to transmit and receive to both ways.
- Facilitate the implementation of the model.

In our case, due to the specific nature of the new protocol, we will only use the following layers:

- Physical layer
- Link Layer

- Network layer

4.1.1 Physical layer

This layer defines the physical support of the network connections. Because we are working in a wireless network, this layer will control the antenna parameters, the power output, SNR and power consumption.

This particular area will not be detailed on this Project, since it is the subject of another Project called “A multi-agent payload management approach for femto-satellite applications” [10].

4.1.2 Link Layer

The main purpose of this layer is to provide an efficient communication with no error between two nodes. This layer takes consideration on the following aspects:

- Physical routing
- Network Topology
- Medium Access
- Error detection
- Flux Control
- Arranged Frame Distribution

4.1.2.1 Physical Routing

Each node will have a unique address that will enable communications between any two nodes. The memory size of this address is 4 Bytes.

4.1.2.2 Network topology

It is known as network topology the physical disposition of the different nodes inside the net. Taking note on this, there are several structures, being some of them: tree-shaped, star-shaped, mesh, or simply a combination of them.

The MAS have a mesh topology, meaning each and every node is connected to all the others, achieving a very redundant system that always offers many different paths to reach the same destination. Thus we avoid problems due to link failure.

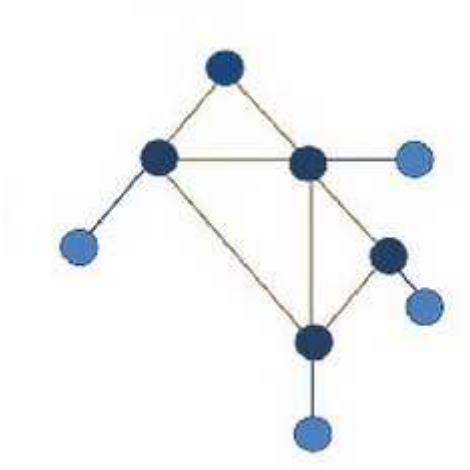


Figure 9 Mesh network

4.1.2.3 Medium Access

The medium access is composed by the all the mechanism that enable the different nodes to access the transmission medium.

There are several ways to control this access by the nodes of the net, and we have chosen the most suitable.

The protocol we have selected is CSMA/CD (Carrier Sense Multiple Access with Collision Detection). In this protocol, any node wanting to transmit will have to previously listen to the channel. There are mild variations from the original, and we have selected the non-persistent mode.

On this mode, if the cannel is occupied when we listen to it, we will wait a random amount of time. Once this time has elapsed, we will listen again and so on until the channel is free and we can transmit.

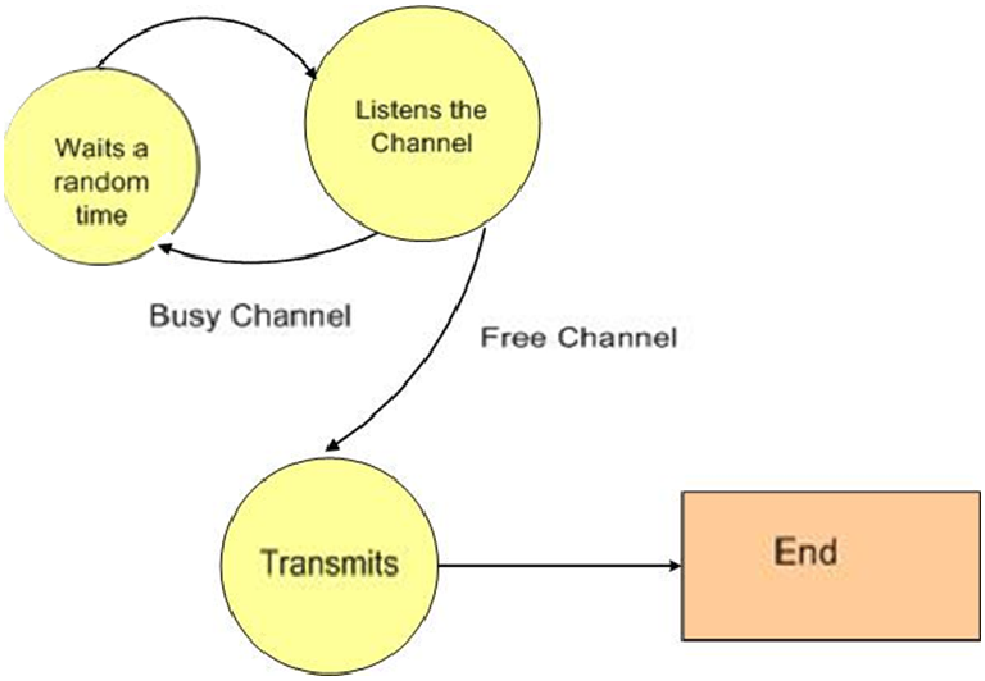


Figure 10 Carrier Sense Multiple Access with Collision Detection not persistent

This algorithm is the best choice because it will not consume any power until it is sure that can transmit, avoiding any waste of power on lost transmissions or simply listening all the time. Also, it provides good throughput on high traffic conditions, which will be the rule in our video streaming net [15].

4.1.2.4 Error detection

When a node receives a package, there's a chance that we may find distortion or alterations in the data. If we don't include a set of mechanisms in order to know this, we could relay incorrect data to the next node, which is not acceptable.

This set of tools enables us to detect and mark errors produced during the transmission. There are a lot of strategies to follow nowadays.

The kind of data packages that will be received will depends on the mission the node is realizing at the moment, but it will usually be a video data stream. Streaming packages which has been dubbed as erroneous will not be transmitted further, they will be discarded. Due to the sheer amount of data we manage on streaming, we do not find important to implement a fixing tool to recover damaged packages.

The best method we have available is the CRC we can add at the end of the frame.

The CRC is a mathematic method through which we can detect transmission errors. This technique consists in calculating the CRC parameter using part of the data and adding it at the end of the frame. Once the receiver has the

package, it will then recalculate the CRC in order to compare it with the received one.

If they coincide, there has been no error and we will transmit forwards. If not, the data has been somehow altered and no longer valid, so it will be discarded. The CRC parameter will expend 16 bits of the frame, which is the optimal length for the protocol due the length of the data encased.

4.1.2.5 Arranged frame distribution

This feature has no need to be implemented, since the CA does not need the packages to arrive in the right order.

4.1.2.6 Flux Control

This protocol will not account flux control, because in a streaming data flux structure, there will only be a single node working as PA, which in turn does not provide a great traffic.

4.2 Network layer

This layer is charged with the task of routing the packages inside the MAS. This is the reason why packages reach its final destination even if not directly linked, as shown in chapter 1 scenario.

***4.2.1** Routing*

Routing in an ad-hoc network is a delicate issue, because the non-fixed nodes will constantly alter the topology of the net and the QoS of its links. In order to route all the packages efficiently, we need an adaptive protocol where paths are refreshed dynamically.

There are two protocol groups for routing on ad-hoc networks [14]:

- The proactive approach
- The reactive approach

In the proactive strategy, each node always has a path to final destination. This way, said protocol tries to have always a “fresh” routing table for all the network nodes. In order to do so, the nodes will have to be in constant communication between them, leading to an overcharged network case.

The reactive approach, in the other hand, only refresh the routing table when data needs sending, saving both power and band width. The down hand is that it adds an important delay to total transmission time.

Due to we will not be always transmitting, we will use the reactive approach, more suitable to the MAS structure.

