ESEIAAT

Cubesat Constellation
Astrea

Report

Degree: Aerospace Engineering
Course: Engineering Projects
Group: G4 EA-T2016

Delivery date: 22-12-2016

Students:

Cebrián Galán, Joan
Foreman Campins, Lluís
Fuentes Muñoz, Óscar
Harrán Albelda, Fernando
Martínez Viol, Víctor
Pla Olea, Laura
Puig Ruiz, Josep
Tarroc Gil, Sergi
Urbano González, Eva María
Fontanes Molina, Pol
Fraixedas Lucea, Roger
González García, Sílvia
Kaloyanov Naydenov, Boyan
Morata Carranza, David
Pons Daza, Marina
Serra Moncunill, Josep Maria
Tió Malo, Xavier

Customer: Pérez Llera, Luís Manuel

Contents

List of Tables

List of Figures

Part I

Communications

Chapter 1

Space Segment Protocol Stack

"The wonder is, not that the field of stars is so vast, but that man has measured it."

Anatole France, 1894

1.1 Introduction

Over this chapter, the **space communication protocols** are going to be defined. That is, a set of rules are going to be established in order to achieve the actual node-to-node communication. Altough the scope of the chapter is limited to the space segment, this inital introduction on the protocol definition is usfeul for the ground segment. Having said that, several factors constrain the design of this relation of rules:

- **Speed:** As it has already been mentioned, each node should be capable of handling at least **25** Mbit/s. Even though this doesn't mean that the design should be able to fit 25 Mbit/s of pure costumer data, it is still a strong requirement with many effects over the system. For example, some protocols are just too slow establishing the connection; those will be directly discarded.
- **Reliability:** The protocols have to assure that the messages are going to arrive to their destination. In order to achieve this, a routing protocol has to be used as well.
- **Security:** Messages are not just required to arrive to their destination but they also must be ordered and coherent when they reach the client. That is the reason why error control is taken into consideration very seriously along the design process.

In the diagram 1.5.1, it can be clearly seen the structure of a protocol stack. Each layer has an underlying protocol, designed to achieve a specific task. There are **low-level** protocols, dealing with *hardware* or with the establishment of the *physical* path between two nodes. Also, there are the **high-level** protocols, dealing with session control, optimum *logical* paths generation and bridging the application layer with the physical layer.

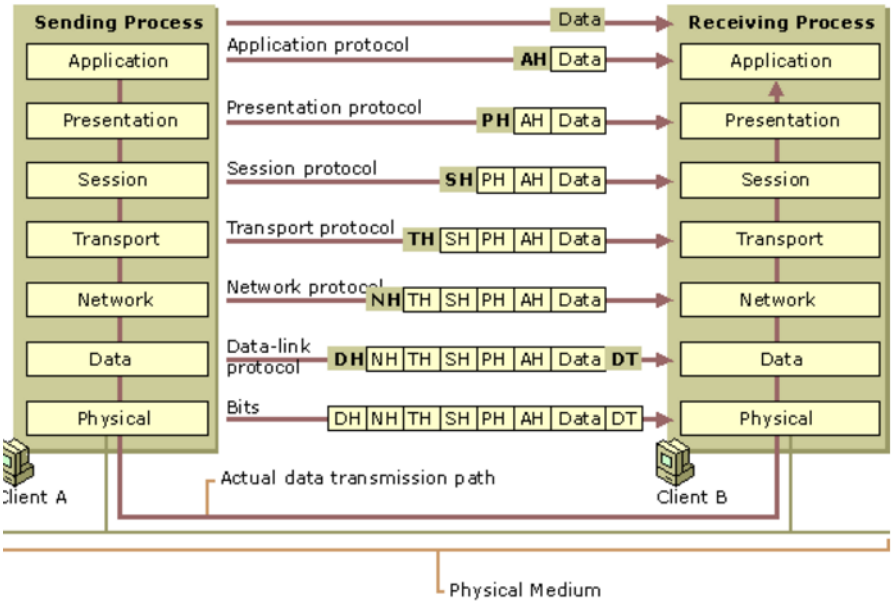


Figure 1.1.1: OSI Model layers

As it can be deducted, both the sending and the receiving node have to work in coordination. Also, each layer is designed with encapsulation in mind. Therefore, each layer does not depend on the others and develops its own task independently.

This philosophy has been started by the *International Telecommunications Unit* or **ITU** who first proposed, in the late 1970, the previously depicted **OSI Model**. Basiaclly, it establishes a conceptual framework for when new protocols are to be designed. Each layer can be understood easily if one thinks as the act of sending mail by post, for instance.

As far as *Astrea constellation* is concerned, the physical layer is already defined in detail in the **Satellite Desgin** part. Since the **Data Link layer**, the **Network layer** and the **Transport/Session layer** are of vital importance for the communication to work, the aim of the next sections will be to define the protocol that the constellation will be using for each one of those layers.

The presentation and the application layer are more client oreinted. In other words, if one client's satellite sends some data formatted with an unknown application protocol, *Astrea* will not be affected in any way. What astrea will do is add to this stream of bits, some headers, in order for the message to arrive in time to its destination. This methodology is undoubtedly positive for *Astrea* since the respobility of the application data will be solely for the customer.

1.2 Layer 2: Data Link

1.2.1 Functions of the DLL

The Data-Link layer is the protocol layer in a program that handles the moving of data in and out across a physical link in a network. The Data-Link layer is layer 2 in the Open Systems Interconnect (OSI) model for a set of telecommunication protocols. According to the IEEE-802 LAN standards, the DLL can be divided into two sublayers:

- Logical Link Control (LLC): Deals with protocols, flow-control, and error control.
- Media Access Control (MAC): Deals with actual control of the media.

The DLL is responsible for converting data stream to signals bit by bit and to sent that over the underlying hardware.At the receiving end, DLL picks up data from hardware which are in the form of electrical signals, assembles them in a recognizable frame format, and hands over to upper layer. The DLL also ensures that an initial connection has been set up, divides output data into data frames, and handles the acknowledgements from a receiver that the data arrived successfully. It also ensures that incoming data has been received successfully by analyzing bit patterns at special places in the frames. The specific functions of the DLL are explained in the following lines.

- **Framing**: Data-link layer takes packets from Network Layer and encapsulates them into Frames.Then, it sends each frame bit-by-bit on the hardware. At receiver end, data link layer picks up signals from hardware and assembles them into frames.
- **Adressing**: Each device on a network has a unique number, usually called a hardware address or MAC address, that is used by the data link layer protocol to ensure that data intended for a specific machine gets to it properly.
- **Synchronization**: When data frames are sent on the link, both machines must be synchronized in order to transfer to take place.
- **Error control**: Sometimes signals may have encountered problem in transition and the bits are flipped. These errors are detected and attempted to recover actual data bits.
- **Flow control**: Stations on same link may have different speed or capacity. Data-link layer ensures flow control that enables both machine to exchange data on same speed.

1.2.2 Working procedure

In the previous section, the functions of the DLL have been determined. Now, the way it is achieved will be exposed. To do so, a list of possible protocols from the simplest one to the more complex will explained. A ranking of the prefered working procedures will be done. All the images have been extracted from [1].

1.2.2.1 Simplest Protocol

This protocol has no error or flow control. It is supposed that the frames are traveling only in one direction, from the sender to the receiver. It is also supposed that the receiver can immediately handle the frames received, so there is no overwhelming. The DLL of the sender site gets data from its network layer, makes a frame out of the data and sends it. The DLL at the receiver site receives a frame from its physical layer, extracts data from the frame and delivers the data to its network layer. The problem here is that the sender site cannot send a frame until its network layer has a data packet to send and the receiver site cannot deliver a data packet to its network layer until a frame arrives. There is the need to introduce the idea of events in the protocol. The procedure at the sender site is constantly running; there is no action until there is a request from the network layer. The procedure at the receiver site is also constantly running, but there is no action until notification from the physical layer arrives.

```
1  while (true)                    // Repeat forever
2  {
3    WaitForEvent()i               // Sleep until an event occurs
4    if(Event(RequestToSend»       //There is a packet to send
5    {
6        GetData()i
7        MakeFrame()i
8        SendFrame()i              //Send the frame
9    }
10 }
```

Figure 1.2.1: Sender algorithm for the simplest protocol.

```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent()i                    II Sleep until an event occurs
4    if(Event(ArrivalNotification»      IIData frame arrived
5    {
6        ReceiveFrame()i
7        ExtractData()i
8        DeliverData ()i                //Deliver data to network layez
9    }
10 }
```

Figure 1.2.2: Receiver algorithm for the simplest protocol.

1.2.2.2 Stop-and-Wait Protocol

If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames,we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender. In the Stop-and-Wait Protocol the sends one frame, stops until it receives confirmation from the receiver and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to our previous protocol. In this case the algorithms of the sender and the receiver are the following ones.

```
1  while(true)                          IIRepeat forever
2  canSend = true                       IIAllow the first frame to go
3  {
4    WaitForEvent()i                    II Sleep until an event occurs
5    if(Event(RequestToSend) AND canSend}
6    {
7       GetData()i
8       MakeFrame()r
9       SendFrame()i                    I/Send the data frame
10      canSend = false;                I/cannot send until ACK arrives
11   }
12   WaitForEvent()i                    II Sleep until an event occurs
13   if(Event(ArrivalNotification)      /I An ACK has arrived
14    {
15      ReceiveFrame();                 I/Receive the ACK frame
16      canSend = true;
17    }
18 }
```

Figure 1.2.3: Sender algorithm for the Stop-and-Wait Protocol.

```
1  while (true)                         IIRepeat forever
2  {
3    WaitForEvent();                    II Sleep until an event occurs
4    if(Event(ArrivalNotification)}     IIData frame arrives
5    {
6       ReceiveFrame(};
7       ExtractData(}i
8       Deliver(data};                  /IDeliver data to network layex
9       SendFrame();                    IISend an ACK frame
10   }
11 }
```

Figure 1.2.4: Receiver algorithm for the Stop-and-Wait Protocol.

The two protocols explained are protocols that can be suitable for noiseless channels. However, noiseless channels are nonexistent. There is a need to add error control to the protocol. Three protocols are discussed with the aim of doing so.

1.2.2.3 Stop-and-Wait Automatic Repeat Request

The Stop-and Wait ARQ adds a simple error control mechanism to the Stop-and-Wait Protocol. To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently

discarded. The detection of errors in this protocol is manifested by the silence of the receiver. Frames are also numbered so if the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated. What is done to solve the error is that when the sender sends a frame, it keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network. Since an ACK frame can also be corrupted and lost, it too needs redundancy bits and a sequence number. In the following figure is possible to see more clearly what is going on with this protocol.