4.2.1.1 Path selection

Once the routing protocol has been established, now we have to explain how to choose the best path to the final destination.

As said, we will be using the reactive approach, only refreshing the route tables when needed. In the MAS scenario, this process starts when the CA begins the data transfer, and it alerts the PA.

- When the CA wants to start transmitting, it will overflow the net with R_REQ (Route Request) packages on broadcast.

- Each node receiving this package will check:
    - Source field.
    - Package ID field.
    - Hop count field.
    - Destination field.

  If source field and package ID coincide with a previously received package, we will compare the hop count field. If the new package has a greater value, it will be discarded. If not, destination field will be checked. If the node is not the final destination, it will add his address to the path queue saved in the data frame and broadcast it again.

- Once a R_REQ package reaches the final destination, the node will send a R_REP (Route Reply) following the same path it has taken from the source backwards, thanks to the queue saved in the data field. When the relay nodes receive a R_REP, they will create a new routing table only valid for a fixed time (Lifetime field).

- If the CA does not receive the R_REP within a limited time, it will start again sending a new R_REQ a maximum of three times. If I has not been able to establish a route by then, it will assume that the link is down and no route can be mapped.

The following diagram exposes graphically the full routing process:

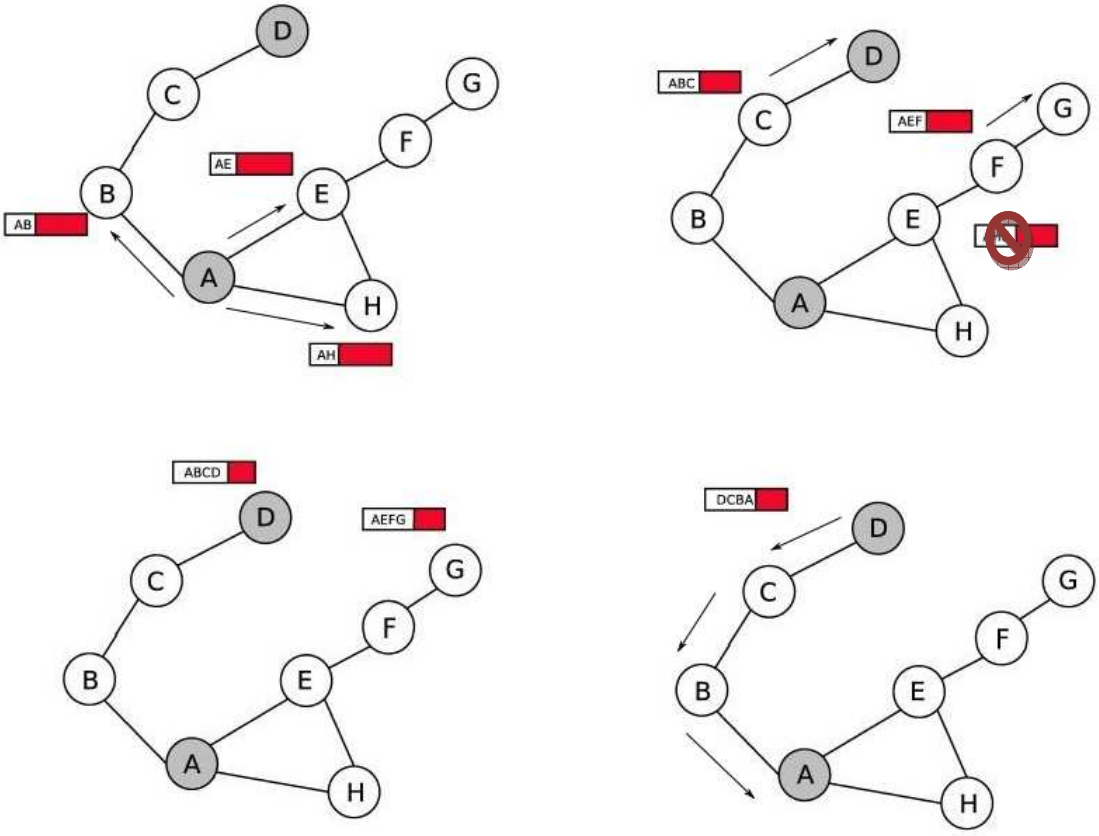


Figure 11 Routing Process

4.2.1.2 Kinds of packages

On this protocol we can differentiate between three types of packages:

- DATA package
- R_REQ package
- R_REP package

Some of these packages has already been defined. Now we will present them in detail:

- DATA package: This package is charged with transporting data between nodes.

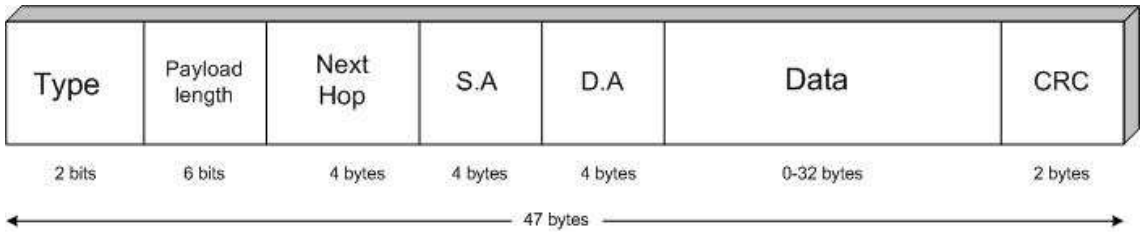


Figure 12 Data package

- Type (2 bits): This field is shared by all the frames on network layer, and identifies the kind of frame. Because we only work with three types, a two bit field will suffice; being ‘01’ the DATA frame identifier.
- Payload length (6 bits): This field shows the length of the data field, which can vary from zero to thirty two bytes. We have diverted 6 bits to this field, which is more than we actually need, but can easy future expansions of the frame.
- Next Hop (4 bytes): Field containing the address of the next agent.
- Source Address (4 bytes): This field contains the address of the source agent.
- Destination Address (4 bytes): This field contains the address of the final destination agent.
- DATA (0-32 bytes): This length is provided by the platforms [16] [17] that the protocol will use.
- CRC (2 bytes): This field contains the CRC parameter that allows us to the error checking.

- Package R_REQ: This package is charged with the path routing.

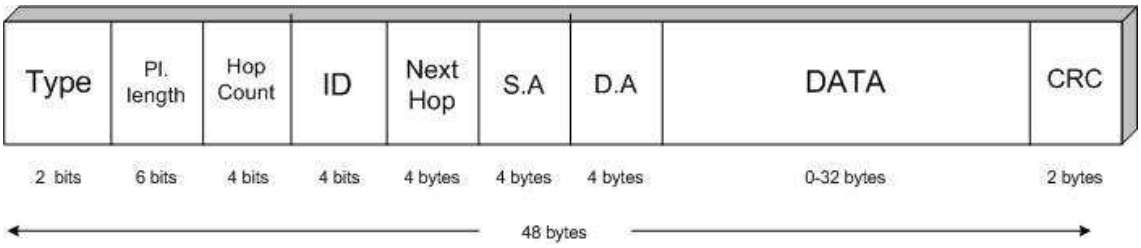


Figure 13 Route request packet

- Type (2 bits): This field is shared by all the frames on network layer, and identifies the kind of frame. Because we only work with three types, a two bit field will suffice, being ‘10’ the R_REQ frame identifier.
- Payload length (6 bits): This field shows the length of the data field, which can vary from zero to thirty two bytes. We have diverted 6 bits to this field, which is more than we actually need, but can easy future expansions of the frame.
- Hop Count (4 bits): Used to know how many links the package has crossed so far. It has been dimensioned for a 6 node MAS.
- ID (4 bits): This field allows us to identify the route request.
- Next Hop (4 bytes): Field containing the address of the next agent.
- Source Address (4 bytes): This field contains the address of the source agent.
- Destination Address (4 bytes): This field contains the address of the final destination agent.
- DATA (0-32 bytes): This field contains the addresses of all the nodes the query have passed through so far [16].
- CRC (2 bytes): This field contains the CRC parameter that allows us to the error checking.