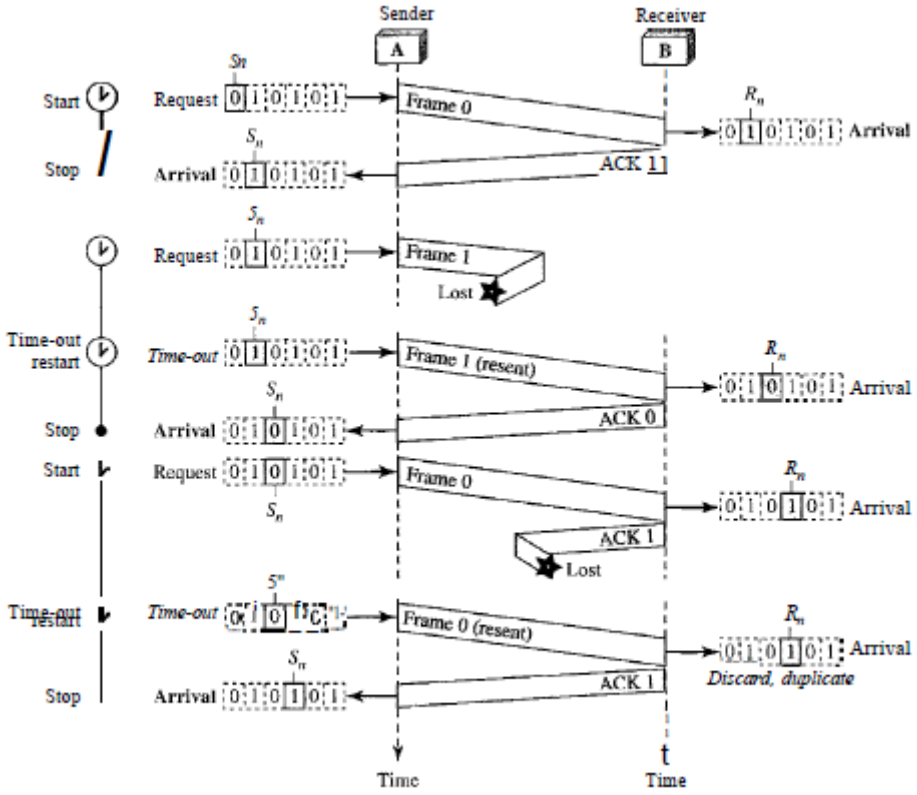


Figure 1.2.5: Flow diagram of the Stop-and Wait ARQ.

The main problem of this protocol is its efficiency. The Stop-and-Wait ARQ is very inneficinet if our channel is thick and long. The product of thickness and longitud is called the bandwidth-delay product. We can think of the channel as a pipe. The bandwidth-delay product then is the volume of the pipe in bits. The pipe is always there. If we do not use it, we are inefficient.

1.2.2.4 Go-Back-N Automatic Repeat Request

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment. In the Go-Back-N Automatic Repeat Request the sender sends several frames before receiving acknowledgments. It also keeps a copy of these frames until the acknowledgments arrive. Although there can be a timer for each frame that is sent, in this protocol only one is used. The reason is that the timer for the first outstanding frame always expires first and then all outstanding frames when this timer expires are sent again. The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame at the sender site to expire. This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames. That is the reason why the protocol is called Go-Back-N. The flow diagram and the and algorithms of the sender and the receiver are shown next.

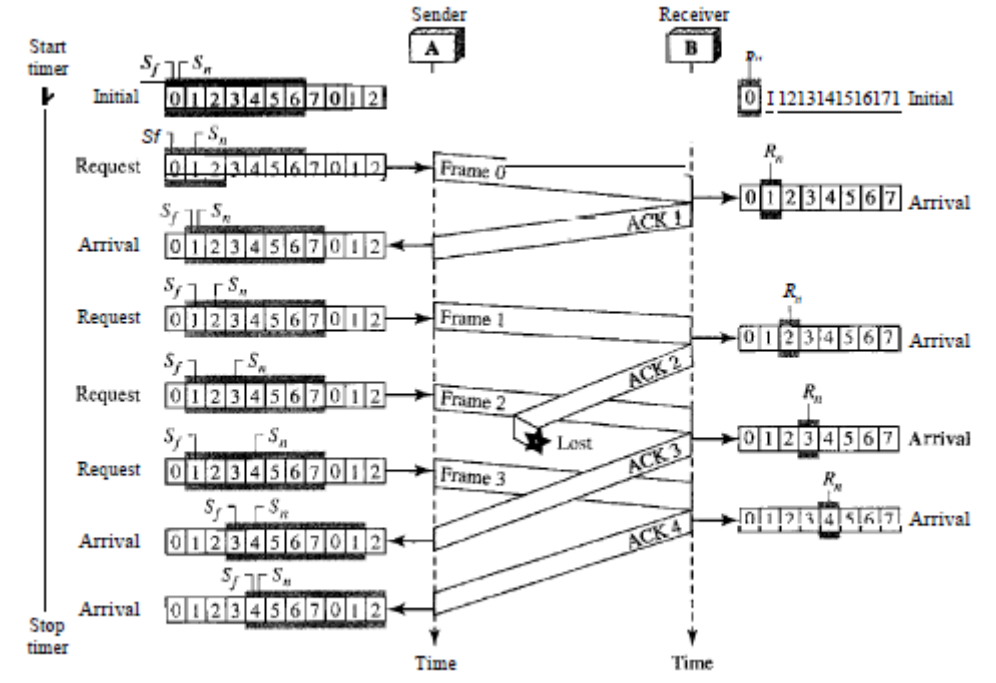


Figure 1.2.6: Flow diagram of the Go-Back-N ARQ.

```
1  Rn = 0;
2
3  while (true)                          //Repeat forever
4  {
5    WaitForEvent();
6
7    if(Event{ArrivalNotification»       /Data frame arrives
8    (
9        Receive(Frame);
10       if(corrupted(Frame»
11            Sleep();
12       if(seqNo == Rn)                 //If expected frame
13       {
14         DeliverData()i                //Deliver data
15         Rn =Rn + 1;                   //Slide window
16         SendACK(Rn);
17       }
18   }
19 }
```

Figure 1.2.7: Receiver algorithm for the Go-Back-N ARQ.

```
1  Sw = 2** - 1;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7   WaitForEvent();
8    if(Event(RequestToSend)              //A packet to send
9    {
10      if(Sn-Sf >= Sw)                  //If window is full
11            Sleep();
12      GetData();
13      MakeFrame(Sn);
14      StoreFrame(Sn);
15      SendFrame(Sn);
16      Sn = Sn + 1;
17      if(timer not running)
18            StartTimer();
19   }
20
21   if(Event(ArrivalNotification)       //ACK arrives
22   {
23      Receive(ACK);
24      if(corrupted(ACK)
25            Sleep();
26      if((ackNo>sf)&&(ackNO<=Sn)       //If a valid ACK
27      While(Sf <= ackNo)
28       {
29         PurgeFrame(Sf);
30         Sf = Sf + 1;
31       }
32       StopTimer();
33   }
34
35   if(Event(TimeOut)                   //The timer expires
36   {
37    StartTimer();
38    Temp = Sf;
39    while(Temp < Sn);
40     {
41      SendFrame(Sf);
42      Sf = Sf + 1;
43     }
44   }
45 }
```

Figure 1.2.8: Sender algorithm for the Go-Back-N ARQ.

1.2.2.5 Selective Repeat Automatic Repeat Request

Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. In the case of these protocol, the Selective Repeat ARQ, the processing at the receiver is more complex but is more efficient for noisy links. The Selective Repeat Protocol allows a number of frames to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. The handling of the request event is similar to that of the previous protocol except that one timer is started for each frame sent. The arrival event is more complicated here. An ACK or a NAK frame may arrive. If a valid NAK frame arrives, the corresponding frame is resent. If a valid ACK arrives the corresponding timer stops. When the time for a frame has expire, only

this frame is resent.

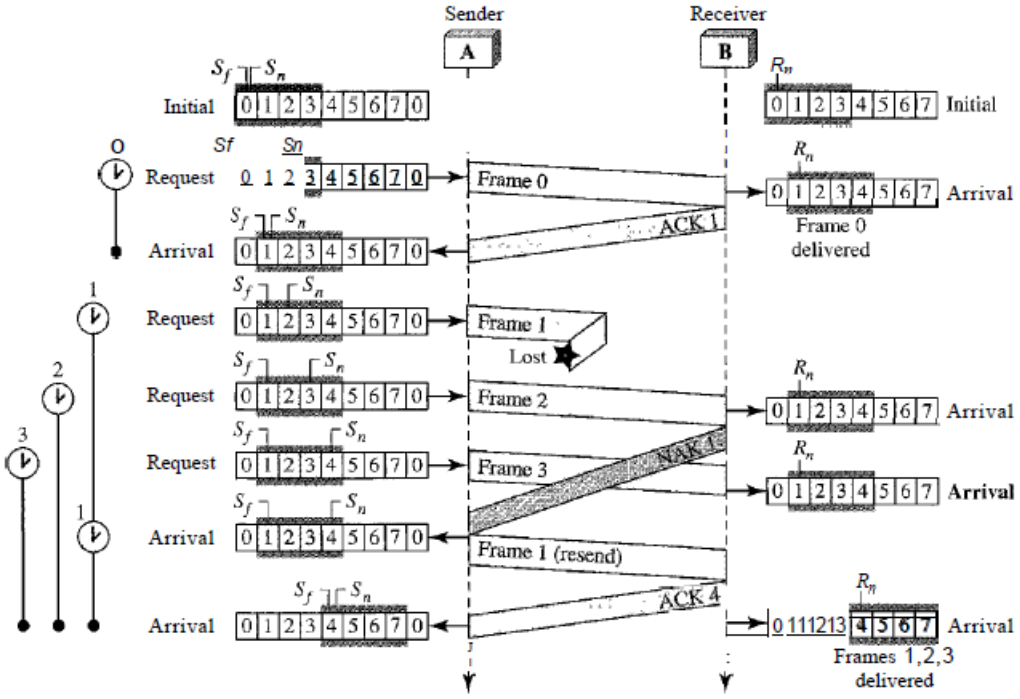


Figure 1.2.9: Flow diagram of the Selective Repeat ARQ.

```
1     = 2m-1 i
2     = Oi
3     = Oi
4
5   hile (true)                    //Repeat forever
6   {
7     WaitForEvent()i
8     if(Event(RequestToSend)}       //There is a packet to sen
9     {
```

```
10      if(Sn-S;E >= Sw)                    I/If window is full
11            Sleep();
12      GetData();
13      MakeFrame(Sn);
14      StoreFrame(Sn);
15      SendFrame(Sn);
16      Sn = Sn + 1;
17      StartTimer(Sn);
18    }
19
20    if(Event(ArrivalNotification»   IIACK arrives
21    {
22      Receive(frame);                   I/Receive ACK or NAK
23      if(corrupted(frame»
24            Sleep();
25      if (FrameType == NAK)
26         if (nakNo between Sf and So)
27         {
28          resend(nakNo);
29          StartTimer(nakNo);
30         }
31      if (FrameType = ACK)
32         if (ackNo between Sf and So)
33         {
34           while(sf < ackNo)
35           {
36            Purge(sf);
37            stopTimer(Sf);
38            Sf = Sf + 1;
39           }
40         }
41    }
42
43    if(Event(TimeOut(t»)              liThe timer expires
44    {
45     StartTimer(t);
46     SendFrame(t);
47    }
48  }
```

Figure 1.2.10: Sender algorithm for the Selective Repeat ARQ.

```
1  Rn = 0;
2  NakSent = false;
3  AckNeeded = false;
4  Repeat(for all slots)
5      Marked(slot) = false;
6
7  !while (true)                              IIRepeat forever
8  {
9    WaitForEvent()i
10
11   if{Event{ArrivalNotification»            jData frame arrives
12   {
13      Receive(Frame);
14      if(corrupted(Frame»&& (NOT NakSent)
15      {
16       SendNAK(Rn);
17       NakSent = true;
18       Sleep{};
19      }
20      if(seqNo <> Rn)&& (NOT NakSent)
21      {
22       SendNAK(Rn);
23       NakSent = true;
24       if {(seqNo in window)&&(IMarked(seqNo»
25       {
26        StoreFrame{seqNo)
27        Marked(seqNo)= true;
28        while(Marked(Rn)
29        {
30         DeliverData(Rn);
31         Purge(Rn)r
32         Rn = Rn + 1;
33         AckNeeded = true;
34        }
35         if(AckNeeded);
36         {
37         SendAck(Rn);
38         AckNeeded = false;
39         NakSent = false;
40         }
41       }
42      }
43   }
44 }
```

Figure 1.2.11: Receiver algorithm for the Selective Repeat ARQ.