- Package R_REP: This package is charged with the path routing.

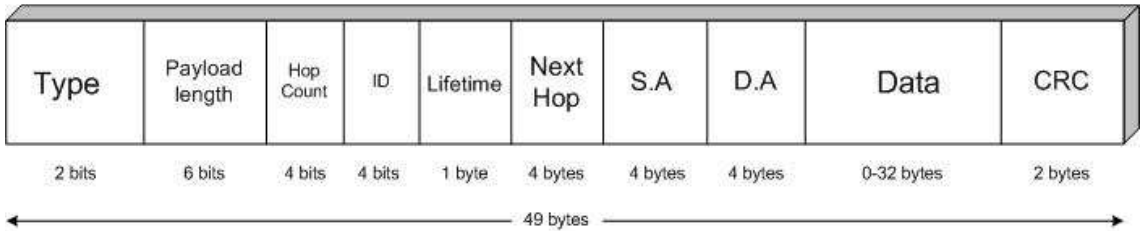


Figure 14 Route replay packet

- o Type (2 bits): This field is shared by all the frames on network layer, and identifies the kind of frame. Because we only work with three types, a two bit field will suffice, being '11' the R_REP frame identifier.
- o Payload length (6 bits): This field shows the length of the data field, which can vary from zero to thirty two bytes. We have diverted 6 bits to this field, which is more than we actually need, but can easy future expansions of the frame.
- o Hop Count (4 bits): Used to know how many links the package has crossed so far. It has been dimensioned for a 6 node MAS.
- o ID (4 bits): This field allows us to identify the route request.
- o LifeTime (1 byte): This value limits the lifetime of the current route.
- o Next Hop (4 bytes): Field containing the address of the next agent.
- o Source Address (4 bytes): This field contains the address of the source agent.
- o Destination Address (4 bytes): This field contains the address of the final destination agent.
- o DATA (0-32 bytes): This field contains the addresses of all the nodes the query have passed through so far [16].
- o CRC (2 bytes): This field contains the CRC parameter that allows us to the error checking.

4.2.1.3 Route refresh

The red MAS is mobile, so mapped paths will not last indefinitely. That is why we must refresh the routing tables from time to time, constantly assuring that the path used is valid and the most optimal at any time. In order to avoid path duplicity, each path will have an attached lifetime, a numeric value that limits the validity of the path.

The lifetime equation (4.1) will provide the maximum time, which will always be at top the transmitting time from source to destination. We obtain it through this formula:

$$LifeTime_{min} = N(t_{tx} + t_{process} + t_{prop}) \quad (4.1)$$

Where N is the number of links crossed between source and destination.

4.2.1.4 Metric

The metric used on the routing protocol is particularly important. There exist a lot of standards: RTT, PktPair, ETX, Min_HOP…

We must find a metric that suits the needs of our protocol, remembering that we work in a wireless scenario where the shortest path is not always the fastest nor the most efficient.

The most suitable metric was found in an UPC study [13] about the metric efficiency in scenarios with few nodes and minimum external effects. In said study, a comparison is drawn between several metrics and the MIN_HOP metric, one of the most popular nowadays because it aims to minimize the hop count from source to destination.

The conclusion we extract from the study is that MIN_HOP metric overall performance is superior to any other one given the simplicity of its implementation. Other metrics do have better workouts in specific areas, but are also much more complex to implement.

Table 1 Metric

	PATH_DR	MIN_HOP	PERIODIC_RREQs
Total TX	6000	6000	6000
Total RX	5725	5633	5863
PDR total	95,41 %	93,88 %	97,7 %
Total HOP	46,89	46	47,21

CHAPTER 5. Adaptive protocol implementation

In this chapter it will be explained the process of implementation of the full protocol.

With this in mind, it has been used an object oriented software (C#) in order to represent the classes and functions that will be necessary during the implementation.

5.1 Implementation design

In order to realize the implementation, it has been chosen to abstract the needed functions and later move them to other platforms.

To be able to understand the whole code, the chapter has been split in two main issues: class use and agent behaviour.

5.1.1 Class diagram

To represent all the classes used in the implementation of the protocol, we will use a class diagram (also known as UML). In it we can see which classes will be used, along with its methods and features.

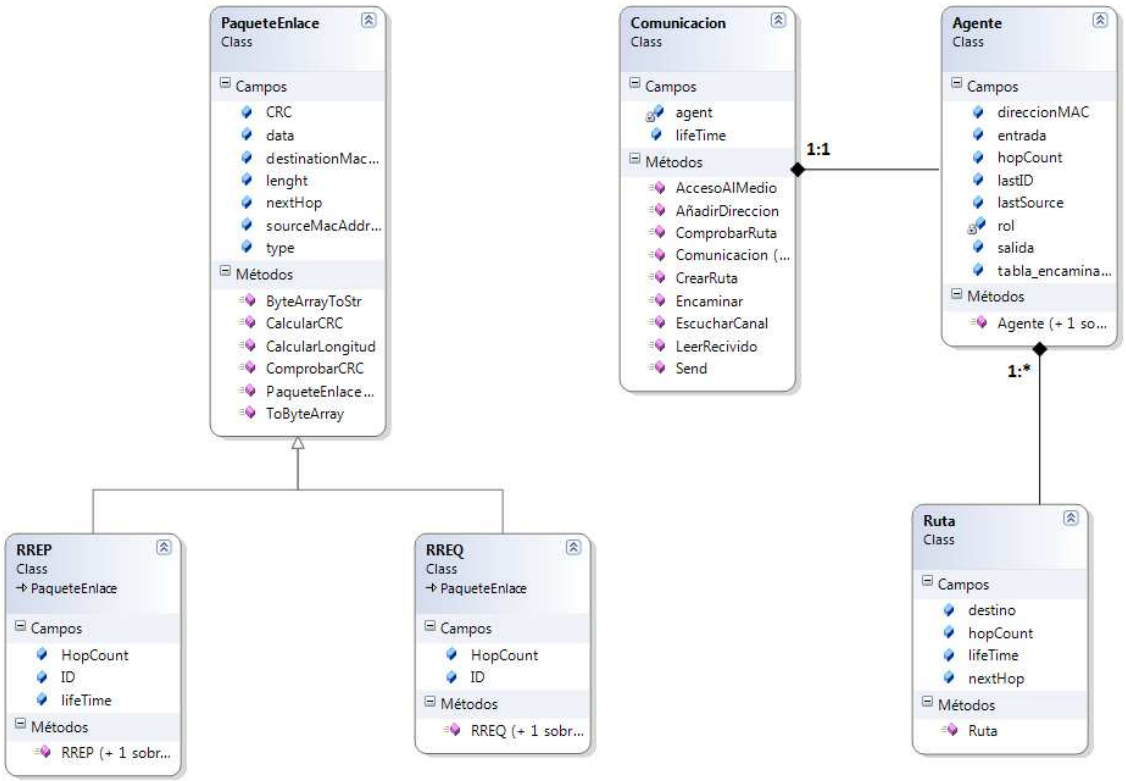


Figure 15 Class diagram

In these classes are represented all the elements of our protocol. We can see the three kinds of packages, the agent, the path and other communication elements. More extensive information about the classes is shown at the annex, but for now we will make a short briefing on each class:

- Package Link Class: represents all data packages. It consists of all the fields of the header and any data they contain. Its methods are package creation oriented.
- R_REP y R_REQ classes: these two classes represent the packages used on routing, intimately related with its parent class, the package link. They share methods and features with the data packages, and have some of their own.
- Agent class: it represents each and all of the agents of the MAS, and their features are oriented towards all the elements that fully define our agents.
- Path class: it lets us to represent the routing table of each agent, enabling the routing mechanics.
- Communication class: the most important of them all, it contains all the methods needed for our agents in order to communicate between themselves.