1.2.2.6 Bidirecional links: Piggybacking

Piggybacking is not a protocol, is a technique. All que protocols explained until now are all unidirectional: data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction. In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions. Piggybacking is usedto improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can

also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

1.2.2.7 Working procedure ranking

Now its time to choose the working procedure that best fits the needs of the mission. To do so, an OWA (Ordered Weighted Average) will be used. The criteria to consider is the following one:

- Efficiency: This fact deals with how the channel is being used. Protocols will be classified as non-efficient or efficient.
- Time: This fact deals about the time needed to transmit the data satisfactory.
- Error correction: Deals about whether a protocol can correct an error of transmission or not.

It is important also to take into account that the protocol to use should have a flow control, that is, should know if the receiver is available or not to receive the data. For this reason the Simplest Protocol is rejected and won't be studied in the OWA. Regarding the factors of the OWA, all of them will be rated from 0 to 1. In this project the fact of transmitting the data without errors is more important than transmitting it fast, as is possible to appreciate un the project charter (the latency can be relative high, but incorrect information is useless). The efficiency of the protocol is very important too, because the less the efficiency the less power provided by the CubeSat is being used. Since the CubeSat has limited space, ideally al the power it can gives for transmittion will be used for it. Then, the weights of the different factors are the following ones:

- Efficiency: 40
- Time: 30
- Error correction: 60

In the following table the rating of each protocol together with the corresponding OWA is shown.

Protocol	Efficiency	Time	Error correction	OWA
Stop-and-Wait Protocol	0	0	0	0
Stop-and-Wait ARQ	0	0	1	0,46
Go-Back-N ARQ	1	0	1	0.69
Selective Repeat ARQ	1	1	1	1

Table 1.2.1: OWA of the DLL protocols.

Then, the ranking of working procedures is the following one:

1	Selective Repeat ARQ
2	Go-Back-N ARQ
3	Stop-and-Wait ARQ
4	Stop-and-Wait Protocol

Table 1.2.2: Ranking of working procedures

It has to be said that when dealing with bidirecional links piggybacking technique will be used if possible.

1.2.3 Protocols

The standards of the CCSDS will be followed in order to allow interoperability with other satellites such as the one of the client. The CCSDS has developed four protocols for the Data Link Protocol Sublayer of the Data Link Layer [2]:

- TM Space Data Link Protocol
- TC Space Data Link Protocol
- AOS Space Data Link Protocol
- Proximity-1 Space Link Protocol-Data Link Layer

These protocols provide the capability to send data over a single space link. TM, TC, and AOS can have secured user data into a frame using the Space Data Link Security (SDLS) Protocol.

CCSDS has also developed three standards for the Synchronization and Channel Coding Sublayer of the DLL:

- TM Synchronization and Channel Coding
- TC Synchronization and Channel Coding
- Proximity-1 Space Link Protocol—Coding and Synchronization Layer

TM Synchronization and Channel Coding is used with the TM or AOS Space Data Link Protocol, TC Synchronization and Channel Coding is used with the TC Space Data Link Protocol and the Proximity-1 Space Link Protocol—Coding and Synchronization Layer is used with the Proximity-1 Space Link Protocol—Data Link Layer. This can be seen better in the following image.

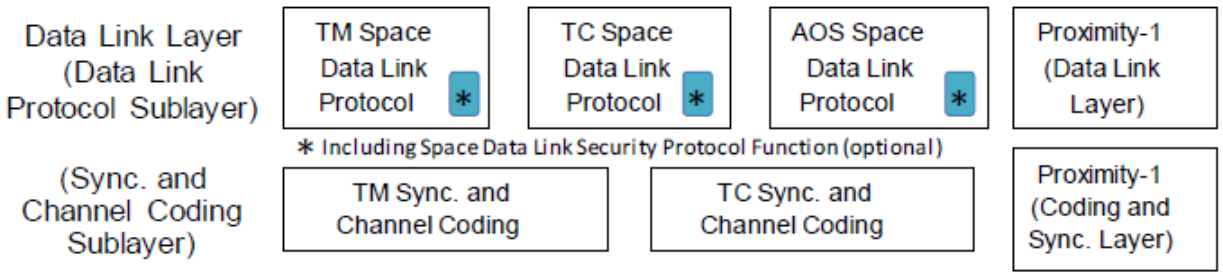


Figure 1.2.12: DLL of the CCSDS.

Now the reliability of each of the protocols of the Data Link Protocol Sublayer will be compared in order to know which one is the best of them. This will be done because reliability is the most important feature of the DLL.

Protocol	System used for reliability
TM	Stop-and-Wait Protocol
TC	Type-A: Go-Back-N ARQ, Type-B:Stop-and-Wait Protocol
AOS	Stop-and-Wait Protocol
Proximity-1	Go-Back-N ARQ

Table 1.2.3: Reliability of CCSDS protocols

According to the table and to the ranking of working procedures done previously, only TC Type-A and Proximity-1 will be considered from now on. Security is another important feature to take into account when taking this decision. TM Space Data Link Protocol has provision for inserting secured data into a frame using the Space Data Link Security (SDLS) Protocol. However, there have been no security requirements to date established for Proximity-1. The SLDS protocol can provide security services, such as authentication and confidentiality for TC Transfer Frames (it can also do it with TM and AOS, that have been previously discharted). Both the TC and the Proximity-1 use variable-lenght Transfer Frames to facilitate reception of short messages with short delay. Another key feature to take into account when deciding a protocol, is the concept of "Virtual Channels". The Virtual Channel facility allows one Physical Channel (a stream of bits transferred over a space link in a single direction) to be shared among multiple higher-layer data streams, each of which may have different service requirements. A single Physical Channel may therefore be divided into several separate logical data channels, each known as a Virtual Channel (VC). The TC has the following identifiers: the Transfer Frame Version Number (TFVN), the Spacecraft Identifier (SCID), and the Virtual Channel Identifier (VCID). It also uses an optional identifier, called the Multiplexer Access Point Identifier (MAP ID), that is used to create multiple streams of data within a Virtual Channel. In contrast, the Proximity-1 uses a triad of multiplexing capabilities, which is incorporated for specific funcionality within the link. The Spacecraft Identifier (SCID) identifies the source or destination of Transfer Frames transported in the link connection based upon the Source-or-Destination Identifier. The Pysical Channel Identifier (PCID) provides up to two independently multiplexed channels. The Port ID provides the means to route user data internally to specific logic ports, such as applications or transport processes, or to physical ports, such as onboard buses or physical connections. Now a table with the identifiers of the TC and the Proximity-1 will be shown:

Identifiers	TC Space Data Link Protocol	Proximity-1 Space Link Protocol- Data Link Layer
TFVN	00	10
SCID	0 to 1023	0 to 2013
PCID	N/A	0 to 1
VCID	0 to 63	N/A
MAP ID	0 to 63	N/A
Port identifier	N/A	0 to 7

Table 1.2.4: Identifiers of TC and Proximity-1 Space Data Link Layer Protocols

Having Virtual Channels is important for the mission that is exposed in this project because it

allows having more than one stream of bits to take place at the same time, that is to say that more than one client can communicate with their satellite without having to wait for another client to finish.
The decision taken is to use the TC Space Data Link Protocol with the TC sync. and channel coding together with the Space Data Link Security Protocol. The reasons for doing so are mainly:

- Security: Incorporing the SLDS authentication and confidentality is provided.
- More virtual channels: This feature allow more clients communicating with their satellites at the same time.

1.2.4 TC Space Data Link Protocol

Now some specifications of the chosen protocol will be exposed in order to know how it is structured and how many bits it adds to the original data. Further information of the protocol can be found in [3]. The protocol specifications will be explained when it is used with the support of the SDLS protocol. In this section is important to know that 1 octet is an eight-bit word. The structure of the transfer frame in this protocol is the following one:

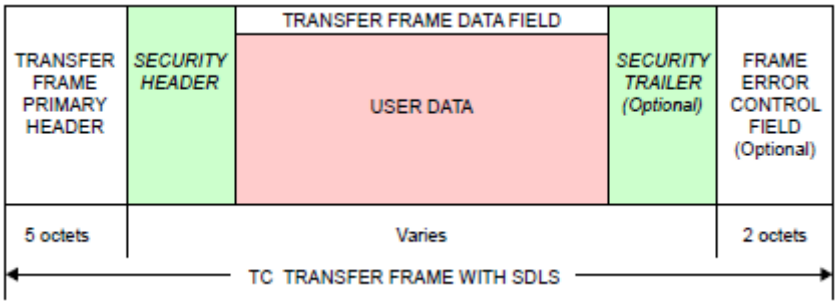


Figure 1.2.13: Transfer frame structure of the TC Space DL Protocol with SDLS.

In the transfer frame primary header, the following information is contained:

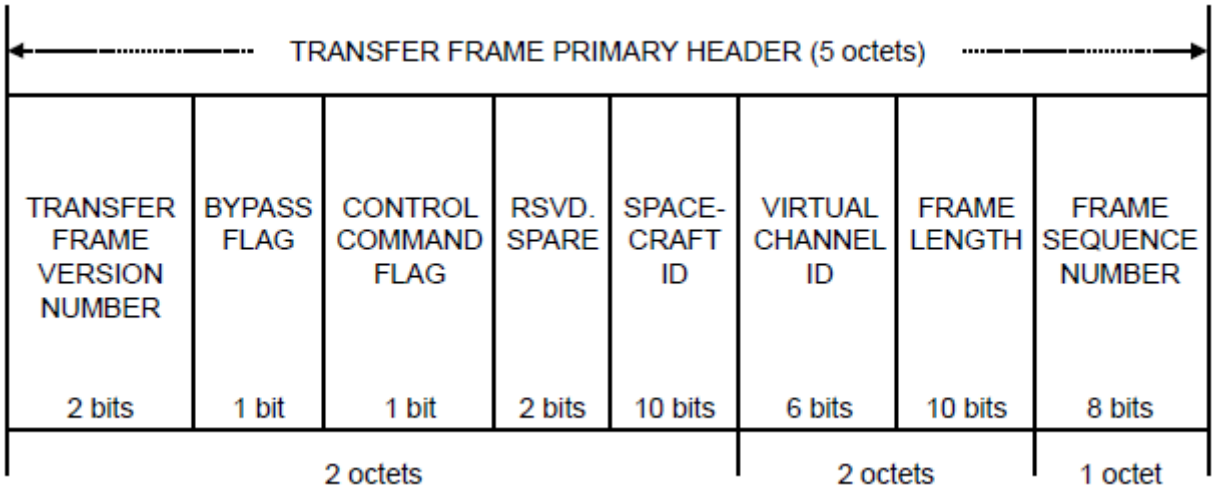


Figure 1.2.14: Transfer frame primary header.

With this data, is possible to say that the TC Space Data Link Protocol will add to data coming from the Network layer at least 5 octets (40 bits).

1.2.5 TC Sync and Channel Coding

This protocol is the corresponding to the Synchronization and Channel Coding Sublayer that has be used with the TC Space and Data Link Protocol. It has functions as for example, encapsuate the data units so that the start and end can be detected by the receiving end, ensure there are sufficient bit transitions in the transmitted bit stream so that the receiver can mantain bit synchronization during the reception of the data unit, etc. In a nutshell, one instance of the Synchronization and Channel Coding Sublayer processes the data stream for a single Physical Channel, making it a stream of bits that can be transferred over a space link in a single direction. The procedures can be differenciated between the ones that occur in the sending end and the one that occur in the receiving end. The procedures are the following ones:

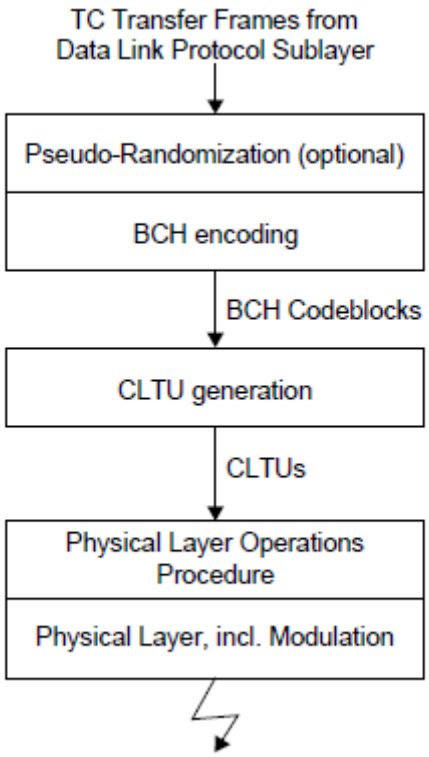


Figure 1.2.15: Procedure at the sending end.

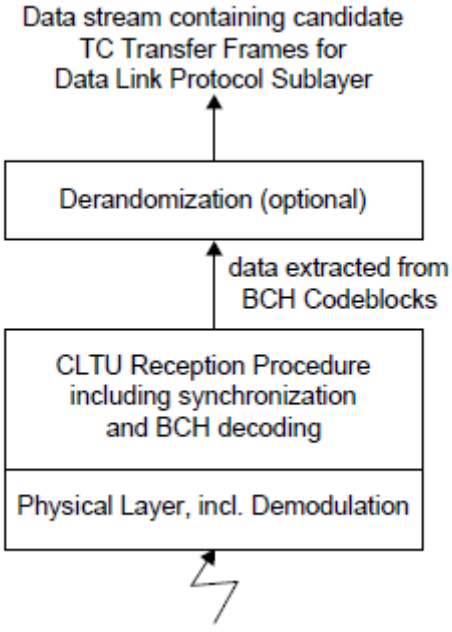


Figure 1.2.16: Procedure at the receiving end.

Is possible to see that two packets of data are created, BCH Codeblocks and CLTUs. From the point of view of the Synchronization and Channel Coding Sublayer, the content of the Frames parameter is a single block of data. For a single Channel Access request, the Synchronization and Channel Coding Sublayer generates a set of BCH Codeblocks, and that set of BCH Codeblocks is placed in a single CLTU. One of the managed parameters for the Physical Channel is the maximum length of a CLTU. The lenght of the CLTU can be calculated as follows (in octets):

$$Lenghtof the CLTU = 10 + 8 \cdot (\frac{Totallenghtof the frames + 6}{7}) \tag{1.2.1}$$

Since with the TC Space Data Link protocol the frames can have different sizes, the CLTU can also have different sizes. More information about this sublayer of the DLL can be found in reference [4]

1.3 Layer 3: The Network

1.3.1 Functions of the Network Layer

According to [5], the Network layer is the third layer in the Open Systems Interconnection (OSI) model. It is located above the Data link layer and below the Transport layer. This layer is used for transmitting data sequences called datagrams between a sender and a reciever than may not be directly connected through only one link. The Network layer provides the following functions:

- **Routing**: Selects the best path between two nodes in a network, often using itermediate nodes called routers.
- **Network fow control**: Routers may indicate a transmitting node to reduce its transmission when the router's buffer becomes full.
- **Package fragmentation**: If the message to be transmitted is too large to be transmitted in the Data link layer, the network may split it into several packages in one node, send them independently an reassemble them in another node. Optionally, it can provide error control.
- **Logical-physical adress allocation**: Translates the logical adress (or names) of the network nodes into a unique physical adress.
- **Message forwarding**: A network may be divided into subnetworks, connected through specialized hosts, called gateways or routers, that forward packets between those subnetworks.

1.3.2 Protocols

The Consultive Comitee for Space Data Systems (CCSDS) [6] has two standards for using in the Network layer in conjuction with the Space Data Link Layer Protocols recommended by the CCSDS. Those two standards are the Space Packet Protocol (SPP) [7] and the Encapsulation Service [8]. With the Space Packet Protocol, application processes generate and consume Protocol Data Units (PDU). The Encapsulation Service encapsulates PDU of recognized protocols defined in a Space Assigned Number Authority (SANA) [9] registry into two types

of packets, either Space Packets or Encapsulation Packets. External protocols data units, such as the Internet Protocol datagrams, can be transmitted by CCSDS Space Data Link Protocols, although they cannot be directly encapsulated by the Encapsulation Service, and an intermediate service, such as IP over CCSDS (IPoC) [10], must be used.

Figure 1.3.1, shows the recommended protocols by the CCSDS for Space Communications. In Figure 1.3.2 those protocols arearranged in some possible combinations. As it can be seen, IP cannot be directly used neither by the protocols in the Data Link layer nor the Encapsulation Service.

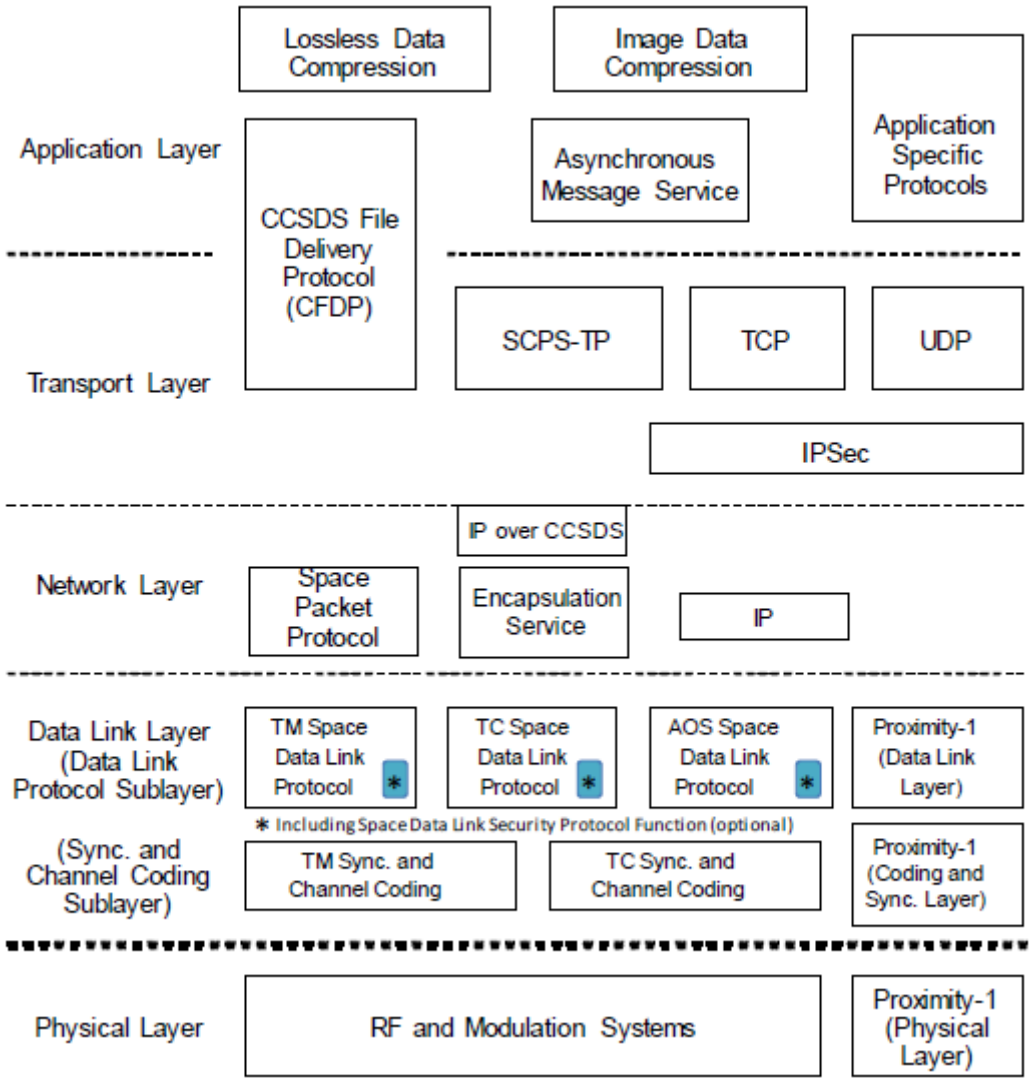


Figure 1.3.1: Protocols recommended by the CCSDS, classified in their respective OSI layers. Extracted from [6].

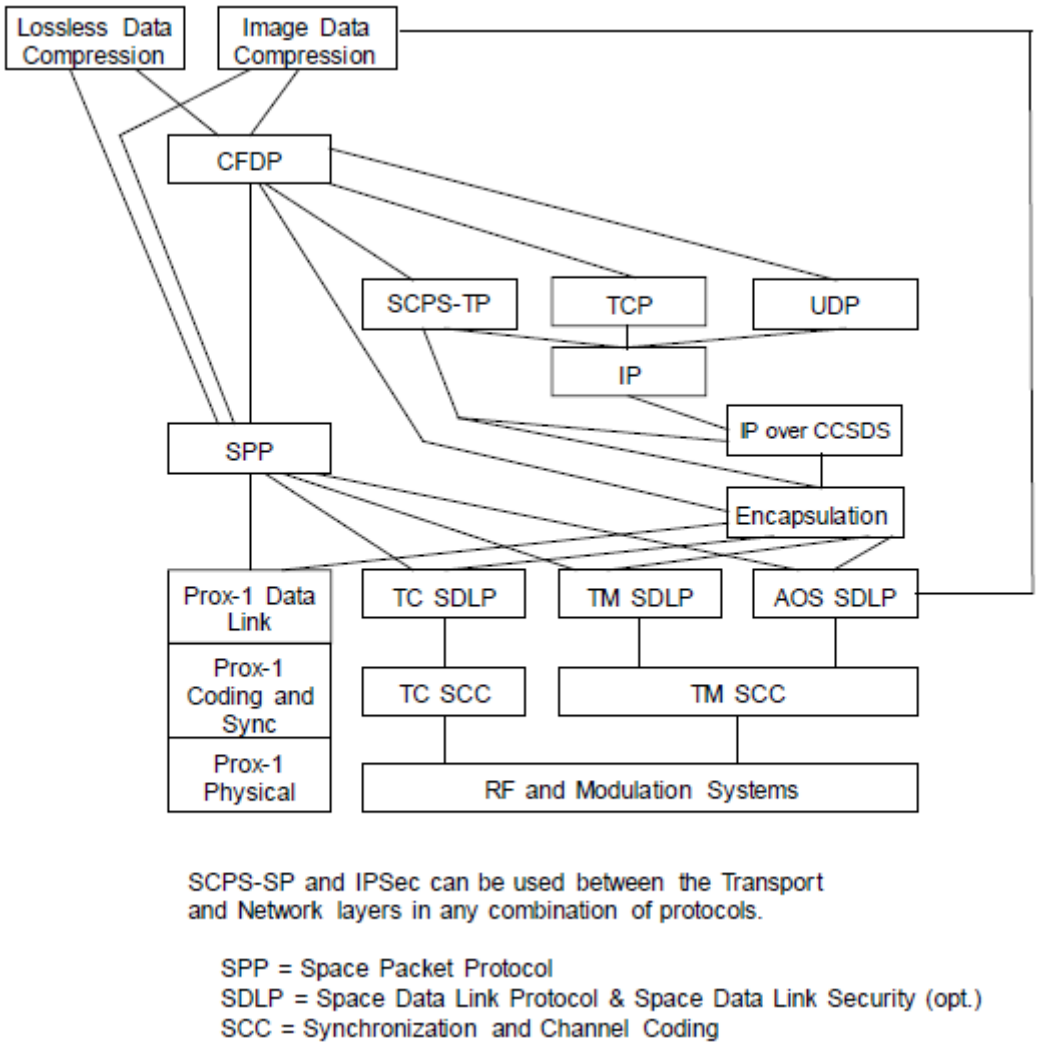


Figure 1.3.2: Possible Combinations of the CCSDS recommended protocols. Extracted from [6].