5.1.2 State diagram

Once observed all the different classes and its methods, we will explain the work and decision making process of the agents. For it, a state diagram will be used, where all agents start at RX state.

As can be seen, each agent must take some decisions depending on the feed he gets from the MAS environment. This rules the behaviour of each agent according with the protocol design we established before.

DIA[4] software has been used in the making of the class diagram.

[4] http://live.gnome.org/Dia

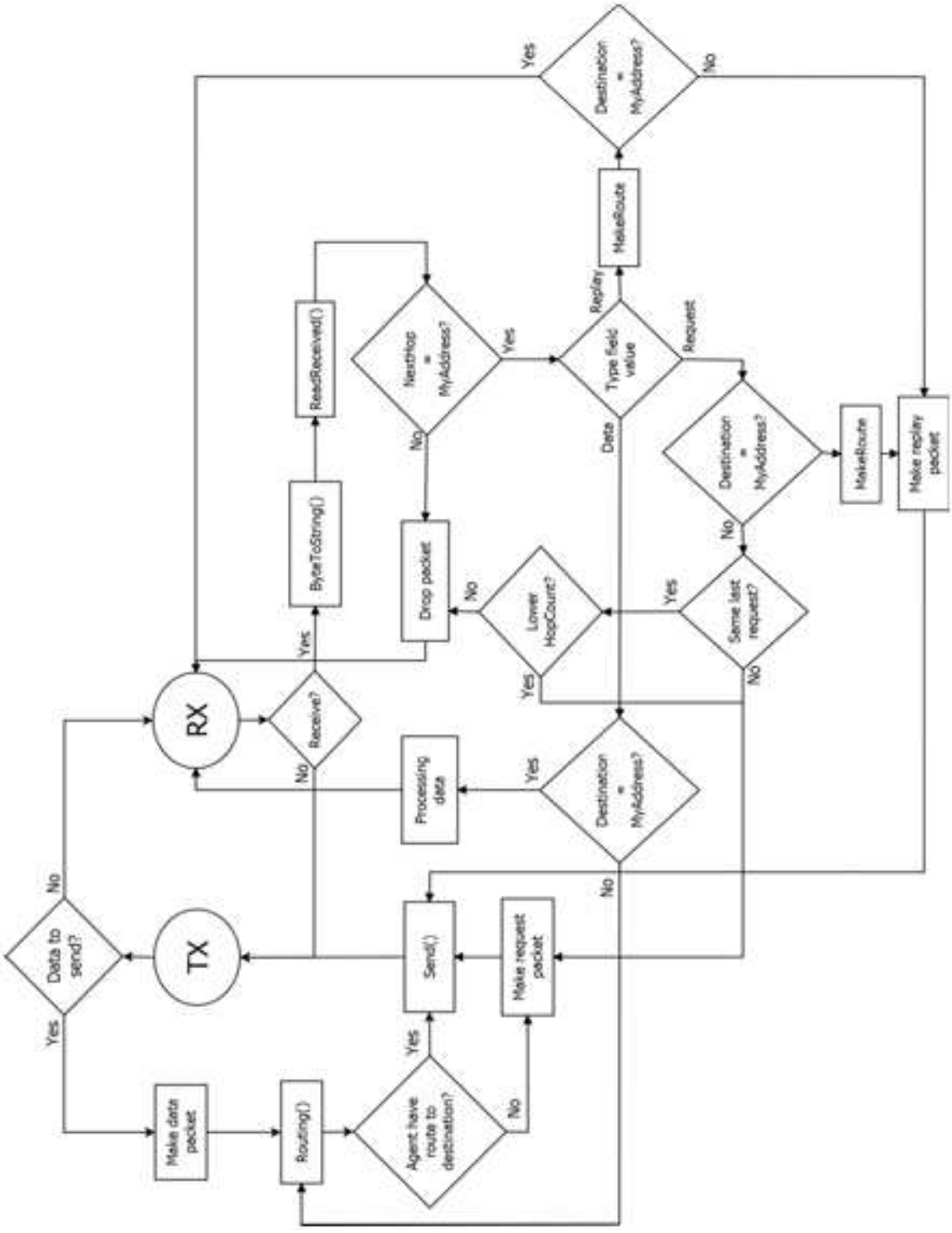


Figure 16 States diagram.

CHAPTER 6. Case study – Femto-satellite constellation

In this chapter a case study of a Femto-satellite constellation is presented in order to understand the established protocol and to test the rules shown in the chapter 3. The use of the protocol is justified in terms of to take the best benefit of the transmittion channel where the Producer agent can be changing, thus the Consumer agent will be the same.

6.1 Scenario

A constellation of Femto-satellites is injected in a 250 kilometres LEO orbit. Each satellite goes far from the others one kilometre per day and due the weak atmosphere, and the constellation stays in orbit for one month. The mission consists in recording any celestial phenomenon. Each Femto-satellite has a HD camera and an Inertial Platform. When the celestial phenomenon occurs, a low resolution picture of every satellite is sent to the GS. In addition, due to the low orbit, the link is established for few minutes, so only one of the satellites will send a high resolution picture and others will in low resolution. The high resolution is from the satellite that has a better picture. In the Table 2 the coverage time is tabulated and it is a function of the maximum distance that corresponds to the sixth satellite. Calculations will be detailed in section 6.4.

Table 2 Schedule for the constellation

	Sat_1	Sat_2	Sat_3	Sat_4	Sat_5	Sat_6	Time
Day 00	0 km	0 km	0 km	0 km	0 km	0 km	256 s
Day 01	0 km	1 km	2 km	3 km	4 km	5 km	256 s
Day 02	0 km	2 km	4 km	6 km	8 km	10 km	257 s
Day 03	0 km	3 km	6 km	9 km	12 km	15 km	258 s
Day 04	0 km	4 km	8 km	12 km	16 km	20 km	258 s
Day 05	0 km	5 km	10 km	15 km	20 km	25 km	259 s
Day n	0 km	1*n km	2*n km	3*n km	4*n km	5*n km	

6.2 Definition of the agent

For this case study we will use a WikiSat V3 that is shown in Figure 17. The WikiSat has a High Gain Antenna that allows a 1,000 km link with a ground station in the 2.4 GHz frequency and a bandwidth of 250 kbps. This frequency is affected by the rain and Wireless devices so we will need a ground station network to download the information every day.

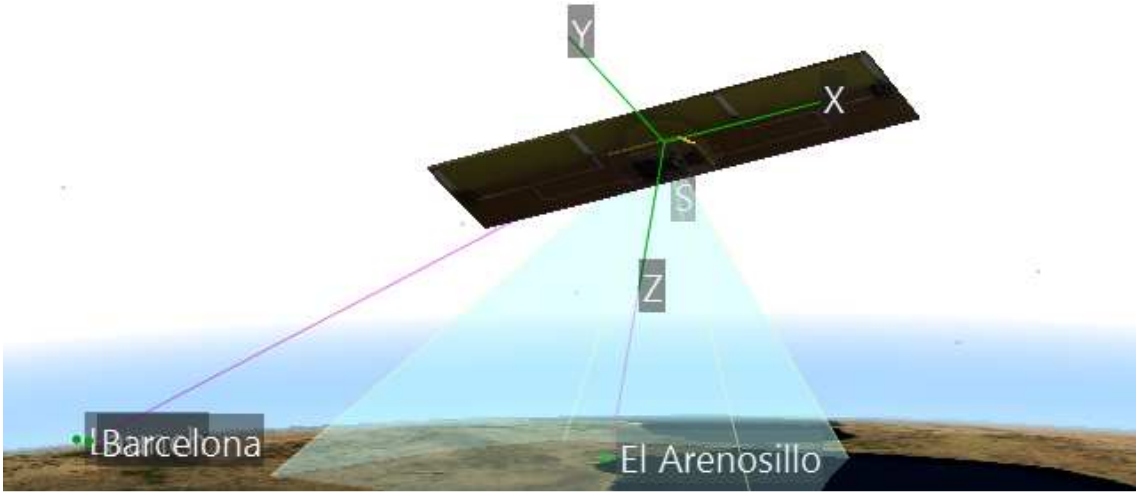


Figure 17 WikiSat V3 Femto-satellite as the agent

The HD camera is a TOSHIBA TCM8230MD[5], having a resolution of 660x492 pixels in 32 bits in a format called BGGR Bayer filter. It has up to 30 frames per second. The field of view (FOV) in the horizontal axis is 57.4 degrees and 44.5 degrees in the vertical axis.

The inertial platforms consist of 3 axis gyros and 3 axis accelerometers. The main processor unit is an ATmega328.

6.3 Wikisat ground station network

Let us consider that we use a ground station network consisting of four stations:

- Copenhagen (Hold by OZ9AEC)
- Barcelona (Hold by UPC)
- El arenosillo (Hold by INTA)
- Maspalomas (Hold by IDeTIC)

6.3.1 OZ9AEC

The main interest of the group OZ9AEC created by Alexandru Csete is satellite communications, radio software and free software development.

The ground station is located in Copenhagen. Copenhagen has a latitude of 55 º 40 'and longitude 12 °35'. We have chosen thi s ground station as this group has contributed directly to the Team FREDNET GROUP.

6.3.2 UPC

University group that supports WikiSat. Located in Barcelona with a north latitude of 41 °18 'and longitude 2 °06'.

[5] http://pdf1.alldatasheet.com/datasheet-pdf/view/227785/TOSHIBA/TCM8230MD.html (Feb/2011)

6.3.3 INTA

INTA (National Institute of Aerospace Technology), a group that aims to acquire, maintain and improve technology in the aerospace field. Arenosillo base is located near Mazagon (Huelva) with a latitude of 37 º 07 'west longitude and 6 °50'.

6.3.4 IDeTic

IDeTic (Instituto para el Desarrollo Tecnológico e Innovación en Comunicaciones) is a group that have hoped to maintain, and to enhance, the work research, deepen the integration between the groups.

The ground station is localized in Gran Canarias. Gran Canarias is in 28º North latitude and 15º35' West length. This ground station was chosen at this point because in this city can be found the IDeTic group that take part of the WikiSat group.

6.4 Downlink window

The information will be downloaded once every day. Based on Laia Morcillo[9] calculations, the satellite has a typical orbital speed of 7,516 m/s (27,000 km/h) and a period of 5,369 seconds; what means 1 hour, 29 minutes and 29 seconds per revolution or orbit. The revisiting period is about 85,876 seconds; which is 1 day, 57 minutes and 53 seconds that happens each 16 orbits.

In Figure 18 there is a coverage map when the satellite passes over one of the four ground stations. Every day there are two or three opportunities that helps in case of bad weather. The constellation is in orbit for 5 days.

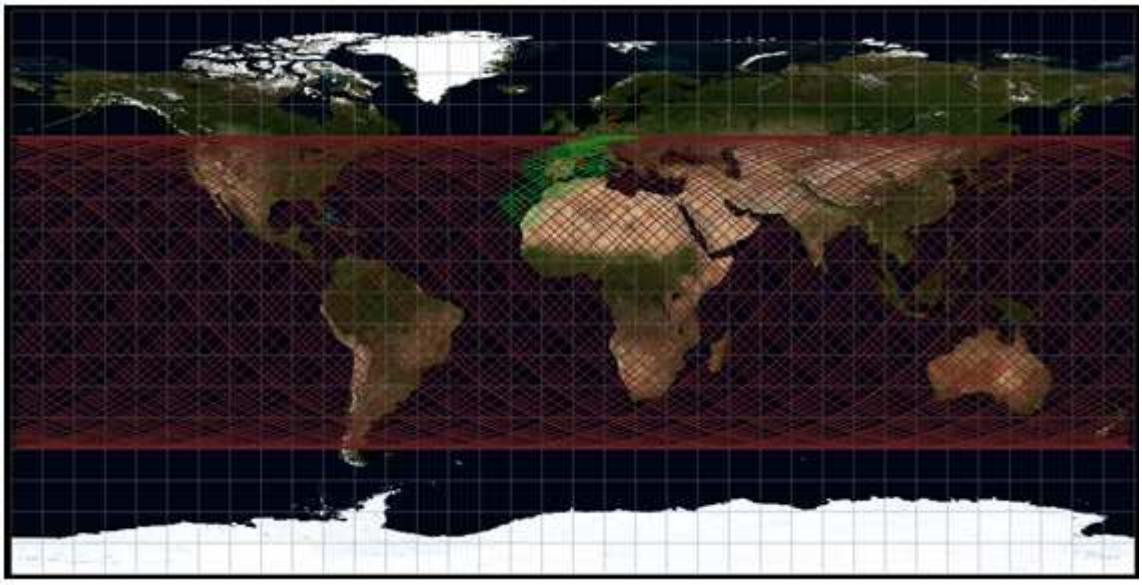


Figure 18 Coverage map (GREEN) for the WikiSat V3 and the WikiSat ground station network

The coverage time or visibility orbit sector (Angle β) can be calculated as the angle or arc of the orbit (6.1) where the satellite is visible from the ground station and an Earth central body reference system. The visibility angle β can be calculated by iteration means for a given elevation and orbit; see Figure 19.

$$Cos\ \beta - Tan\ \varphi\ Sin\ \beta = R/(R+h) \qquad (6.1)$$

Where:

β: Visibility angle in the orbit
φ: Minimum ground station elevation $\varphi = 10^{o}$
R: Earth radius $R = 6{,}371$ km
h: Orbit altitude $h = 250$ km