Protocols in the Network Layer can be classified according if they are the main protocol (SPP or IP, for example) or they provide additional features so that the main protocol can work efficiently. An example of the latter are routing protocols, and also for IP, IPoC and Encapsulation Service.

In the following pages, a brief review of distinct protocols on the Network layer will take place. Since CCSDS ecommends using SPP or Encapsulation Service, only SPP and protocols that can be encapsulated by the Encapsulation Service, eiyher directly or indirecly, will be reviewed. The protocols reviewed will be classified according if they are the main protocol,

auxiliary protocols, or routing protocols.

1.3.2.1 Main protocols

Space Packet Protocol (SPP) [7]

The Space Packet Protocol (SPP) is a protocol designed to efficiently transfer application data over a network of space links. SPP provides a unidirectional data transfer service from a single source user application to one or more destination user applications through one or more subnetworks. The path from the source user application to the destination user application is called a Logical Data Path (LDP). Every LDP is uniquely identified by a Path Identifier (Path ID). The protocol data unit used by this protocol is the Space Packet. Each Space Packet is defined by a header section and a data section.

Each LPD is uniquely identified by a Path ID. A Path ID consists of an Application Process Identifier (APID) and an optional APID Qualifier. APID Qualifiers identify the naming domain for an APID. APIDs are unique in a single naming domain. The APID is part of the header of the Space Packet, but the APID Qualifier must be carried by a protocol of an underlying layer.

The following features are common to the services of the SPP:

- Pre-configured Services. The user can send or receive data only through a preconfigured LDP established by management.
- Unidirectional Services. One end of an LDP can send, but not receive, data through the LDP, while the other end can receive, but not send. This means A can send to B through a LPD, but for B to send to A has to use a different LDP
- Asynchronous Services. There are no predefined timing rules for the transfer of service data units supplied by the service user. The user may request data transfer at any time it desires, but there may be restrictions imposed by the provider on the data generation rate.
- Unconfirmed Services. The sending user does not receive confirmation from the receiving end that data has been received.

- Incomplete Services. The services do not guarantee completeness, nor do they provide a retransmission mechanism.
- Non–sequence Preserving Services. The sequence of service data units supplied by the sending user may not be preserved through the LDP.

The following services are assumed from the underlying layers:

- Adressing and routing capabilities for establishing LDPs
- Capability for associating an APID Qualifier for each Space Packet.

The structure of a Space Packet consists of a Packet Primary Header, and a Packet Data Field, which can contain an optional Secondary Header. Figure 1.3.3 shows the structure of the SPP primary header:

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th></tr>
<tr><td>0</td><td>0</td><td colspan="3">Packet Version Number</td><td>Packet Type</td><td>Secondary Header Flag</td><td colspan="11">Application Process Identifier (APID)</td></tr>
<tr><td>4</td><td>32</td><td colspan="2">Sequence Flags</td><td colspan="14">Packet Sequence Count or Packet Name</td></tr>
<tr><td>8</td><td>64</td><td colspan="16">Packet Data Length</td></tr>
</table>

Figure 1.3.3: Example of a header for an SPP Space Packet.

Internet Protocol version 4 (IPv4) [11]

The Internet Protocol version 4 (IPv4) is the fourth version of the Internet Protocol (IP). It is one of the core protocols of standards-based internetworking methods in the Internet. Despite the ongoing deployment of a successor protocol (IPv6), the IPv4 still routes most of the Internet traffic. IPv4 is a conectionless protocol and does not guarantee delivery, nor does it assure proper sequencing or avoidance of duplicate delivery. These aspects are adressed by a transport layer protocol.

One of the features of IPv4 are adresses. Network adresses are the identification number of any device that is part of a network. IPv4 uses 32-bit (4 byte) adresses. Therefore, the adress

space is limited to 4294967296 (2^{32}) adresses. A IPv4 adress is usually represented in two ways: in binary notation, where each group of 8 bits is separated by a dot, or in decimal notation, where each 8-bit binary number is translated to decimal, as it can be seen in Table 1.3.1.

IP adress	10101100000100001111111000000001
Dot-binary notation	10101100.00010000.11111110.00000001
Dot-decimal notation	172.16.254.1

Table 1.3.1: IP adress notation in dot-decimal and dot-binary.

Packets in the IPv4 consist of a header section and a data section. There is no footer at the end of the data section since the protocols in the data link layer and the transport layer provide error correction controls. Headers in a IPv4 packet contain 14 fields, one of them being optional. The fields are packed with the most significant byte first, and the most significant bit is also the first. Headers have a lenght between 20 and 60 bytes. The data section comes after the header, and its format depends on the protocol used (for example, ICMP, IGMP, TCP, etc.). FIgure 1.3.4 shows the structure of a IPv4 header.

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total Lenght</td></tr>
<tr><td>4</td><td>32</td><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><td>8</td><td>64</td><td colspan="8">Time to Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><td>12</td><td>96</td><td colspan="32">Source IP Adress</td></tr>
<tr><td>16</td><td>128</td><td colspan="32">Destination IP Adress</td></tr>
<tr><td>20</td><td>160</td><td colspan="32" rowspan="4">Options (if IHL>5)</td></tr>
<tr><td>24</td><td>192</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
</table>

Figure 1.3.4: Example of a header for an IPv4 packet. In this case, it has a lenght of 36 bytes.

IPv4 provides fragmentation of packets. If size of the packet is bigger than the maximum transmission unit (MTU) of the destination, and the message allow fragmentation (the option of Do not Fragment in the header of the packet is set to 0) the transmitting router will divide the packet in fragments smaller than the MTU.

Internet Protocol version 6 (IPv6) [12]

The Internet Protocol version 6 (IPv6) is the most recent version of the Internet Protocol, developed to solve the problem of the exhaustion of IP adresses of the IPv4. IPv6 is intended

to replace IPv4. The new features of the IPv6 compared of those of the IPv4 are the following:

- Larger adress space: The length of IPv6 adresses is 128 bits, which is four times the length of IPv4 adresses. It offers a capacity of 2^{128} adresses.
- Multicasting: IPv6 acomplishes multicasting without using other protocols (such as IGMP for IPv4)
- Stateless address autoconfiguration (SLAAC): IPv6 hosts can configure themselves automatically when they are connected to a IPv6 network using the Neighbor Discovery Protocol via Internet Control Message Protocol version 6 (ICMPv6) router discovery messages. When a host is connectes for the first time, it sends a link-local router solicitation multicast request for its configuration parameters. Then, routers respond to the request with a router advertisement packet that contains Internet Layer configuration parameters.
- Network-layer security: Internet Protocol Security was developed for IPv6 before it was adapted for IPv4.
- Simplified processing by routers: Packet headers and the process of packet forwarding have been simplified, so packet processing by routers is more efficient. Headers now have a fixed length of 40 bytes, and may have an optional section aimed for options between the header section and the data section. Figure 1.3.5 shows the structure of a IPv6 header. IPv6 routers do not perform fragmentation.
- Mobility: Mobile IPv6 avoids triangular routing (unlike IPv4) and is as efficient as native IPv6.
- Options extensibility: IPv6 headers have astructure capable of extending the protocol in the future without affecting the core packet structure.
- Jumbograms: IPv4 limits packets to (2 power 16) - 1 octets per payload. A IPv6 node can handle packets of (2 power 32) - 1 octets (called jumbograms).

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				Traffic Class								Flow Label																			
4	32	Payload Length																Next Header								Hop Limit							
8	64	Source Adress																															
12	96																																
16	128																																
20	160																																
24	192	Destination Adress																															
28	224																																
32	256																																
36	288																																

Figure 1.3.5: Example of a header for an IPv6 packet.

1.3.2.2 Auxiliary protocols

Encapsulation service [8]

The Encapsulation Service is a service used to transfer data units that can not be directly transferred by the CCSDS Space Data Link Protocols. In order to be directly transferred by a Space Data Link Protocol, a data unit must have a Packet Version Number authorized by the CCSDS (a list of PVN authorized by CCSDS is contained in [13]). With the Encapsulation Service, data units that do not have an authorized VPN can be transmitted with Space Data Link Protocols. The data unit to be transmitted must be of an integral number of octets.

A user of the Encapsulation Service is identified by the combination of the following:

- A Packet Version Number (PVN) that indicates whether Space Packets (PVN=1) or Encapsulation Packets (PVN=8) are used for encapsulation,
- An Encapsulated Protocol Identifier (EPI), whichis either:
  - An Application Process Identifier (APID) defined in reference (if Space Packets are used).
  - a Protocol ID defined in section 4 of this document (if Encapsulation Packets are used).

The APIDs used by the Encapsulation Service must be registered as ‘reserved APIDs’ in [14]. The Protocol IDs used by the Encapsulation Service must be registered as ‘defined Protocol IDs’ in [15].

If the Data Unit is encapsulated in a Space Packet, the header format of the Space Packet is the same as the one used by Space Packet Protocol, only that the values of the parameters are restricted to some values. On the other hand, if the Data Unit is encapsulated in a Encapsulation Packet, a different header format will be used. This header have a length of 1-8 octets, and for the case of 8 octet it can be shown in Figure 1.3.6

	Bit							
Octet	0	1	2	3	4	5	6	7
0	Packet VersionNumber			Protocol ID			Length of length	
1	User defined fields				Protocol ID extension			
2	CCSDS defined field							
3								
4	Packet length							
5								
6								
7								

Figure 1.3.6: Example of a header for an Encapsulation Packet of maximum length. Some parameters may vary its length in other cases.

IP over CCSDS (IPoC) [10]

The IP over CCSDS is used to transfer IP Data Units over CCSDS Space Data Link Protocols. IP Data Units are encapsulated in Encapsulation Packets and sent through Space Data Link Protocols. IPoC uses the CCSDS Internet Protocol Extension (IPE) convention in conjunction with the CCSDS Encapsulation Service. The IPE convention is used to add IPE octecs at the beginning of a IP Data Unit, encapsulate the result in an Encapsulation Packet, and transmit it with a CCSDS Space Data Link Protocol. It is used because not all protocols that use an IP datagram have a Protocl ID used by the Encapsulation Packet.