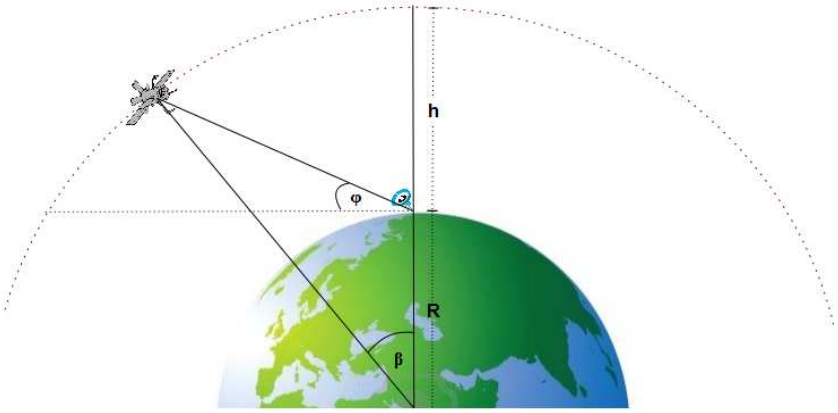
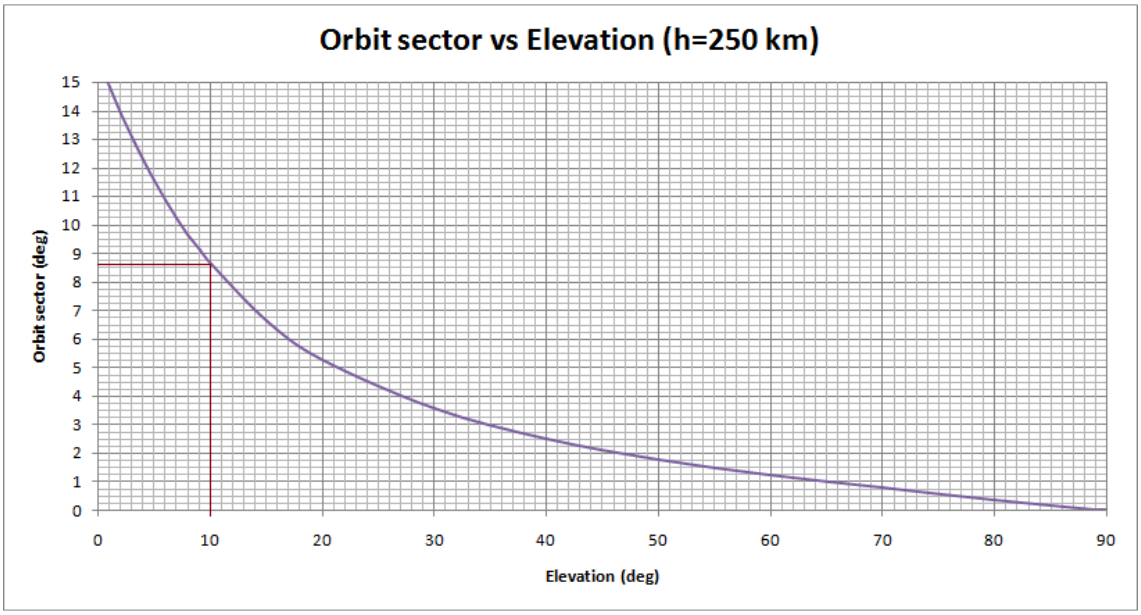


Figure 19 Visibility (Orbit sector **β**) from a ground station and a given elevation (**φ**)

Table 3 Visibility angle β (Orbit sector) for a 250 km LEO orbit and ground station elevation (**φ**)



Based on Table 2 the orbit sector is about 8.6 degrees that for a period of 5,369 seconds corresponds to a visibility time of $t_{visibility} = 5{,}369\ s * 2 * 8.6 / 360 = 256\ s$. The orbit perimeter is $P_{250km} = 2 * pi * 6371\ km + 250\ km = 41{,}600\ km$. In the better case, the fact that the constellation has a maximum separation of 25 km, it only represents 3 seconds of extra coverage time $t_c = 256 + 3 = 259\ s$. Based on this

calculations, the best strategy is that only one satellite should be responsible for the connection with the ground station. The fore satellite will be the so called Consumer agent.

Based on simulations, the average visibility time is about 235 seconds, around 4 minutes. In the case of best visibility, the maximum amount of information able to be downloaded in a single event is $\text{memory}_{\text{max}} = 230\ \text{s} * 250\ \text{kbps} = 7.2\ \text{Mbytes}$. Since there are up to eight passes every day for at least one of the Wikisat ground station network, let's keep this value as a reference.

6.5 Results of the paradigm over this case study

As seen in this case study, is not necessary to switch the CA because it will always be those who first see the GS. Yet the choice of PA will remain dynamic and will route the information that it captures to the CA. It will therefore be necessary to use the protocol set forth.

CHAPTER 7. Case study - PicoRover swarm

In this chapter a case study of a PicoRover swarm is presented in order to understand the established protocol and to test the rules shown in the chapter 3 in extreme and dynamic conditions. The use of the paradigm is justified in terms of being adaptive to the unknown map and at the same time to keep the contact with all the agnets. The Producer agent and the Consumer agent can be changing depending on a basic rule.

7.1 Scenario

In this scenario, there are an swarm of five equal robots. Each agent should be in contat with others agents in the swarm. One or more agents can not be isolated from the rest in order to guarantee the transmittion inside the swarm. The agents are over the surface of the Moon. There is not atmosphere so only line of sight allow us a communication so if in any moment any agent can not see other, they lose the contact. Typical maximum distance is about 50 meters.

The mission consists of to achieve the maximum distance in order to record a video generated by the Producer. In the beginning all the agents are deployed from the same poit, the Lunar Lander. The role of Producer will be set in terms of the agent which is far enough from the Lunar Lander. The computing of this rule will be done in the Lunar Lander. Also, the role of Consumer will be set to the agent that is closer to the Lunar Lander. Following the proposed protocol, decissions are taken in the Lunar Lander and the network provides updated information of the state of each agent, in example, the distance.

7.2 A Definition of the agent

The agent is a PicoRover[6]. It is a Lunar Rover designed to win the Google Lunar X Prize, which consists in sending your own lunar rover to the Moon, travel 500 meters and transmit video, images and data back to the Earth.

The prototype is a ball of 12 cm of diameter and low weight less than one kilogram, having the indispensable to fulfill the goals of the Prize: a motor, a battery, a tiny computer and a high definition camera. The ball has been done with low cost materials, such as bulbs, wire and glass fiber, although it bears the extreme lunar temperatures, it adheres easily to the lunar surface and it climbs slopes up to 33 degrees.

The PicoRover, as a explorer is an autonomous vehicle that moves in a constant direction. If the PicoRover senses an obstacle or a high slope angle, it tries to avoid them. When it detects a weak communication, before to lost the link, it stops and back until it recovers good signal of the link. There is a point when forces between expand and keep tigether cancels and it is when the PicoRover stops.

[6] http://www.cienciaenaccion.org/inscription/picorover-un-robot-lunar-de-bajo-coste

7.3 Ground station network

The ground station is achieved by the Lunar Lander which is not an agent and it can not move. The Lunar Lander has a large antenna that is used for relaying with the Earth's ground station network.