IPoC adds a header at the beginning of the IP Data Unit, called IPE header. The sum of the IP Data Unit and the IPE header is the Data Unit used by the Encapsulation Service. In other words, for the Encapsulation Service, the IPE header and the IP Data Unit are a whole.

The structure of the IPE header will be the following. It must be of a length of an integral number of octets, with a minumum length of 1 octet. Each octet will be divided into two parts: the first seven bits (bits 0-6), and the least significant bit (LSB, bit 7). If more octets are

added, the LSB of all octets except the last octet are set to '0'. The value of the IPE header is the decimal value of all the octets. The value of the IPE header must be one of the possible values in [16].

Internet Control Message Protocol (ICMP) [17]

The Internet Control Message Protocol (ICMP) is one of the main protocols of the TCP/IP protocol suite. It is used to send error messages to the source IP of the data packet. It is assigned IP protocol number 1. ICMP messages are typically used for diagnostic, control purposes or generated in response to errors in IP operations. They are processed differently than normal IP proceessing.

There are many types of control messages that the ICMP can send:

- Souce quench: Used to request the sender to decrease the rate of messages sent to a router.
- Redirect: Used to request the sender to send the data to another router.
- Time exceeded: Used by a gateway to inform the sender of a discarded datagram due to the time to life field reaching zero. It is also used to inform the sender that a fragment of a message has not been reassembled within the time limit
- Timestamp: Used for time syncronization. The sender sends the timestamp it last touched the packet (in miliseconds since midnight)
- Timestamp reply: Used to reply a timestamp. The reciever of the timestamp message replies the sender with the original timestamp, the timestamp when the message was recieved, and the timestamp when the reply was sent.
- Adress mask request: Used by a host to obtain the subnet mask of a router
- Adress mask reply: Used to reply the adress mass request returning the subnet mask.
- Destination unreachable: Used by the host or its inbound gateway to inform the client that the destination is unreachable.

Internet Control Message Protocol version 6 (ICMPv6) [18]

The Internet Control Message Protocol version 6 (ICMPv6) is the implementation of the ICMP for IPv6. Several extensions have been published that define new types of ICMPv6 messages, as well as new options for existing message types. One of those is the Neighbor Discovery Protocol (NDP), a node discovery protocol for IPv6 that replacesand enhances the features of the Address Resolution Protocol (ARP). Secure Neighbor Discovery (SEND) is, respectively, an extension of NDP with extra security. Multicast Router Discovery (MRD) allows discovery of multicast routers.

Internet Group Management Protocol (IGMP) [19]

The Internet Group Management Protocol (IGMP) is used by hosts and adjacent routers on IPv4 networks to establish multicast group memberships. It is a part of IP multicast, and it is used in one-to-many networking applications such as online streaming video. IGMP operates between the client computer and a local multicast router. IGMP messages are carried in bare IP packets with protocol number 2.

Internet Protocol Security (IPsec) [20]

The Internet Protocol Security (IPsec) is a protocol suite for secure Internet Protocol (IP) communications. It authenticates and encrypts each IP packet of a communication session. IPSec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host). It supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

IPsec uses the following protocols to perform various functions;

- Authentication Headers (AH): Provides connectionless data integrity and data origin authentication for IP datagrams, and provides protection against replay attacks.

- Encapsulating Security Payloads (ESP): Provide confidentiality, data-origin authentication, connectionless integrity, an anti-replay service, and limited traffic-flow confidentiality.
- Security Associations (SA): Provides the bundle of algorithms and data that provide the parameters necessary for AH and ESP operations.

Protocol Independent Multicast (PIM) [21] [22]

The Protocol Independent Multicast (PIM) is a family of multicast routing protocols for Internet Protocol (IP) networks that provide one-to-many and many-to-many distribution of data. PIM does not include its own topology discovery mechanism, but instead uses routing information supplied by other routing protocols.

There are four variants of PIM:

- PIM Sparse Mode (PIM-SM): It builds unidirectional shared trees rooted at a rendezvous point (RP) per group, and optionally creates shortest-path trees per source. It is called sparse-mode because it is suitable for groups where low percentage of the nodes will subscribe to the multicast session.
- PIM Dense Mode (PIM-DM): It uses dense multicast routing. It builds shortest-path trees by flooding multicast traffic domain wide, and then pruning back branches of the tree where no receivers are present. Dense mode is ideal for groups where many of the nodes will subscribe to receive the multicast packets.
- Bidirectional PIM: It builds shared bi-directional trees. It never builds a shortest path tree, so may have longer end-to-end delays than PIM-SM.
- PIM Source-Specific Multicast (PIM-SSM): It builds trees that are rooted in just one source, offering a more secure model for a limited amount of applications (mostly broadcasting of content). In SSM, an IP datagram is transmitted by a source S to an SSM destination address G, and receivers can receive this datagram by subscribing to channel (S,G).

1.3.2.3 Routing protocols

Enhanced Interior Gateway Routing Protocol (EIGRP) [23]

The Enhanced Interior Gateway Routing Protocol (EIGRP) is a routing protocol used on a computer networks for automating routing decisions and configuration. This protocol was designed by Cisco Systems and it was only avaiable for Cisco routers. In 2003, partial functionality of EIGRT was converted to an open standard and in 2016 was published with informational status. EIGRP is used on a router to share routes with other routers in the same autonomous system.

All routers contain a routing table that lists the routes to network destinations. If a router cannot find a valid path to the destination, the traffic is discarded. EIGRP is a dynamic routing protocol, which means that routers automatically exchange information about routes and, therefore, the administrator does not have to change the routing table manually. Besides the routing table, routers adittionlly have two more tables.

- Neighbour table. It stores the IP adress of the routers that have a direct connection with this router. If a router is connectedto another with an intermediate router, it will not be recorded in this table.
- Topology table. It keeps record of routes that has learned from neightbouring router tables, and also records the distance (number of intermediate routers) of each route, the feasible successor and the successors (other routes that have the same destination and are loop free). Routes in this table are either labelled as "passive" or "active". Passive means that EIGRP has determined the path for the specific route and has finished processing. Active means that EIGRP is still trying to calculate the best path for the specific route. The router dones not use the routes in this table. A route in this table will be inserted in the routing talbe when is marked as passive, is not a feasible successor and does not have a higher distance than an equivalent path

If there is a change in the network (a link fails, or a router is disconnected), the path becomes unavaiable, and is removed from the routing table. The routing table of a router will be updated, and only the changes since the previous update will be transmitted to the neightbouring routers. The information about the changes in the routing table is not transmitted periodically, but only when a change actually occurs.

EIGRP supports the following features:

- Support for Classless Inter-Domain Routing (CIDR) and variable length subnet masking. Routes are not summarized at the classful network boundary unless auto summary is enabled.
- Support for load balancing on parallel links between sites.
- The ability to use different authentication passwords at different times.
- MD5 authentication between two routers.
- Sends topology changes, rather than sending the entire routing table when a route is changed.
- Periodically checks if a route is available and propagates routing changes to neighboring routers if any changes have occurred.
- Runs separate routing processes for Internet Protocol (IP), IPv6, IPX and AppleTalk through the use of protocol-dependent modules (PDMs).

EIGRP does not operate using the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP). This means that EIGRP does not use a port number to identify traffic. Rather, EIGRP is designed to work on top of layer 3. Since EIGRP does not use TCP for communication, it implements Cisco's Reliable Transport Protocol (RTP) to ensure that EIGRP router updates are delivered to all neighbors completely.

Open Shortest Path First (OSPF) [24] [25]

The Open Shortest Path First (OSPF) is a routing protocol for Internet Protocol (IP) networks that operates in a single autonomous system. OSPF version 2 is designed for IPv4, while OSPF version 3 is designed for IPv6. It works by gathering link state information from available routers and constructing a topology map of the network. The topology is presented as a routing table to the Internet layer which routes packets based solely on their destination IP address. OSPF detects changes in the topology, such as link failures, and creates a new loop-free routing structure. It computes the shortest-path tree for each route using a method based on Dijkstra's algorithm. OSPF does not use a transport protocol, such as UDP or TCP, but encapsulates its data directly in IP packets with protocol number 89. It implements its

own transport layer error detection and correction functions. OSPF uses multicast addressing for distributing route information within a broadcast domain.

OSPF supports complex networks with multiple routers, including backup routers, to balance traffic load on multiple links to other subnets. Routers form adjacencies when they have detected each other. This detection is initiated when a router identifies itself in a Hello protocol packet. Upon acknowledgment, this establishes a two-way state and the most basic relationship. The routers in an Ethernet or Frame Relay network select a Designated Router (DR) and a Backup Designated Router (BDR) which act as a hub to reduce traffic between routers. OSPF establishes and maintains neighbor relationships for exchanging routing updates with other routers. The neighbor relationship table is called an adjacency database. Two OSPF routers are neighbors if they are members of the same subnet and share the same area ID, subnet mask, timers and authentication. OSPF adjacencies are formed between selected neighbors and allow them to exchange routing information. Two routers become adjacent if at least one of them is Designated Router or Backup Designated Router (on multiaccess type networks), or they are interconnected by a point-to-point or point-to-multipoint network type.

OSPF does not carry data via a transport protocol. Instead, OSPF forms IP datagrams directly, packaging them using protocol number 89 for the IP Protocol field. OSPF defines five different message types, for various types of communication:

- Hello: It is used to allow a router to discover other adjacent routers on its local links and networks. The messages establish adjacencies between neighboring devices. uring normal operation, routers send hello messages to their neighbors at regular intervals. If a router stops receiving hello messages from a neighbor, after a set period the router will assume the neighbor has gone down.
- Database Description: It contain descriptions of the topology of the autonomous system or area. They convey the contents of the link-state database (LSDB) for the area from one router to another. Communicating a large LSDB may require several messages to be sent.
- Link State Request: These messages are used by one router to request updated information about a portion of the LSDB from another router. The message specifies exactly which link about which the requesting device wants more current information.

- Link State Update: These messages contain updated information about the state of certain links on the LSDB. They are sent in response to a Link State Request message, and also broadcast or multicast by routers on a regular basis. Their contents are used to update the information in the LSDBs of routers that receive them.
- Link State Acknowledgment: These messages provide reliability to the link-state exchange process, by explicitly acknowledging receipt of a Link State Update message.

Routing Information Protocol (RIP) [26] [27]

The Routing Information Protocol (RIP) is a routing protocol. It uses a hop count to establish the distance between two routers and, in order to prevent loops, establishes 15 as the limit number of hops in a route. If the number of hops is 16, the distance between the two routers is considered infinite. Each router has a routing table with all the routes to each possible destination, and the number of hops to get there. There are 3 versions of RIP: RIPv1, which is the original, RIPv2, which is an updated version of RIPv2, and RIPng, which is the new generation of RIP compatible with IPv6.

The operating principle of the RIP is the following: When a RIP router comes online, it sends a broadcast message to all of its RIP enabled interfaces. All the neighbouring routers that receive the Request message respond back with the Response Message containing their Routing table. The Response Message is also gratuitously sent when the Update timer expires (by deffect, 30 seconds). On receiving the Routing table, the router processes each entry of the routing table as per the following rules:

- If there are no route entries matching the one received then the route entry is added to the routing table automatically, along with the information about the router from which it received the routing table.
- If there are matching entries but the hop count metric is lower than the one already in its routing table, then the routing table is updated with the new route.
- If there are matching entries but the hop count metric is higher than the one already in its routing table, then the routing entry is updated with hop count of 16 (infinite hop). The packets are still forwarded to the old route. A Holddown timer is started and all the updates for that route from other routers are ignored. If after the Hold-down timer (per deffect 180 seconds) expires and still the router is advertising with the same higher hop

count then the value is updated into its routing table. Only after the timer expires, the updates from other routers are accepted for that route.