7.4 Logistics

It is true that the ideal geometric distribution is a stright line of agents at the maximum distance of 50 meter each to have the maximum theoretical distance of 250 meters. In practice, there are many obstacles and high slopes are not achiveable by the agents. In figure

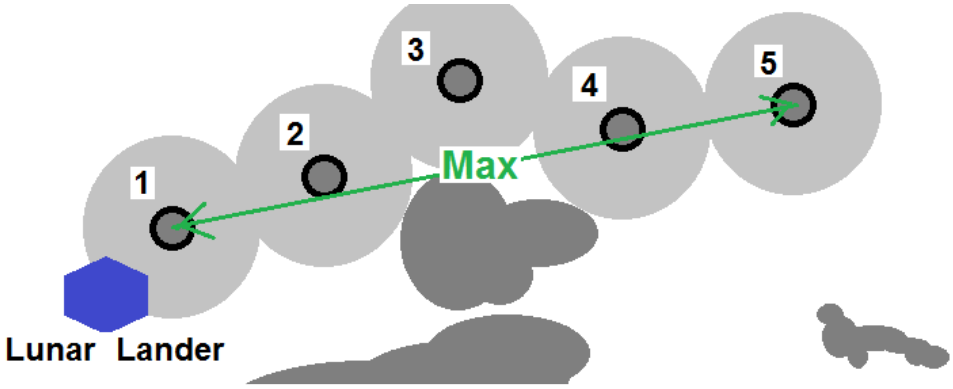


Figure 20 Example of geometrical configuration for a PicoRover swarm

Some times, it is possible to see a tight wave that makes the PicoRover move forward and backwards. Also lateral tensions appears that introduces some vibration to the swarm. There is not a single solution and the swarm never stops and explores like a snake.

7.5 Results of the paradigm over this case study

This is a robust protocol that helps to keep the swarm together. In the previous case, Figure 20, the rovers had followed random direction and because rover 5 has got better propagation, it will be the Producer. Also the worst propagation designed the Consumer agent.

Not scaning it is required for the rovers because the rover algorithm ensures an imaginary force that keeps togrther the swarm and faster they achieve the maximum distance.

CHAPTER 8. Environmental impact

The purpose of this TFC is the communication between several agents, always within a framework aerospace.

This framework always has to think where this waste lead, from chemicals used to perform the shuttle to the energy needed to produce each vehicle. All of them are adverse on the environment.

Precisely for this reason, we want to reduce the impact this cause on Earth. One of the main pillars in the construction of each of the devices is optimizing resources and re-use of materials.

Also we must keep in mind the debris issue. In this case, instead of performing a single large satellite, we will be using a swarm of small satellites. Should they enter the atmosphere, they would disintegrate before reaching Earth due to their small dimensions. This is also valid for the shuttle.

Thus, both with the TFC as with many other tasks, we are trying to develop aerospace technology that allows man research the space, always taking into account the environmental impact would result.

CONCLUSIONS

The aim of this studio was to get a protocol that allows agents to communicate inside MAS.

Upon completion of this work, we have noted the importance of the analysis of agents and the environment, since the protocol has to adapt to the needs and capabilities of the system.

It has also been observed that we can modify existing standards to better suit the objectives, although it is very useful to draw on them.

And the most important conclusion is that, even without an actual implementation for results, it is considered that the protocol would be feasible and would meet the TFC requisites. It will also help a lot in the future groups like WikiSat and FredNet to achieve their goals.

Also indicate that some objectives have not been achieved due to time and / or poor planning, but remains open to future research to improve it. Some of these future improvements could include:

- Real implementation and field study results
- Perform confirmation without losing stream data
- That all nodes could send info at the same time without adding too much delay, perhaps with other routing methods or changing the media access.

BIBLIOGRAPHY

[1] Bruninga, B. (callsign WB4APR), *APRS, aprs.org, Glen Burnie US. (1984)*

[2] Tristancho, J., *Implementation of a femto-satellite and a mini-launcher*, upcommons, Castelldefels. (2010)
http://upcommons.upc.edu/pfc/handle/2099.1/9652

[3] Jamison, T. K., *Rocket launching ramp and mechanism,* United States Patent Office, 2990143 (7), (1961)
http://www.freepatentsonline.com/2990143.html

[4] Arias, E., *Design and implementation of a second stage nozzle for a low cost mini-launcher*, upcommons. Barcelona. Submitted (2011)
https://mitra.upc.es/SIA/PFC_PUBLICA.DADES_PFC?w_codipfc=6519

[5] Rodríguez, R., *Study of a nozzle vector control for a low cost mini-launcher*, upcommons, Barcelona. Submitted (2011)
https://mitra.upc.es/SIA/PFC_PUBLICA.DADES_PFC?w_codipfc=6450

[6] J.A.A., Title 14, PART 101 "Moored balloons, kites, amateur rockets and unnamed free balloons", *Federal Aviation Requirements*, ecfr.gpoaccess.gov, (2001)

[7] David J. Barnhart, et all., *A low-cost femtosatellite to enable distributed space missions*, Acta Astronautica 64. Pages 1123–1143 (2009)
http://www.sciencedirect.com/science/article/pii/S0094576509000198

[8] http://www.ryerson.ca/news/news/Research_News/20100128_femto.html

[9] Morcillo, L., *Use of hardware-on-the-loop to test missions for a low cost mini-launcher*, upcommons, Barcelona (2011)
http://upcommons.upc.edu/pfc/handle/2099.1/11754

[10] Navarro-Morales, L., *A multi-agent payload mangement approach for femto-satellite applications*, upcommons, Barcelona (2011)
https://mitra.upc.es/SIA/PFC_PUBLICA.DADES_PFC?w_codipfc=6600

[11] Kravchencko,V., *Desing and implementation of a femto-satellite technology demonstrator*, upcommons, Barcelona (2011)

[12] Fernandez-Murcia, E., *Implementation and verification of a Lunar mission Subsystems*, upcommons, Barcelona (2011)
http://upcommons.upc.edu/pfc/handle/2099.1/8292

[13] Navarro-Morales, L., *A femtosatellite based on Commercial-Of-The-Shelf: Payload integration in the design cycle*, upcommons, Barcelona. Submitted (2011)

[14] Berzosa-Calpe, S., *Optimización de un protocolo de encaminamiento en redes de sensores IEEE 802.15.4,* upcommons, Barcelona (2011) http://upcommons.upc.edu/handle/123456789/2277

[15] Balado-Suarez, L., *Evaluación de los protocolos de acceso al medio CSMA/CD (Acceso Multiple con escucha de canal y detección de colisión) y paso de testigo, en redes locales con topología bus*, upcommons, Barcelona (1987) http://upcommons.upc.edu/handle/123456789/194528

[16] NORDIC Semiconductor*, nRF24L01+ Single Chip 2.4GHz Transceiver Product Specification v1.0*, (2008)

[17] Texas Instruments, *CC2500 Low-Cost Low-Power 2.4 GHz RF Transceiver.*

[18] Microsoft Developers Network website: http://msdn.microsoft.com/es-es/

Annex A

Abstraction functions and class codes, made by Joan Naudó.

```
//Class Agente

using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication2
{
    class Agente
    {
        string rol;
        public string direccionMAC;
        public List<Ruta> tabla_encaminamiento = new List<Ruta>();
        public byte[] entrada;
        public byte[] salida;
        public string lastSource;
        public string lastID;
        public int hopCount;

        public Agente(string rol,string direccionMAC)
        {
            this.rol = rol;
            this.direccionMAC = direccionMAC;
        }
        public Agente()
        {
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;


namespace ConsoleApplication2
{
    class Comunicacion
    {
        Agente agent = new Agente();
        public int lifeTime;

        public Comunicacion()
        {

        }
        public Comunicacion(Agente agent, int lifeTime)
        {
            this.agent = agent;
            this.lifeTime = lifeTime;
        }
```

```
        public void CrearRuta(RREP rep)
        {
            string
nextHopToDestination=rep.data.Substring(rep.data.Length-4,4);
            rep.data.Remove(rep.data.Length - 4, 4);
            string nextHopToSource =
rep.data.Substring(rep.data.Length - 8, 4);
            Ruta RouteToSorce = new
Ruta(rep.sourceMacAddress,nextHopToSource,lifeTime);
            Ruta RouteToDestination = new
Ruta(rep.destinationMacAddress,nextHopToDestination,lifeTime);
            agent.tabla_encaminamiento.Add(RouteToDestination);
            agent.tabla_encaminamiento.Add(RouteToSorce);
        }
        public void Send(PaqueteEnlace paqueteMAC)
        {
            AccesoAlMedio();
            agent.salida = paqueteMAC.ToByteArray();
            //Activar envio a nextHop
        }
        public void Encaminar(PaqueteEnlace paqueteMAC)
        {
            foreach(Ruta route in agent.tabla_encaminamiento)
            {
                if (route.destino ==
paqueteMAC.destinationMacAddress)
                    paqueteMAC.nextHop = route.nextHop;
            }
        }
        public PaqueteEnlace LeerRecivido()
        {
            PaqueteEnlace paqueteMac = new PaqueteEnlace();
            string packet =
paqueteMac.ByteArrayToStr(agent.entrada);
            paqueteMac.type = packet.Substring(0, 4);
            if( paqueteMac.type == "data")
            {
                paqueteMac.lenght = int.Parse(packet.Substring(4,
2));
                paqueteMac.nextHop = packet.Substring(6, 4);
                paqueteMac.sourceMacAddress = packet.Substring(10,
4);
                paqueteMac.destinationMacAddress =
packet.Substring(14, 4);
                paqueteMac.data = packet.Substring(18,
paqueteMac.lenght);
                paqueteMac.CRC = int.Parse(packet.Substring(18 +
paqueteMac.lenght, 2));
                return paqueteMac;
            }
            else if (paqueteMac.type == "Rreq")
            {
                RREQ req = new RREQ();
                req.type = paqueteMac.type;
                req.lenght = int.Parse(packet.Substring(4, 2));
                req.HopCount = int.Parse(packet.Substring(6,1));
                req.ID = packet.Substring(7,3);
                req.nextHop = packet.Substring(10,4);
                req.sourceMacAddress = packet.Substring(14, 4);
                req.destinationMacAddress = packet.Substring(18,
4);
```

```
                req.data = packet.Substring(22, paqueteMac.lenght);
                req.CRC = int.Parse(packet.Substring(22 +
paqueteMac.lenght, 2));
                return req;
            }
            else
            {
                RREP rep = new RREP();
                rep.type = paqueteMac.type;
                rep.lenght = int.Parse(packet.Substring(4, 2));
                rep.HopCount = int.Parse(packet.Substring(6, 1));
                rep.ID = packet.Substring(7, 3);
                rep.lifeTime = int.Parse(packet.Substring(10,2));
                rep.nextHop = packet.Substring(12,4);
                rep.destinationMacAddress = packet.Substring(16,
4);
                rep.sourceMacAddress = packet.Substring(20, 4);
                rep.data = packet.Substring(24, paqueteMac.lenght);
                rep.CRC = int.Parse(packet.Substring(24 +
paqueteMac.lenght, 2));
                return rep;
            }

        }
        public void AccesoAlMedio()
        {
            bool libre = EscucharCanal();
            int tiempo=500;//Tiene que ser aleatorio

            while (!libre)
            {
                System.Threading.Thread.Sleep(tiempo);//esperamos
un tiempo randomm
                libre = EscucharCanal();
            }
        }
        public bool EscucharCanal()
        {
            return true;
        }
        public void ComprobarRuta(RREQ req)
        {
            if(req.sourceMacAddress!=agent.lastSource ||
req.ID!=agent.lastID)
            {
                agent.lastSource = req.sourceMacAddress;
                agent.lastID = req.ID;
                agent.hopCount = req.HopCount;
                req.HopCount++;
                Send(req);
            }
            else
            {
                if (req.HopCount < agent.hopCount)
                {
                    agent.hopCount = req.HopCount;
                    req.HopCount++;
                    Send(req);
                }  // Si no se entra en el if, se descarta el
paquete
```

```
            }
        }
        public void AñadirDireccion(RREQ req)
        {
            req.data = req.data + agent.direccionMAC;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication2
{
    class PaqueteEnlace
    {
        public string type;
        public string sourceMacAddress;
        public string destinationMacAddress;
        public string nextHop;
        public int lenght;
        public int CRC;
        public string data;

        public PaqueteEnlace()
        {

        }
        public PaqueteEnlace( string nextHop,string
sourceMacAddress,string destinationMacAddress,int CRC, string data)
        {
            this.type = "Data";
            this.nextHop = nextHop;
            this.sourceMacAddress = sourceMacAddress;
            this.destinationMacAddress = destinationMacAddress;
            this.CRC = CRC;
            this.data = data;
            lenght = CalcularLongitud();
        }

        public void CalcularCRC()
        {
            //Calculo para CRC-16 bits
        }
        public bool ComprobarCRC()
        {
            //Se vuelve a calcular el CRC. SI no coincide con el
que recibimos, se descarta el paquete
            return true;
        }
        public int CalcularLongitud()
        {
            int tamaño = sourceMacAddress.Length +
destinationMacAddress.Length + type.Length + data.Length + 2;
            return tamaño;
        }
        public byte[] ToByteArray()
        {
            string packet = type + sourceMacAddress +
```

```
destinationMacAddress + lenght.ToString() + data + CRC.ToString();
            System.Text.UTF8Encoding encoding = new
System.Text.UTF8Encoding();
            return encoding.GetBytes(packet);
        }
        public string ByteArrayToStr(byte[] dBytes)
        {
            System.Text.UTF8Encoding enc = new
System.Text.UTF8Encoding();
            string packet = enc.GetString(dBytes);
            return packet;

        }
    }
}


using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication2
{
    class RREP:PaqueteEnlace
    {
        public int HopCount;
        public string ID;
        public int lifeTime;

        public RREP( string nextHop,string sourceMacAddress, string
destinationMacAddress, int CRC, string data, int HopCount, string
ID,int lifeTime)
            : base( nextHop,sourceMacAddress,
destinationMacAddress, CRC, data)
        {
            type = "reply";
            this.HopCount = HopCount;
            this.ID = ID;
            this.lifeTime = lifeTime;
        }
        public RREP()
        {
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication2
{
    class RREQ:PaqueteEnlace
    {
        public int HopCount;
        public string ID;

        public RREQ(string nextHop,string sourceMacAddress, string
destinationMacAddress, int CRC, string data, string ID)
            : base(nextHop,sourceMacAddress,
```

```
destinationMacAddress,CRC,data)
        {
            type = "request";
            this.HopCount=0;
            this.ID=ID;
        }
        public RREQ()
        {
        }
    }

    class CopyOfRREQ : PaqueteEnlace
    {
        public int HopCount;
        public string ID;

        public CopyOfRREQ(string nextHop, string sourceMacAddress,
string destinationMacAddress, int CRC, string data, string ID)
            : base(nextHop, sourceMacAddress,
destinationMacAddress, CRC, data)
        {
            type = "request";
            this.HopCount = 0;
            this.ID = ID;
        }
        public CopyOfRREQ()
        {
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication2
{
    public class Ruta
    {
        public string destino;
        public string nextHop;
        public int hopCount;
        public int lifeTime;

        public Ruta(string destino, string nextHop, int lifeTime)
        {
            this.destino = destino;
            this.nextHop = nextHop;
            this.lifeTime = lifeTime;
        }
    }
}
```