If the Invalid timer (per deffect 180 seconds) expires and a routing entry has not been updated, the hop counter of that route will be set to 16, marking the route as invalid. Then, if the Flush timer (per deffect 240 seconds) expires, the invalid route entry will be removed

1.3.3 Protocol Selection

1.3.3.1 Choice of the main protocol

The choice of the main protocol will be between SPP, IPv4 and IPv6. Tho make the choice, it is important to take into account that the Astrea constellation is a network that can be of more than two hundred satelites, which will communicate point-to-point. Each node can be the source ,the destination or an intermediate node of a communication route.

SPP has the advantage of being designed to work easily with the protocols of the adjacent layers, while IP needs IP over CCSDS and Encapsulation Service. However, SPP requires a parametre called Path ID, which is the identifier of a Logical Data Path. Since each satellite of Astrea constellation can be the source or the destination of a data path, this means that for a network of 200 nodes, there are 200x199=39800 possible routes. The parameter to indicate the Path ID has a length of 11 bits, which can identify 2048 different routes, which is not enough. Another issue to take into account is that since the ground station nodes of the constellation are moving respect the satellite nodes, their relative position changes and, therefore, paths also change. If the path associated to a Path ID changes during a transmission, or if is not updated for all nodes at the time of the transmission, errors can occur. This does not happen with IP, since instead of Path ID it uses the IP adress of the source and destination node. For this reason, SPP is discarded.

The main differences between IPv4 and IPv6 are the header of the datagram and the IP adresses of the nodes. Since our network is private and it is not intended to be connected to the Internet, nodes can have an arbitrary IP adress assigned. For this reason, IPv4 adresses are better, since they are shorter than IPv6 adresses. The size of the header would also be smaller in IPv4 than IPv6. However, for long datagrams, the extra length of IPv6 headers is

irrelevant. Another difference is that IPv6 datagrams require less processing power, however, since the processing power is very small compared to the power required by the antennas this factor also has little importance in terms of power. However, it is important in terms of time, since less processing means less time to process. Other features of IPv6 that, in Astrea network, do not privide benefits are the multicast and mobility features, which the network ill not have. Additionally, due to the changing nature of the constellation, jumbograms will not be used because a packet so long may be interrumted when the path changes.

The real benefits of IPv6 over IPv4 is that there are less additional protocols compared to IPv4 to perform the same features, since ICMPv6 provide the features of ICMP, ARP and IGMP, and some features of IPv6 itself and its additional protocols have been eliminated since they were already performed by other layer protocols and were redundant. All of this helps to reduce the time required to process the data and this, in long paths, is a significant factor.

If reliable adjacent layer protocols are provided, IPv6 is the best option, due to less processing in routers and more simple additional protocols. Additionally, IPv6 is progressively replacing IPv4 and, therefore, using IPv6 has no risk of being obsolete.

1.3.3.2 Choice of routing protocol

The choice of the routing protocol will be between EIGRP, OSPF and RIP.

EIGRP is a protocol compatible with either IPv4 and IPv6. Contrary to other protocols, it only sends topology changes instead of the whole routing table, allowing for less data transmitted. It also contains more information about routes than other routing protocols, and provides authentication processes.

RIP is a protocol that, compared to EIGRP and OSPF, has the drawback that its time to converge and its scalability are poor. Additionally, RIP uses the User Datagram Protocol (UDP) as its transport protocol. On the other, it is easier to configure than other protocols.

OSPF is a protocol also compatible with IPv4 and IPv6. Unlike EIGRP, each router exchanges its adjacency links with adjacent routers and then, each router creates its own map

of the network and, using this map, each router creates its own routing table. However, it has mechanisms to ensure that thare are not loops in the network.

Taking into account that nodes in the Astrea network have an order of magnitude of 200 and is continously changing the data paths. Also, since Astrea is a network where a node can be the beginning or the end of a communication, this means that for a given node there has to be a route to every other node in the network, and for a network of 200 nodes, there are 199 possible routes for the 200 nodes, which is a total of 39800 different entries in the routing table only for the satellite nodes. Since RIP has longer time to converge compared to other protocols, and due to the huge size of the routing table, RIP is discarded.

EIGRP does not have this problem because it does not transmit the whole routing table, but only the changes. Although the network is continously moving, the paths between the satellite nodes remain the same. The problem happens with the ground nodes, which are continouslly changing its position respect the satellite nodes due to Earth's rotation. And since each satellite node can communicate with every ground station, the number of entries in the routing table that will be updated for a network of 200 satellite nodes and 5 ground stations is 200x5, which is 1000 entries that will be updated frequently. Since OSPF does not transmit the routing table but only the adjacencies, only 205 entries will be transmitted. This reduces the time to share the updated information to the whole network. For this reason, OSPF is chosen.

1.3.3.3 Choice of complementary protocols

The choice of which protocols include will depend on the main protocol of the network layer and the degree of services featured by the communication process.

Since IPv6 has been chosen, IP over CCSDS and Encapsulation Service are necessary. Additionally, ICMPv6 greatly expand the features of IPv6 such as flow control. Security features are already provided in the Data Link layer and, therefore, IPsec is not necessary. Also, no multicast features are required, so no multicast protocols will not be used.

1.3.3.4 Conclusion

It has been decided that IPv6 will be the network layer protocol, complemented with IPoC, Encapsulation Service and ICMPv6, and with OSPF as the routing protocol.

1.3.4 Final structure

As the protocols have already been chosen, it is time to establish how will be the headers of the different protocols.

The IPv6 header will depend greatly on the protocol of the upper layers, or the auxiliar protocol (OSPF, ICMPv6). The main parameters of the IPv6 header, that can be seen in Figure 1.3.5, are the following:

- **Version** Current version of IP, which for IPv6 is 6 (bit sequence 0110).
- **Traffic Class**. The bits of this field hold two values. The 6 most-significant bits are used for differentiated services, which is used to classify packets. The remaining two bits are used for ECN; priority values subdivide into ranges: traffic where the source provides congestion control and non-congestion control traffic.
- **Flow Label**. The flow label when set to a non-zero value now serves as a hint to routers and switches with multiple outbound paths that these packets should stay on the same path so that they will not be reordered.
- **Payload Length**. The size of the payload in octets, including any extension headers. The length is set to zero when a Hop-by-Hop extension header carries a Jumbo Payload option.
- **Next Header**. Specifies the type of the next header. This field usually specifies the transport layer protocol used by a packet's payload. When extension headers are present in the packet this field indicates which extension header follows. The values are shared with those used for the IPv4 protocol field, as both fields have the same function (see List of IP protocol numbers in [28]).
- **Hop Limit**. This value is decremented by one at each intermediate node visited by the packet. When the counter reaches 0 the packet is discarded.

- **Source Address**. The IPv6 address of the sending node.
- **Destination Address**. The IPv6 address of the destination node.

It has been stated that, since Astrea network is a private network that will not be connected to the Internet, IP adresses will be arbitrary assigned to the nodes of the network.

For the IPoC heaver, the value for IPv6 datagrams is 87, so the header of OPoC will be 01010111

For the Encapsulation Service, depending of the length of the data unit transmitted, the header will very. For data units up to 65531 octets, the Encapsulation Service header will be the following: 11101010-00000000-XXXXXXXX-XXXXXXXX, where XXXXXXXX-XXXXXXXX is the binary number of the total length of the Encapsulation Packet, including the Encapsilation Packet header.

1.4 Layer 4: Transport and Session

This layer is the one in charge of the free-of-error transference of data from one process to another. Therefore, its goal is to provide and guarantee a reliable and cheap flow of the data.

Whereas the network layer oversees source-to-destination delivery of individual packets, it does not recognize any relationship between those packets. It treats each one independently, as though each piece belonged to a separate message, whether or not it does. The transport layer, on the other hand, ensures that the whole message arrives intact and in order, overseeing both error control and flow control source-to-destination level.

A transport layer can be either connectionless or connection-oriented. A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine. A connection-oriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data is transferred, the connection is terminated.

In the transport layer, a message is normally divided into transmittable segments. A connectionless protocol, such as UDP, treats each segment separately. A connectionoriented protocol, such as TCP and SCTP, creates a relationship between the segments using sequence numbers.

The transport layer is responsible for process-to-process delivery, i.e, the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship.

Regarding addressing, at the transport layer, it is necessary a transport layer address, called a port number, to choose among multiple processes running on the destination host. The destination port number is needed for delivery, whereas the source port number is needed for the reply.

The addressing mechanism allows multiplexing and demultiplexing by the transport layer.

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	Telnet	Telnet protocol
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Service
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

1.4.1 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol. The only new feature regarding IP is that it provides process-to-process communication instead of host-to-host communication, and performs a very limited error checking. It might seem a powerless protocol, but its main point is that is a very simple protocol using a minimum of overhead. Therefore, if a process wants to send a small message and no extremely reliability is required, UDP is a good choice.

Nevertheless, regarding the aim of this project, it is unacceptable to use UDP, since reliability is a key factor and must be taken into account.

1.4.2 Stream Control Transmission Protocol (SCTP)

The Stream Control Transmission Protocol is a new reliable, message-oriented transport layer protocol. Nevertheless, it has been designed and implemented mostly for Internet applications, such as IUA or SIP. But precisely it does not fit the goal of this project.

Therefore, as there is a better choice (which will be deeply and widely explained in the following section), this protocol will not be considered.

1.4.3 Transmission Control Protocol (TCP)

The Transmission Control Protocol is again a process-to-process protocol. Consequently it uses port numbers. The main difference with the UDP is that TCP is a connection-oriented protocol, which means that creates a virtual connection between two TCP's in order to send data. Moreover, TCP uses flow and error control mechanisms. It is then a more reliable protocol that UDP. It adds connection-oriented and reliability features to the services of IP.

This will be the protocol chosen for this project, so it will be explained in detail in this section.

1.4.3.1 TCP Services

Process-to-process communication: Like UDP, TCP provides this type of communication, using port numbers. In the following image there are the main well-known port numbers used by

TCP.

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FIP, Data	File Transfer Protocol (data connection)
21	FIP, Control	File Transfer Protocol (control connection)
23	TELNET	Tenninal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Stream Delivery Service: as has been mentioned before, TCP, unlike UDP, is a stream-oriented protocol. UDP does not recognize any relationship between the datagrams. TCP, in contrast, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. A way of explaining this would be an environment in which the two processes seems to be linked by and imaginary "tube" that carries the data across the Internet. The sending process produces the stream of bytes and the receiving process consumes them. This is, the first writes and the last reads.

Sending and Receiving Buffers: Since the sending and receiving processes might not write or read data at the same speed, there is a need for storage in TCP. Therefore, TCP includes two buffers, the sending buffer and the receiving buffer. A deeper look into those buffers can be performed by looking at the bibliography.

Full-Duplex Communication: TCP allows full-duplex service, so that data can flow in both directions at the same time. Each TCP has a sending and receiving buffer, and segments move

in both directions. This feature is very important for the goal of this project.

Segments: Although buffering solves the problem of different speeds of producing and consuming, there is still one impotant feature to be discussed. The data needs to be sent in packets, not as a endless stream of bytes. Therefore, TCP groups a number of bytes together into a packet called a segment. A header is added to each segment for control purposes.

1.4.3.2 TCP features

In order to provide the services that have been explained, TCP has some features that will be briefly discussed.

1.4.3.3 Numbering Systems

TCP keeps track of the segments being transmitted or received, using the header previoulsy discussed. There are in addition two fields, the sequence number and the acknowledgement number, which refer to the byte number, not the segment number.

TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them. Typically, it generates randomly a number between 0 and $2^{32} - 1$ for the number of the first byte. For example, if the random number happens to be 1427 and the total data to be sent are 5000 bytes, the bytes are numbered from 1427 to 6426. This system is used for flow and error control.

After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent. The sequence number for each segment is the number of the first byte carried in that segment. This is, the value in the sequence number field of a segment defines the number of the first data byte contained in that segment.

The value of the acknowledgement field in a segment defines the number of the next byte a party expects to receive. It is a cumulative number.

1.4.3.4 Flow Control

TCP provides flow control, which means that the receiver can control the amount of data that if to be sent by the sender. The purpose of this is to avoid over-whelmed receivers.

1.4.3.5 Error Control

In order to provide a reliable service, TCP implements an error control mechanism. It considers a segment as the unit of data for error detecting, even though there is also a byte-oriented control mechanism.

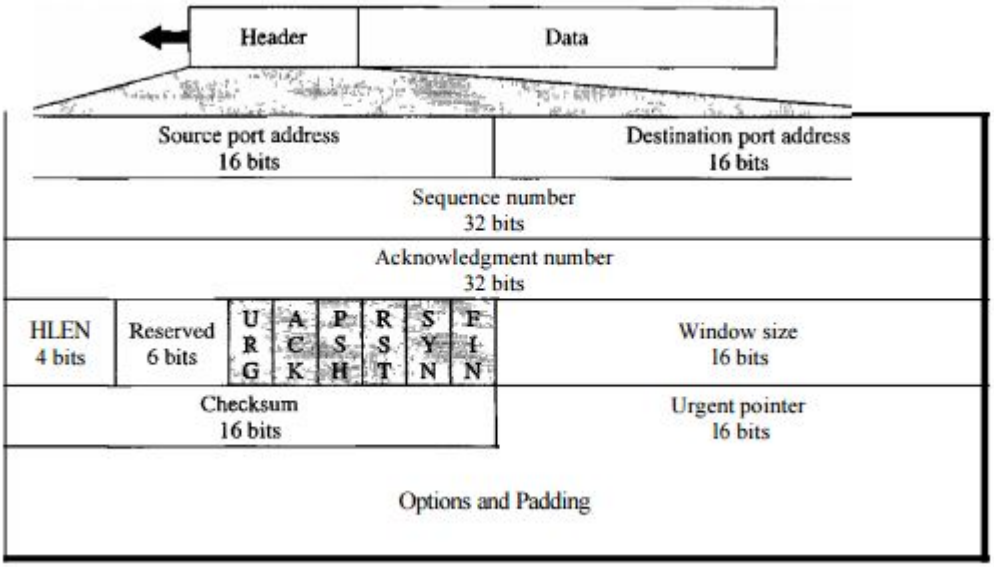
1.4.3.6 Congestion Control

TCP also takes into account congestion in the network, by the detenning of the flow depending on the level of congestion in the network.

1.4.3.7 Segment

As has been explained before, a packet in TCP is called a segment. The aim of this point is to explain in detail what a segment is and how its structure is.

The typical format of the segment is shown in the next figure.



The segment consists of a 20 to 60-byte header, followed by data frim the application program. The byte is 20-byte long if there are no options, and up to 60-bytes if there are options.

The main parts of the format are to be discussed in the following lines.

1.4.3.8 Source Port Adress

This is a 16-bit field that states the port number of the application program in the host that is sending the segment.

1.4.3.9 Destination Port Adress

It is also a 16-bit that defines the port number of the application program in the host that is receiving the segment.

1.4.3.10 Sequence Number

This 32-bit field defines the number assigned to the first byte of the data contained in the segment considered. This numeration has been previously explained.

1.4.3.11 Acknowledgement Number

This is a 32-bit field that defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x, it defines x + I as the acknowledgement number.

1.4.3.12 Header Length

A 4-bit field that indicates the number of 4-byte words in the TCP header. As seen, the length of the header can be between 20 and 60 bytes. Then, the value of this field can be between 5 and 15 (since $5 \times 4 = 20$, and $15 \times 4 = 60$).

1.4.3.13 Reserved

This is a 6-bit field reserved for future usage.

1.4.3.14 Control

This field defines 6 different control bits or flags. One or more of those bits can be set at a time.

1.4.3.15 Window Size

This fiels defines the size of the window, in bytes, that the other party must maintain. Since the length of this field is 16 bits, the maximum size of the windows is $2^{16} = 65535$ bytes.

1.4.3.16 Urgent Pointer

Another 16-bit field, which is only valid if the urgent flag is set, which means that the segment contains urgent data. In actually defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

1.4.3.17 Options

As has been explained, there can be up to 40 bytes of optional information in the TCP Header. This is the purpose of this last field.

1.4.3.18 Adaptation to space needs

TCP was estblished for wired connections initially. Therefore, in order to be eligible for the purpose of this project, it is highly recommended that some slight modifications are done. The Space Communications Protocols Specification (SCPS) defines a set of revisions to the protocols to enable them to operate properly. This is, SCPC-TCP becomes an "upgraded" TCP, specially designed for space application.

With SCPS, TCP the bandwidth of an existing link will be utilized to a significantly higher percentage and more efficiently. It also supports end-to-end communications between applications and is designed to meet the needs of a broad range of space missions.

This is all achieved because of an extension that is added to the header shown before. This extension header is shown next. Each line is a octet of bits; i.e, 8 bits:

1.4.4 Choice of protocol for the transport layer

Three protocols have been discussed, the UDP, the SCTP and the TCP. The first one has some disadvantages which make it not suitable for the purpose of the project, such as the fact that no reliabililty is guaranteed, for example, amongst others. The second one is designed mostly for Internet applications, which does not fit the goals of this project. Therefore, the only candidate suitable for the project is the TCP, Transmission Control Protocol, which has already been widely explained and analyzed. As it has the required features that the project demands, it is the chosen protocol for this layer. Also, as it has been established, it is very recommended to use the extension SCPS, due to adaptation to space needs.

1.5 Global Overview

For the sake of clarification, all the elected options are going to be put together obtaining the desired fully desgined **protocol stack**.

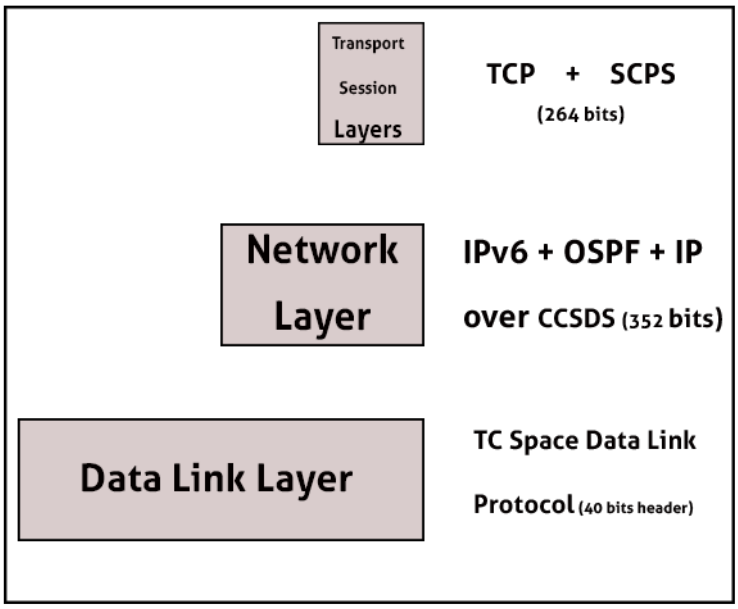


Figure 1.5.1: Overall space communication protocol stack

In total, the overhead is **656 bits**, whith conservative calculations. Hence, the quantity is negligible in comparison to the data rate.

Chapter 2

Ground Segment Protocols

Chapter 3

Ground Station Design

Chapter 4

Bibliography

[1] Behrouz a. Forouzan. *Data Communications and Networking - Global Edition*. 2012.

[2] CCSDS Secretariat. Overview of Space Communications Protocols. (CCSDS 130.0-G-3):43, 2014.

[3] CCSDS. TC Space Data Link Protocol. (September), 2010.

[4] CCSDS. TM Synchronization and Channel Coding—Summary of Concept and Rationale. *CCSDS Green Book*, (November 2012), 2012.

[5] International Telecommunication Union. *X.200: Data Networks and open system communications*, volume 4. 1994.

[6] CCSDS. *Report Concerning Space Data System Standards - Overview of Space Communications Protocols*. Number CCSDS 130.0-G-3. 2014.

[7] CCSDS. *Recommendation for Space Data System Standards - Space Packet Protocol*. Number CCSDS 133.0-B-1. 2003.

[8] CCSDS. *Recommendation for Space Data System Standards - Encapsulation Service*. Number 133.1-B-2. 2009.

[9] Space Assigned Number Authority (SANA) Registry. http://sanaregistry.org/.

[10] CCSDS. *Recommendation for Space Data System Standards - IP over CCSDS Space Links*. Number CCSDS 702.1-B-1. 2012.

[11] Information Sciences Institute University of Southern California 4676 Admiralty Way and California 90291 Marina del Rey. *Internet Protocol Specification*. 1981.

[12] S Deering and R Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. 1998.

[13] Space Assigned Number Authority (SANA) Registry: Packet Version Number. http://sanaregistry.org/r/packet_version_number/packet_version_number.html.

[14] Space Assigned Number Authority (SANA) Registry: Application Identifier. http://sanaregistry.org/r/space_packet_protocol_application_process_id/space_packet_protocol_

[15] Space Assigned Number Authority (SANA) Registry: Protocol Identifier. http://sanaregistry.org/r/protocol_id/protocol_id.html.

[16] Space Assigned Number Authority (SANA) Registry: IP Extension header. http://sanaregistry.org/r/ipe_header/ipe_header.html.

[17] J Postel. Internet Control Message Protocol. pages 1–21, 1981.

[18] A Conta, S Deering, and M Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. 6:1–24, 2006.

[19] H Holbrook, B Cain, and B Haberman. *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast*. 2006.

[20] S Kent and K Seo. Security Architecture for the Internet Protocol. pages 1–101, 2005.

[21] B Fenner, M Handley, H Holbrook, I Kouvelas, R Parekh, Z Zhang, and L Zheng. *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*. 2016.

[22] A Adams, J Nicholas, and W Siadak. *Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)*. 2005.

[23] D Savage, J Ng, S Moore, D Slice, P Paluch, and R White. *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*. 2016.

[24] J Moy. *OSPF Version 2 Status*. 1998.

[25] R Coltun, D Ferguson, J Moy, and A Lindem. *OSPF for IPv6*. 2008.

[26] G Malkin. RIP Version 2. pages 1–39, 1998.

[27] G Malkin and R Minnear. RIPng for IPv6. pages 1–19, 1997.

[28] Internet Assigned Number Authority (IANA) Registry: Protocol Numbers. http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml.