Telemetry / Telecommand Frontend - IITMSAT

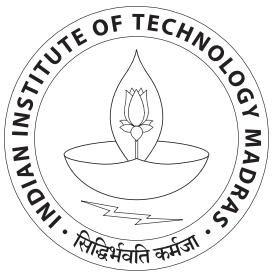
A Project Report

submitted by

ANOOP R SANTHOSH

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

May 2014

THESIS CERTIFICATE

This is to certify that the thesis entitled **Telemetry / Telecommand Frontend - IITMSAT**, submitted by **Anoop R Santhosh**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Shankar. Balachandran
Project Guide
Assistant Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisor Dr. Shankar Balachandran for his inspiration, support and guidance throughout this project and entire 4 years in general. He is a great teacher and a tremendous mentor.

I would also like to thank Dr. Harishankar Ramachandran, for the time he devoted towards discussing the design and motivating me in general.

My sincere thanks also goes to Dr. David R. Koilpillai, for his invaluable feedback and the inspiring talks during every IITMSAT session.

I would like to extend my gratitude to Mr.Akshay Gulati, the person who motivated me to join IITMSAT team . He has been a wonderful mentor , devoting a lot of his time discussing the design and implementaion.

I am grateful to the entire Computer Science Department ,IITMSAT team and my batchmates for their support.

Last, but not least , I am grateful to my parents and friends for their constant support and motivation.

ABSTRACT

KEYWORDS: TMTC FrontEnd, AX.25, IITMSAT

IITMSAT is a low orbit nano satellite being developed by students at IIT Madras. The satellite project has various subsystems, both hardware and software . Ground station software is an important part of it. TMTC Frontend is an important sub module within the ground station software. It essentially functions as the link layer and implements the link layer protocol which in this case is AX.25 .

The main functionality of this module includes AX.25 protocol encoding/decoding , flow control including acknowledgements and counters and packet reassembly. The overall design is loosely inspired by the SwissCube ground station design. But to satisfy IITMSAT mission requirements which are different from SwissCube, many modifications were made, especially in the telecommand transmitter part. The link layer protocol though based on standard AX.25 protocol, has been modified to meet our requirements.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CCSDS	Consultative Committee for Space Data System
CRC	Cyclic Redundancy Check
ECSS	European Cooperation for Space Standardization
GS	Ground Station
ISRO	Indian Space Research Organisation
MCS	Mission Control System
MDR	Mission Data Repository
MIB	Mission Information Base
SDR	Software Designed Radio
SSID	Secondary Station Identifier
TC	Telecommand
TM	Telemetry
TMTC	Telemetry Telecommand
VC	Virtual Channel

CHAPTER 1

INTRODUCTION

This chapter provides a brief introduction to IITMSAT in general and the particular problem statement i.e, TMTC Frontend .

1.1 An introduction to IITMSAT

IITMSAT is a student initiated satellite project being developed at IIT Madras. The project mission is to launch a nano satellite into a low earth orbit of altitude 600 - 1000 km .The orbit passes through the mission's region of interest, the ionosphere. Being an auxillary satellite on the launch vehicle, IITMSAT does not have prior control over the exact altitude. IITMSAT is proposed to carry an on-board High Energy Particle Detector (HEPD) and measure changes in charged particle flux in the upper ionosphere. Data gathered by HEPD, including orientation, time, position of the satellite and flux data is referred to as Science Data (SC). SC data will be stored on the on-board memory of the satellite. Besides HEPD, satellite has various sensors on-board to monitor parameters like temperature, current, voltage etc. This data is referred to as Housekeeping (HK) data.

Both HK data and SC data are transmitted to the ground station when the satellite comes in the field of view. IITMSAT proposes to have just one ground station set up in the IIT Madras campus. The ground station will receive both

HK data and SC data. HK data will be used to monitor the health of the satellite whereas SC data will be analyzed by the ground station software and displayed to users in human interpretable formats.

The main configurations/features of the satellite are as given below:

- **Orbit :** Low earth Orbit
- **Altitude :** 600 km to 800 km
- **Inclination :** >45 degrees
- **Mission Life Time :** >1 year
- **Dimensions :** 29 cm x 29 cm x 26 cm
- **Mass :** 15 kg
- **Payload :** High Energy Particle Detector

1.2 An overview of Ground Station software

Ground station software is responsible for handling the operations of the satellite from ground and process the information transmitted by the satellite on down-link.It provides an interface for interacting with satellite and analyzing data besides keeping track of various parameters of the satellite.The major modules within the ground station software are :

- **User Interface :** UI allows human users to select commands to control the satellite. It also displays various data collected by the satellite payload and housekeeping data after processing it in appropriate ways.
- **Mission Control System** : MCS is responsible for converting the user input to CCSDS telecommands before dispatching them to the satellite . It is also responsible for archiving all packets sent and received. It is also the unit which schedules commands and processes the data received from satellite (CCSDS packets) on downlink. It interacts with UIs on one end and TMTC Frontend on the other. It transmits and receives CCSDS packets to/from TMTC Frontend.

- **TMTC Frontend :** This layer acts as a link layer for communication between MCS and satellite. Functionalities of TMTC Frontend are mentioned briefly in the next section on problem statement. Chapter 2 explains more details of the working of this layer.

1.3 TMTC Frontend - The problem statement

The main problem statement is to implement a robust TMTC Frontend. TMTC Frontend is the layer between MCS and GS hardware. It acts as a link layer for communication between GS and satellite . It is expected to provide the following functionalities :

- **Up-link :**
  - Encoding CCSDS packets (telecommand) to AX.25 frame.
  - Archive AX.25 telecommand frames.
  - Flow Control (resending, counters, acknowledgments etc).
- **Down-link :**
  - Decode AX.25 frame (telemetry) and reassemble the application data to obtain CCSDS packets.
  - Archive raw AX.25 telemetry frames.
- **Replay :**
  - Provide functionality to replay archived raw AX.25 telemetry frames.

TMTC Frontend is explained in more detail in chapter 2.

1.4 Motivation

A robust link layer is very essential for proper communication with the satellite. This layer is responsible for encoding application data (CCSDS packets) to AX.25

protocol on up-link and decoding AX.25 frames to CCSDS packets on down-link. There are two major concerns about the communication link with the satellite which necessitates a robust link layer. One concern is the reliability of the communication link to the satellite. Being an satellite, it is not feasible for IITMSAT to have a professionally competent communication system. This combined with various external interferences like radiation effects and other space environment effects can reduce the reliability of the link. Another major concern is the limited time available to communicate with the satellite. As mentioned earlier IITMSAT proposes to have only one ground station. During each pass GS gets a communication window of around 8 minutes. With just one GS, only two to three passes can be expected in a day. This gives a communication window of roughly around 16 - 24 minutes in a day. Maximum data transfer should be ensured within this limited time.

A robust link layer is essential to counter the effects of limited link reliability and communication time.The link layer should be able to handle acknowledgments and resending of dropped packets automatically without involving the application layer. Once the application layer (in IITMSAT case ,MCS) is involved in flow control like acknowledgments, resending etc, it can slow down operations and affect the amount of data transferred. Since all data/command transfer between MCS and satellite happens through this layer, failure of this layer can cut off communication with the satellite.

1.5 SwissCube - a similar satellite mission

SwissCube is CubeSat class satellite launched and operated by Ecole Polytechnique Fdrale de Lausanne (EPFL), Switzerland. A CubeSat is a miniaturized satellite of volume one liter which is usually used for space research. It was launched a Polar Satellite Launch Vehicle in 2009. It was expected to last three to twelve months, but the mission was extended for an additional 18 months. When IITMSAT project was initiated in 2010, the initial design and model was largely influenced by SwissCube. As the IITMSAT project matured, it has evolved much more than SwissCube project in terms of mission specifications. Still IITMSAT design, atleast at a high level is heavily based on SwissCube. A lot of terminology and organization details used within IITMSAT project has been derived from SwissCube model. Though there are marked differences in the implementation of ground station software of IITMSAT compared to SwissCube, the high level design is still influenced by SwissCube . The organization of UI, MCS, TMTC Frontend etc are based on the SwissCube design. Though a discussion on the details of SwissCube is beyond the scope of this document, certain similarities and differences are mentioned at places where it is deemed necessary.

1.6 Contributions of this work

Though the design is loosely based on GS software design of SwissCube, there are quite a few differences. The transmission of data was not a big concern in case of SwissCube, which is not the case with IITMSAT. Hence in SwissCube GS software, resending dropped frames is handled at MCS through manual control.

However, in case of IITMSAT, we expect to transmit more commands/ data in uplink and cannot depend on manual control for frame retransmission. So our design involves automated retransmission at link layer itself, instead of involving application layer.

Another major difference from SwissCube is that we have modified the AX.25 frame to support 256 outstanding telecommand frames at a time where as SwissCube design allows only for 4. Owing to the manual control over packet transmission , 4 outstanding packets were sufficient for SwissCube. But this is not the case with IITMSAT.

IITMSAT TMTC Frontend ensures minimal human interference in frame transmission as opposed to SwissCube. This ensures faster and more reliable communication with the satellite. All the expected features of TMTC Frontend mentioned in the problem statement were implemented. Modified AX.25 protocol encoding/decoding library was implemented. A new telecommand transmitter was designed and implemented. Telemetry receiver was also implemented .Reassembly unit was implemented to support large data transfer. Archiving support for both telecommand and telemetry frames was implemented with a SQL database. Replay functionality was also implemented and a small GUI was created for replay controller. Finally all the sub-modules were integrated together to build a robust TMTC Frontend, which was tested using simulations of MCS and GS hardware.

1.7 Organisation of report

This report is organised into 6 chapters. Chapter 2 explains a few terminology involved and frame structure of the protocols used. Chapter 3 gives a detailed explanation of the design of TMTC Frontend. Chapter 4 discusses implementation details . Chapter 5 explains how the software system was tested. Chapter 6 is the Conclusion chapter .

CHAPTER 2

BACKGROUND DETAILS

This chapter gives a brief background on IITMSAT ground station software and detailed specifications of TMTC Frontend .

2.1 Ground Station Software

As briefly mentioned in the introduction, ground station software is responsible for providing an interface to interact with the satellite and control the operations of the satellite from ground. The main user requirements are :

- Control the satellite and monitor its operations.
- Process the downlink data and provide payload data in human interpretable format.
- Monitor various house keeping parameters like voltage and alert users in case of an issue or takes action whenever possible.
- A GUI for easy human interaction.
- Implement application layer protocols in compliance with ECSS standards. CCSDS packets are used for application data encoding.
- Provide a reliable communication channel to the satellite.

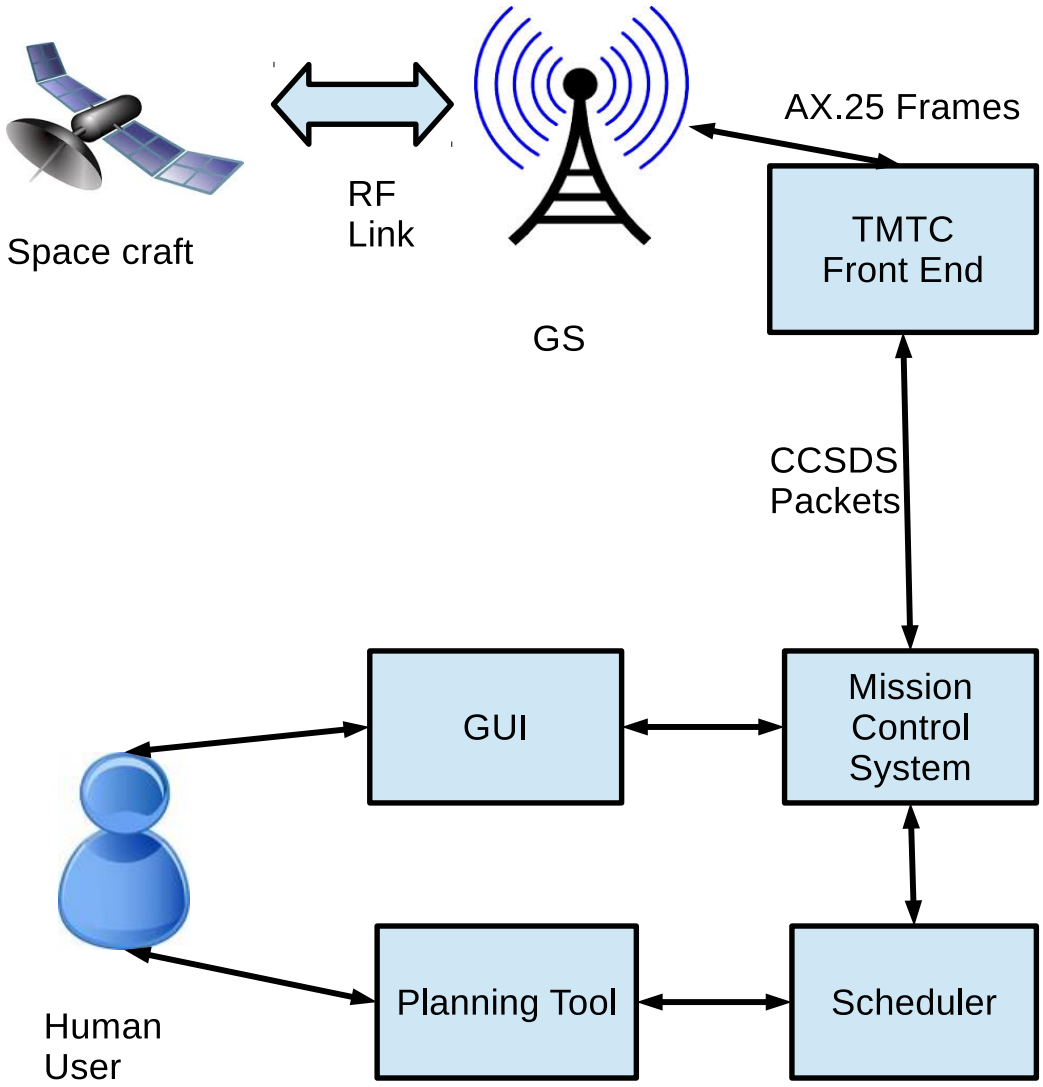


Figure 2.1: Ground Station software block diagram

Note : Within the CCSDS packet telemetry and telecommand concept , an on-

board application process is defined as the source of telemetry source packets and the sink for telecommand packets [3].Throughout this report telecommand refers to commands send to the satellite from GS and telemetry refers to data received by GS from the satellite.

A human user interacts with the ground station software through various graphical user interfaces. There are different GUIs like Mission Configuration Client, Telecommand Manager, Monitoring Data Display Client etc. Mission Control System (MCS) is responsible for handling user inputs from the GUI.

During telecommand operations (uplink), the user selects telecommands for giving instructions to the satellite. The CCSDS encoding for all the telecommands are defined in Mission Information Base (MIB) which is a part of MCS. All telecommand and telemetry packet encoding are defined in MIB. After encoding information into CCSDS packets, these packets are packets are dispatched to TMTC Frontend for transmitting to the satellite. TMTC Frontend as mentioned earlier is the link layer. It encodes the CCSDS packets received from MCS into AX.25 telecommand frames. TMTC Frontend archives the frames and based on the flow control mechanism, transmits the frames to the satellite. TMTC Frontend drops off AX.25 telecommand frames to the Software Defined Radio (SDR) module in IITMSAT case. An SDR is used to implement hardware components of a typical radio communication system by means of software. GNU Radio is the SDR used by IITMSAT. SDR interacts with hardware components rather than TMTC Frontend. The GS hardware then transmits packets over a RF link to the satellite, which receives it and processes accordingly.

During telemetry operations (downlink), the satellite transmits AX.25 telemtry

frames through the RF link, which is the received by GS hardware and passed on to SDR. SDR after processing the frames, transfers them to TMTC Frontend. AX.25 frames are decoded and reassembled to obtain CCSDS packets at TMTC Frontend. It also archives ras AX.25 telemetry frames in an SQL database. If the received frame is an acknowledgement frame, it is passed to the telecommand transmitter. Otherwise the reassembled CCSDS packets are transferred to the MCS. MCS processes the CCSDS frames with the help of encodings defined in MIB. It then displays the information to the human user through the GUI. It also monitor the health of the satellite from housekeeping data.

Besides this MCS also contains another data repository called Mission Data repository (MDR). All telecommands and telemetry which passes through the MCS is archived in MDR. Planning tool and scheduler are not a very essential component of GS software. They can be used for intelligent planning and scheduling of communication with the satellite so as to ensure efficient communication.

The overall design of IITMSAT GS is inspired by the SwissCube GS, though the design of each component is not exactly similar to SwissCube.

2.2 TMTC Frontend

TMTC Frontend as mentioned earlier acts a link layer between MCS and satellite.It handles protocol eencoding.decoding, flow control, reception and transmission of frames etc.

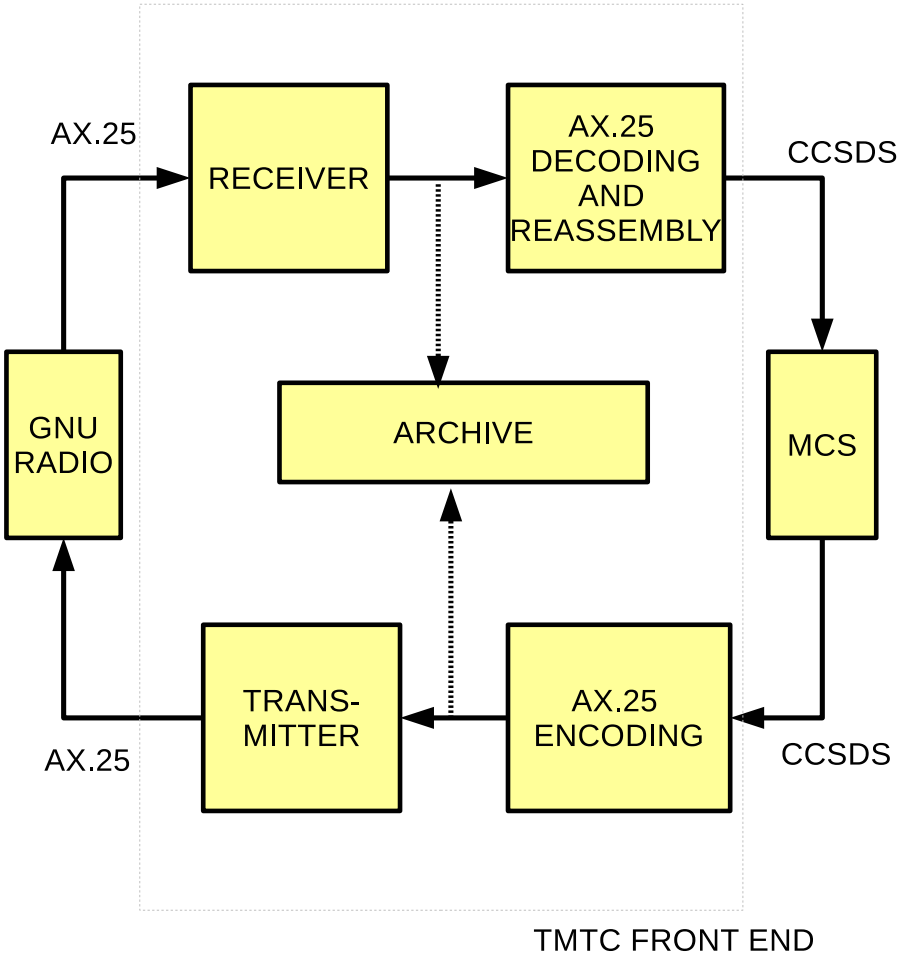


Figure 2.2: TMTC Frontend Block Diagram

Uplink : During uplink or telecommand transmission operations, TMTC Frontend receives CCSDS packets from MCS. These packets are encoded into modified AX.25 telecommand frames. The protocol frame structure is explained in next chapter. After it the packets are encoded to CCSDS frames, they are added to a

transmission queue. The frames are also archived in a SQL database. The flow control mechanism described in next chapter handles the transmission of the frames from that point. When the conditions are suitable for dispatch of the packet, the transmitter drops off the AX.25 frames to the GNU radio module. GNU radio handles rest of the communication with the satellite.

Downlink : During downlink or telemetry reception operations, TMTC Frontend receives AX.25 telemetry frames from GNU Radio. Receiver checks for bit stream errors with a CRC check and then the byte array is parsed to obtain various field of the AX.25 telemetry frames. The source and destination addresses are checked to ensure the validity of the frame. Legitimate frames are then archived and forwarded to appropriate reassembly unit. Reassembly unit reassembles CCSDS packets from different frames and when a CCSDS frame is completely reassembled, it is forwarded to the MCS.

Replay : As mentioned above, all raw AX.25 frames received by the telemetry receiver is archived in an SQL database. Replay functionality allows the user to replay those frames. The users through a GUI can select frames which were received in a time range and simulate the frame reception that happened in that time range. This is useful when some module like MCS fails and we want to simulate operations again. Replay controller retrieves frames from the database and creates a reassemby unit which reassembles CCSDS packets and forwards them to MCS.

2.3 Link Budget

As mentioned earlier, communication link with the satellite may not be extremely reliable. Hence the link should be modeled properly. However such a communication link can be affected by many factors like radiations, ionospheric charged particles etc. For modeling such a communication link, a complex set of parameters need to be considered. Hence rather than creating a model of our own, we have decided to use the SwissCube estimates. Since the altitude and parameters of both satellites are quite similar, this would be a good approximation. The bandwidth according to SwissCube model is 1200 bits/sec and error rate in 7%.

CHAPTER 3

COMMUNICATION PROTOCOL

This chapter briefly discusses the link layer communication protocol used by IITM-SAT.

3.1 Standard AX.25 protocol

AX.25 is a data link layer protocol derived from X.25 protocol suite and designed for use by amateur radio operators[1]. AX.25 is most frequently used to establish direct point to point links.SwissCube uses their own modified version of AX.25. Since the overall IITMSAT design is based on SwissCube design, IITMSAT has decided to use AX.25 frames modified to meet our needs. Most of the fields of the standard frame structure are useful for IITMSAT.

There are generally three types of AX.25 frames:

- **Information Frame :** Used for data transfer.
- **Supervisory Frame :** Used for supervisory link control.
- **Unnumbered Frame :** Provides additional control over the link.

These frames can be distinguished by the value of control field. A discussion on full features of the standard protocol is not necessary for understanding of our system . Hence it is omitted from this document. For more details on the protocol, refer [1].

3.2 Modified AX.25 frame structure for IITMSAT

The protocol and acknowledgement mechanisms which comes with the standard AX.25 protocol does not suit our needs. For example , the protocol allows for only 8 outstanding packets, while our system needs more than 8 out standing packets.

We have decided to use only the frame structure of AX.25 protocol for our system and not the flow control mechanism described by the protocol. We have modified even the frame structure to suit our needs by adding extra fields and changing the interpretation of some fields (from the standard protocol).We are just using the frame encoding with our modifications and not any other standard features of the protocol. For the sake of convenience this modified version will still be referred to as AX.25 protocol/frame throughout this document.

The modified frame structure is briefly described below. The header and trailer common for both telecommand and telemetry will be discussed first, followed by modifications specific to each of them.

3.2.1 Modified AX.25 Frame

The frame structure is as shown below :

Flag	AX.25 Transfer Frame Header (128 bits)				Information Field	Frame Check Sequence	Flag
	Dest. Address	Source Address	Control Bits	Protocol Identifier			
8	56	56	8	8	0 – 2048	16	8

Figure 3.1: AX.25 Transfer Frame Format.Adapted from [1]

Few important fields are briefly explained below :

- **Flag :** A flag field is used to delimit the frames and occurs at both beginning and end. It is 8 bits long and has value 01111110 (0x7E) . To ensure that the flag sequence does not appear anywhere else in the frame, bit stuffing is done. A '0' bit is interested after 5 continuous '1' bits.
- **Destination and Source Address :** Both have similar structure and length of 56 bits. Destination and Source represent satellite or ground station depending on whether it is uplink or downlink. The address field structure is given below:

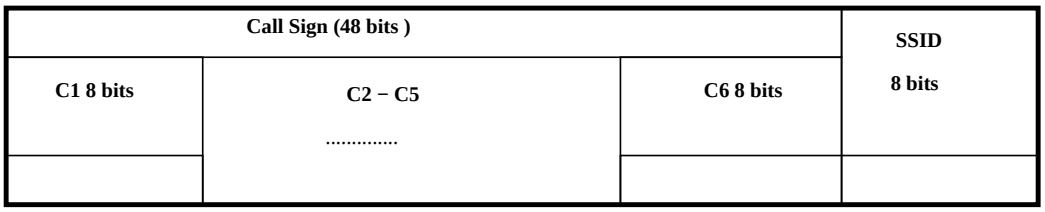


Figure 3.2: AX.25 Address Field.Adapted from [1]

  - **CallSign :** 6 upper case letters.They are placed in C1 to C6 fields.
  - **SSID :** 4 bit integer used to identify multiple stations using the same call sign . They are placed in fields 3 to 6, with fixed values for other fields.

- **Control bits :** 8 bits long. Originally used to differentiate between the types of frames mentioned above, we no longer use it in our version. It is assigned a fixed value of 00000011 (0x03). This actually refers to an unnumbered frame.
- **Protocol Identifier :** 8 bits long.Originally used to represent the layer 3 protocol used, we have modified this field to indicate if a frame is an acknowledgement frame or not. Value 00000011(0x03) indicates that it is an acknowledgement frame. Otherwise value is 11110000(0xF0), which as per standard protocol indicates no layer 3 protocol implemented.
- **Frame Check Sequence :** CRC-CITT is used to calculate a 16 bit CRC. The polynomial used is $x^{16}+x^{12}+x^5+1$.*ACRCcheckisessentialforerrordetectionaftertheframereception*

3.2.2 Telecommand Information Field Usage

In case of telecommands, the only extra support needed is a frame counter of limit more than 8. To number the frames , the first byte of the information field is used

as a frame counter. Thus we have a modulo 256 counter for frames. Rest of the information field carries telecommand data , with a maximum size of 2040 bits or 255 bytes.Virtual channels are not supported on uplink.

3.2.3 Telemetry Information Field Usage

Telemetry information field has much more modifications than telecommand. An extra header and trailer are added. This is because on the top of providing a modulo 256 counter, support for virtual channels and large data transfer has to be provided. Virtual channels and large data transfer are discussed in detail in chapter 4. The information field structure is given below:

Telemetry Transfer Frame Secondary Header (32 bits)						Data	Telemetry Transfer Frame Trailer			
Frame Identification			Master Frame Count	Virtual Channel Frame Count	First Header Pointer		Frame Status			Time
Version Number	Virtual Channel ID	Spare					Time Flag	Spare	TC count	
2	3	3	8	8	8	0 – 2008	4	2	2	0 – 64

Figure 3.3: AX.25 Telemetry Header and Trailer

Secondary Header (32 bits)

The secondary header is 32 bits long. It provides support for virtual channels and large data transfer (which will be discussed in detail later). The important fields are :

- **Frame Identification**
  - Version Number (2 bits) : Fixed to zero (00).
  - Virtual Channel ID (3 bits) : Supports upto 8 virtual channels. Eg. 111 - represents data belongs to channel 8.

  - Spare (3 bits) : Reserved for future use. Value set to 000.

- **Master Frame Count (8 bits)** : Total frame counter , ie., irrespective of the virtual channel. (modulo 256 counter).
- **Virtual Channel Frame Count (8 bits)** : Counter for a particular virtual channel (which corresponds to the current channel) . (modulo 256 counter).
- **First Header Pointer (8 bits)** : Specifies the octet number within the data field that contains the first octet of the first packet header.This allows the reassembly of packets even when previous packets were lost. It has value 11111111(0xFF) if no packet header starts in the frame.

Information Field(0 - 2008 bits)

CCSDS packets are carried by the information field. It has a maximum length of 2008 bits.

Trailer (0 to 72 bits)

- **Frame Status (8 bits)**
  - **Time flag (4 bits) :** Gives size of time field. 000 indicates no time field. If bit 0 is set to 1, bits 1 to 3 indicate the size of time field in octets + 1.
  - **Spare (2 bits) :** Reserved for future application. Set to 00.
  - **TC Counter (2 bits) :** No longer used in the modified version. Set to 00.
- **Time (0 - 64 bits) :** On Board time . Represents the time at which end flag of previous frame was transmitted.

3.2.4 Acknowledgement Frame

As mentioned earlier, a frame is identified as acknowledgement frame by the value 0x03 in the protocol identifier field. All other fields like source and destination addresses, control bits etc are same as any other AX.25 frame. Individual packet numbers (acknowledgments), each a byte long are placed sequentially in

the informtaion field. An acknowledgement frame does not contain a counter (for telecommand) or a telemetry header and trailer.

CHAPTER 4

DESIGN OF TMTC FRONTEND

The TMTC Frontend consist roughly of four major sub modules.

- AX.25 Packet encoding/decoding
- TC Transmitter
- TM Receiver
- Replay Controller

4.1 AX.25 Packet encoding / decoding

The frame structures of modified AX.25 protocol used by IITMSAT was discussed in the previous chapter. This module implements a library to support encoding /decoding modified AX.25 protocol. A class is defined for the AX.25 frame which implements the common header. From this class, two classes are inherited for telecommand and telemetry frames. These classes implement fields specific to telecommand and telemetry. An AX.25 frame object can be created by either specifying values of each field or passing a byte array which is parsed to fill in each field. AX.25 frame classes (both telecommand and telemetry) has functions which return byte array representation of the frame. Implementation details of this module are discussed in next chapter.

4.2 TC Transmitter

The telecommand transmitter is based on a state machine.Initially it was decided to use to the SwissCube transmitter algorithm. However it was quite limited for our requirements.

4.2.1 Limitations of Swiss cube design

The amount of telecommand packet transmission expected by SwissCube is lesser than what is expected by IITMSAT. The following features of SwissCube telecommand transmitter acts as a hindrance for adopting it in IITMSAT.

- It can support only 4 outstanding packets.
- Packets are transmitted one at a time.
- Transmission of packets are manually controlled by the user through MCS.
- There was no option of resending at TMTC Frontend layer.
- In the event of packet drop, the information is carried back to the application level, where a human user decides to resend the packet again. The manual control, lack of automated resending and support for fewer outstanding packets can severely restrict the amount of telecomand which can be transmitted efficiently.

4.2.2 Modified Design

The limitations mentioned above were not really a hindrance for operation of SwissCube. However, since IITMSAT expects more data transfer, it can be a serious concern. It slow down the number of frames transmitted . This is especially important as IITMSAT plans to have only one ground station. This severely

restricts the amount of time a communication link can be opened with the satellite as it comes into the field of view for a short time every day.Hence we modified the design to support more outstanding packets and support for automated resending of frames at TMTC layer. The main characteristics are :

- All the packets in the ready queue are dispatched at the same time one after another. The number of packets is expected to be in the order of 10, which is way less than the 256 outstanding packets allowed.
- Transmitter is half duplex. So we are not implementing an explicit timeout. Acknowledgement for a frame sent in a transmitter period is expected to come in the immediate next receiver period.
- If acknowledgement is not received in the immediate next reception, packet is resent.A packet will be resent a fixed number of times, after which packet drop will be announced to MCS.

The states are :

- READY : Indicates that the transmitter is on and is ready to transmit frames. Positive beacon is the trigger to transmit frames.
- WAIT_FOR_ACK : Indicates that the transmitter state machine is waiting for acknowledgement.
- WAIT_FOR_TRANS : Indicates that the transmitter state machine is waiting for the physical transmitter to switch on.

The external triggers are :

- Transmitter ON : Indicates switch to transmission mode.
- Transmitter OFF : Indicates switch to reception mode.
- Positive Beacon : Indicates that satellite is in field of view and ready for reception.
- Ack received : Indicates the reception of acknowledgement packet by receiver.

The state diagram is as given below :

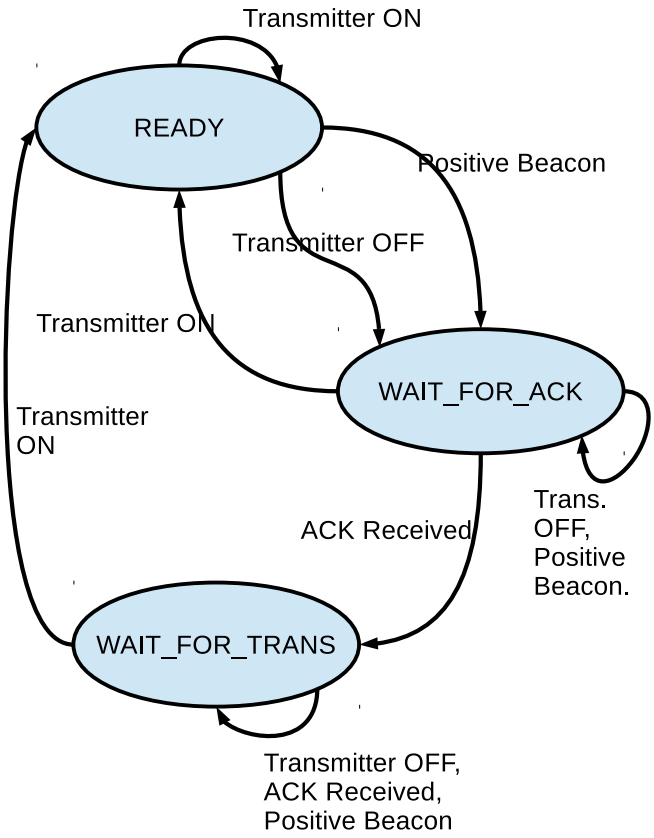


Figure 4.1: TC Transmitter State Diagram

When the transmitter state machine is in ready state, it is switched on and the a positive beacon trigger indicates that satellite is ready for reception. On positive beacon signal, all frames including acknowledgement, frames which are being retransmitted and new frames are dispatched to GNU Radio for transmission over RF link. The state then changes to WAIT_FOR_ACK, which indicates that the transmitter is waiting for acknowledgement. If in ready state , the transmitter is

switched off, the state changes again to WAIT_FOR_ACK. But during this case, unlike positive beacon transition, no operations are done.

When the transmitter state machine is in WAIT_FOR_ACK state and acknowledgement is received, frames for which acknowledgment is received is removed from the outstanding packet list. The state changes to WAIT_FOR_TRANS, which indicates waiting for transmitter to switch on.If the transmitter is switched on when the state machine is in WAIT_FOR_ACK state, all outstanding packets are assumed to be dropped and hence added to the resend queue. The state changes to READY.

When the transmitter is switched on and the state machine is in WAIT_FOR_TRANS state, the state changes to READY.

Besides this, the transmitter archives all AX.25 frames in a SQL database with timestamp.

4.3 Telemetry Receiver

TM Receiver receives the telemetry from GS hardware (or in IITMSAT case , SDR).Before going into details of how receiver processes it , a few important features supported by the IITMSAT downlink are discussed below.

4.3.1 Virtual Channels

There are various subsystems on board the satellite which needs to communicate with GS, like Housekeeping, payload etc. All these subsystems require a com-

munication link with GS, but only one physical data channel exists. The idea of virtual channels alleviate this problem.

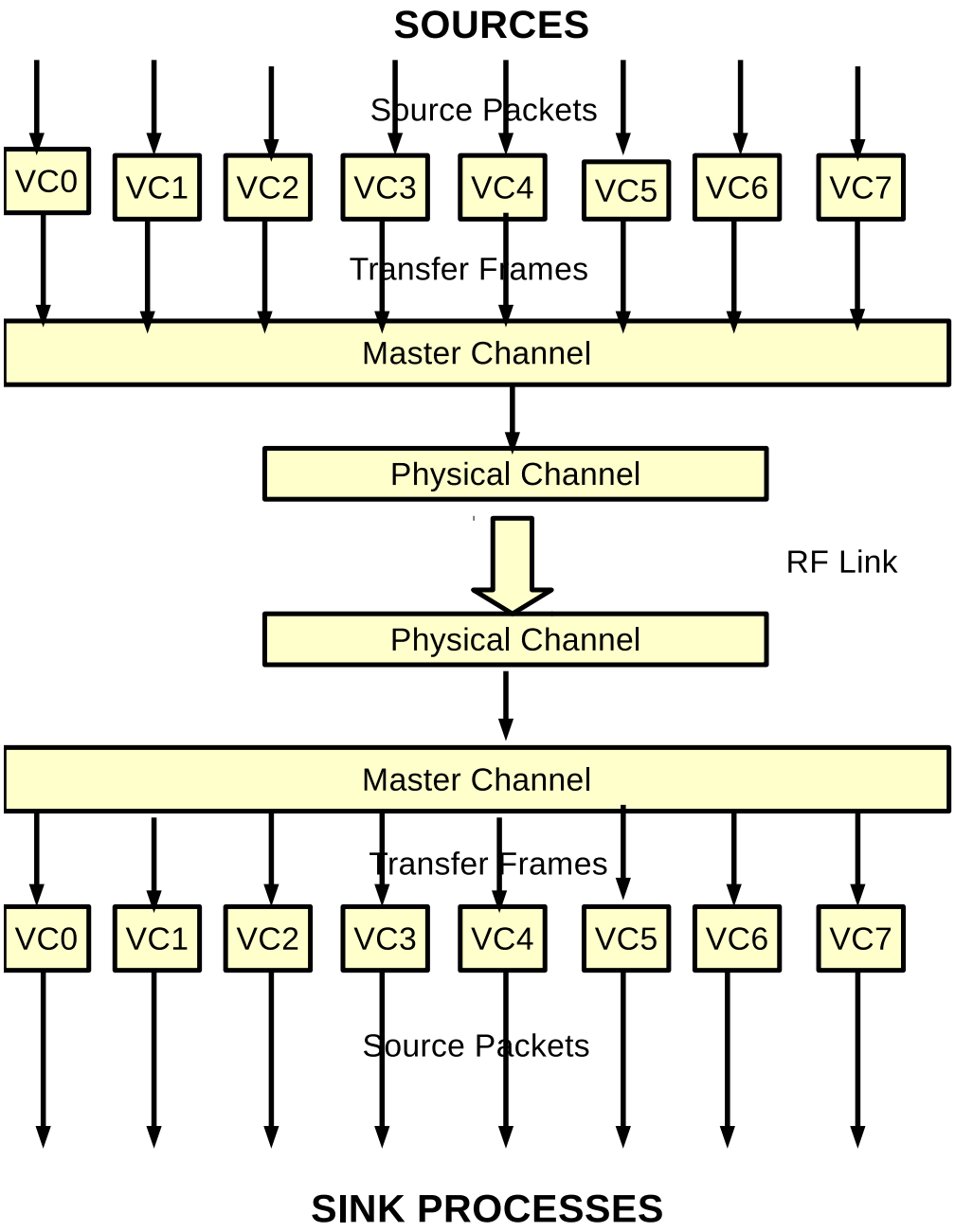


Figure 4.2: Virtual Channel. Adapted from [2]

Each subsystem is assigned a virtual channel to communicate with ground

station and these channels are multiplexed together into the same physical channel. Each frame has 3 bits to indicate which virtual channel the data belongs to. Thus it can support up to 8 virtual channels . It gives an indication on how to process/forward the data at GS . Also it gives all subsystems a fair chance of communication with GS. In the absence of virtual channels , a heavy subsystem like payload may occupy the master channel for most part, giving data light subsystems very less chance for communicating with ground station. Virtual channels ensure that all subsystems get a fair chance. Satellite just sends a packet with empty information field in case there is no data to be transmitted on that virtual channel in that particular slot. As mentioned in previous chapter, the telemetry secondary header supports virtual channels.

4.3.2 Large Data Transfer and Reassembly Unit

In case of telecommand ,it is quite possible to have commands of fixed size and it can usually be carried by the information field of a single AX.25 frame. This is not the case with telemetry. Though we might be able to fix the length of standard services, it is not possible in case of payload data. Payload data length can vary over a large range. Splitting payload data at application level is not a good idea as it will lead to cumbersome processing later at MCS. This is the case with large payload packets. On the other side, payload packets could be quite small in size and more than one packet could fit into the information field of one AX.25 frame. Transmitting them separately adds the header overhead to both the packets and it is a waste of resources to do so. It would be better to fit in both of them into same AX.25 frame. Large Data Transfer Services help in solving these issues.The idea

was proposed in [3].

In large data transfer service, all the packets to be transmitted are sequentially attached one after another.This creates a single service data unit. This data unit is then split into fixed size (usually the size of information field of link layer protocol) and each such unit is transmitted in each frame.

Our design implements this service with the help of first header pointer in secondary header of AX.25 telemetry. This field indicates the octet in data field where the first byte of the header of first packet resides. Header of subsequent packets can be obtained after finding the length of each packet(We assume CCSDS packets and read the length from the predefined length field). First header pointer helps in recovery of CCSDS packets even when the previous packets are lost.

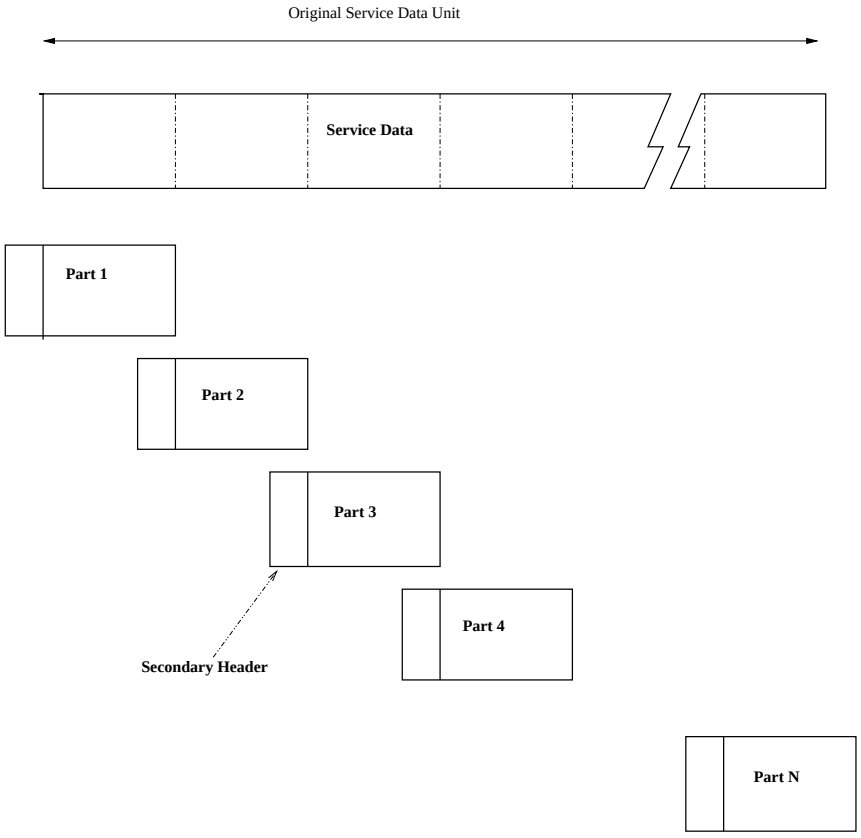


Figure 4.3: Large Data Transfer Service. Adapted from [3]

Reassembly Unit

Reassembly unit is responsible for reassembly of CCSDS packets from packet fragments received by the receiver at GS. Corresponding to each virtual channel, there is a reassembly unit.

4.3.3 Overall design of TM Reeciver

Processing steps after reception of an AX.25 telemetry frame in that order is as follows.

1. Receive AX.25 frame.
2. CRC check to detect frame bit errors.
3. Check the GS and Satellite SSID and Callsign.
4. If the frame is an acknowledgement frame, trigger Acknowledgement received of receiver.
5. Otherwise Archive the raw frame in database .
6. Forward the frame to appropriate virtual channel.
7. Transfer the frame to reassembly unit of the virtual channel.
8. After a CCSDS packet is completely reassembled, forward it to the MCS on appropriate port or channel.

4.4 Replay Controller

Replay controller lets the user replay archived AX.25 raw frames. It provides a small GUI where user can enter the virtual channel ID and the time range of in which the packets he/she wants to replay was received by the receiver. Replay

controller reads all frames in the range for that particular virtual channel from the database. It then creates a reassembly unit and forwards the frames one by one to the unit. Upon completion packets are forwarded to MCS as usual.

CHAPTER 5

IMPLEMENTATION DETAILS

This chapter discusses the implementation details of TMTC Frontend.

5.1 Development Tools and Environment

- **Development Language :** Java 1.6.0
- **Development Environment :** Eclipse IDE
- **Database server :** MySQL 5.5.35
- **Operating System :** Ubuntu 12.04 (or any other OS which supports Java 1.6.0 and MySQL 5.5.35 .)

5.2 Development phases

The development process can be divided into four major phases.

1. Implementation of modified AX.25 frame encoding/decoding.
2. Implementation of TC Transmitter.
3. Implementation of TM Receiver (including reassembly unit).
4. Implementation of Replay Controller.

Besides these, another important module was implementation of SQL archiving support. Details of various classes and methods are described below.

5.3 AX.25 encoding/decoding

This module implements the functionality to encode and decode modified AX.25 frames. It is implemented in the package **AX25**. This package implements all the features of the modified AX.25 frames which IITMSAT uses.The packages contains a main AX.25 Frame class and separate classes for telecommand and telemetry. The classes in the package are described below.

5.3.1 AX25AddressField

This class handles the address fields in the AX.25 frame header. As explained in chapter 3, both source and destination addresses have same format.

Fields

- CallSign : A Java String used to store 6 letter callsign field of address.
- Ssid : A byte to store the secondary station identifier field of address.

Methods

- ToByteArray() : Returns the address field value as a byte array.

Objects of this class can be constructed by either providing values of each field or passing a byte array representing the field. The byte array will be parsed to obtain values of the field. Objects of this class are used as fields within AX.25 Frame class.

5.3.2 AX25FrameIdentification

This class implements the frame identification field which is a part of telemetry secondary header.

Fields

- VersionNumber : A byte to store the version number field.It has a fixed value of 0.
- SpareBits : A byte to store the spare bits field.It has a fixed value of 0.
- VirtualChannelID : A byte indicating the virtual channel ID.

Methods

- ToByteArray() : Return the frame identification field as a byte array.

Objects of this class can be constructed by either providing the virtual channel ID or passing a byte array representing the field.

5.3.3 AX25FrameStatus

This class implements the frame status field which is a part of telemetry trailer.

Fields

- SpareBits : A byte to store the spare bits field.It has a fixed value of 0.
- TimeFlag : A byte to store time flag field.
- TCCounter : A byte to store telecommand counter value.

Methods

- ToByteArray() : Returns the frame identification field as a byte array.
- getTimeLength() : Returns the length of the time field.

Objects of this class can be constructed by either providing the virtual channel ID or passing a byte array representing the field.

5.3.4 AX25Frame

This class implements the header part which is common for both telecommand and telemetry frames. The fields of this header was discussed in chapter 3.

Fields

- DstAddress : An object of the class AX25AddressField, used to store destination address.
- SrcAddress : An object of the class AX25AddressField, used to store source address.
- ControlBits : A byte representing the control bits field of the header. It has a fixed value of 0x03.
- ProtocolIdentifier : A byte representing the protocol identifier field. As mentioned earlier, this field is used to distinguish if a frame is an acknowledgment frame.
- _informationField : A byte array to store the information field of the frame.

Methods

- ToByteArray() : Returns the AX.25 frame as a byte array.
- ToByteArrayWithoutCRC() : Returns the byte array representation without CRC field.

Again constructors are provided for constructing AX.25 frames from values of each field and byte array representing the entire frame.

5.3.5 AX25Telecommand

This class inherits from the AX25Frame class. It implements features specific to telecommand, which in IITMSAT case is just the Master Frame Count. This frame just has one extra field TCCounter. It is actually the value of first byte of information field. All other methods are also same as the super class AX25Frame.

5.3.6 AX25Telemetry

This class inherits from the AX25Frame class. It implements features specific to telemetry, which are the secondary header and trailer in the information field (discussed in chapter 3).While all methods are same as AX25Frame class, there are few additional fields.

Fields

- FrameIdentification : An object of the AX25FrameIdentification class, used to represent the frame identification field.
- MasterFrameCount : A byte to store master channel frame count.
- VirtualChannelFrameCount : A byte to store virtual channel frame count.
- FirstHeaderPointer : A byte to store the first header pointer, which is used to support large data transfer.
- FrameStatus : An object of the AX25FrameStatus class, used to represent the frame status field.
- time : A field of data type long, used to store value of time field.

5.4 Telecommand Transmitter

This module implements the TC transmitter logic (state machine discussed in chapter 3). This module is implemented in the package **TCTransmitter**.It is responsible for handling transmission of telecommand frames including acknowledgments and resending of frames. The package consists of the following.

5.4.1 State

This is a java enum used represent the state of the transmitter state machine. This enum can take three values corresponding to each of the states mentioned in chapter 4 on design. The values State enum can take are :

- READY
- WAIT_FOR_ACK
- WAIT_FOR_TRANS

5.4.2 TCTransmitter

This class implements the transmitter state machine logic. The important fields and methods of this class are discussed below.

Fields

- EncodedPacketQueue : A java LinkedBlockingQueue of AX25Telecommand objects. This queue is used to store AX.25 frames, which are obtained after encoding CCSDS packets received from MCS.
- ResendPacketQueue : A java LinkedBlockingQueue of AX25Telecommand objects. This queue holds AX.25 frames which are to be retransmitted in the next transmission period.

- OutStandingPacketMap : A java HashMap storing outstanding frames, hashed by frame counter.
- currentAck : An acknowledgement frame which is to be transmitted in the next transmission period.
- state : A java enum representing the state of transmitter state machine.
- _sqlClient : An object of SQLClient class, used for archive functionality.

Methods

- receivePacketTCTransmitter() : This function receives the application data packets from MCS and calls a function to encode the data into AX.25 telecommand frames.
- ProtocolEncoding() : This value takes an application data object as its argument and encodes it into AX.25 telecommand frames.It adds the encoded frames to EncodedPacketQueue.
- dispatchPackets() : This function is called when positive beacon signal is received in READY state. It transmits frames to satellite by dropping them to GNU Radio. It first transmits acknowledgement frame, followed by frames from ResendPacketQueue and EncodedPacketQueue.
- positiveBeacon() : Trigger to indicate reception of positive beacon signal.
- receiveAck() : Trigger to indicate reception of acknowledgement frame by TM receiver. Acknowledgments are passed as an argument to this function.
- TransmitterOn() : Trigger to indicate switching on of the transmitter.
- TransmitterOFF() : Trigger to indicate switching off of the the transmitter.

These fields and methods together implements the transmitter state machine described in chapter 4.

5.5 Telemetry Receiver

This modules implements the telemetry reception part of TMTC Frontend. The design of telemetry receiver was discussed in chapter 4. Reassembly unit forms

an important part of it. TM receiver is implemented in the package **TMReceiver**. There are two important classes.

5.5.1 ReAssemblyUnit

This class implements the reassembly unit, which helps in reassembly of CCSDS packets from AX.25 telemetry frames. This class consists of a list of completed packets, an integer indicating the number of reassembled packets and another to store the virtual channel ID. There is just a single method in this class, ReassemblePacket(). AX.25 telemetry frames are passed to this method as an argument. This method implements the logic behind reassembly of CCSDS packets from one or more AX.25 telemetry frames. After a CCSDS packet is reassembled from one or more AX.25 telemetry frames, it is added to the completed queue of that particular reassembly unit.

5.5.2 TMReceiver

This class handles the operations of telemetry receiver. It processes the telemetry frames which are received from the satellite and forwards them to appropriate virtual channels. The important fields and methods of this class are discussed below.

Fields

- DecodedPackets : A Java LinkedList used to store decoded telemetry frames.
- ReceivedPacketCounter : A Java LinkedBlockingQueue used to store counters of received packets.

- ReAssemblyUnitList : A Java ArrayList of 8 reassembly units , one corresponding to each virtual channel.
- receivedFrames : A Java LinkedBlockingQueue of all byte arrays received from GS.
- _sqlClient : For archiving support.

Methods

- acceptPacket() : This method parses a byte array to form an AX25Telemetry object.
- ReceiverController() : This method reads packets from the receivedFrames queue in an infinite loop, checks CRC and then decodes the frame. It then matches the source and destination addresses and archives the frame. If the frame is an acknowledgement frame, flags are set to pass message of acknowledgement reception to transmitter. Otherwise frames are passed to corresponding virtual channels.
- start() : Starts the receiver controller operations in a thread.
- SplitToVirtualChannel() : This method splits the frames to the virtual channels and passes the frames to corresponding virtual channels.

The constructor for the class creates 8 reassembly units and constructs other objects of the class.

5.6 Replay Controller

Replay controller as described earlier provides the functionality for replaying archived telemetry packets. The logic behind replay controller was explained in previous chapter. It is implemented in the package ReplayController. It has just a single class, ReplayController. It has a small UI , which lets the user enter VCID, start and end timestamps. Other than GUI related fields, it has an SQLClient object

for handling reading from SQL. The

The important methods are discuss

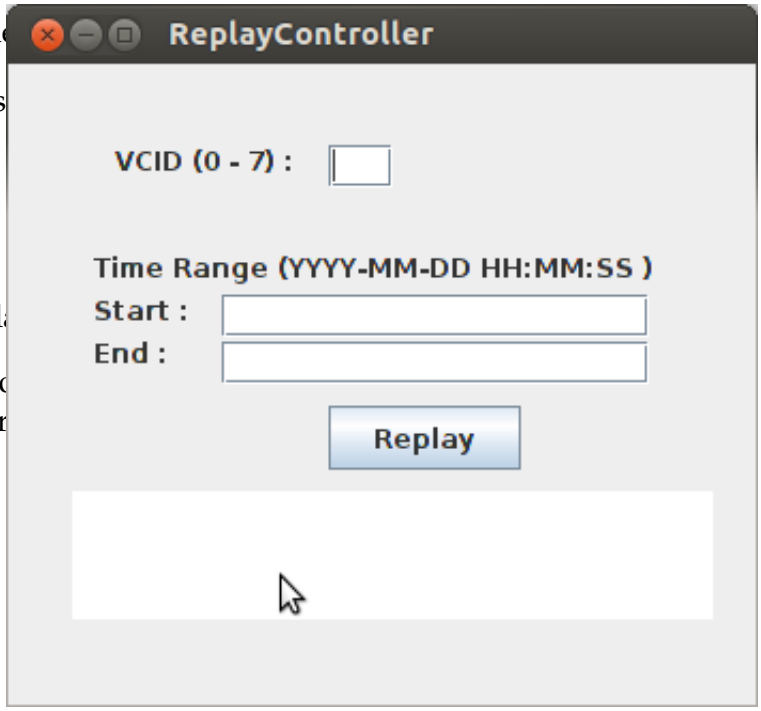


Figure 5.1: Replay Controller GUI

Methods

- Display() : This method displ
- processRequest() : This metho click. It retrieves the frames fr unit and passes the frames to

5.7 SQL Client

SQL archiving support for frames are provided by the SQLClient class of the package **SQLCLient**. The class connects to a mysql database through a jdbc connection. The important methods of the class are mentioned below.

Methods

- ArchiveAX25TeleCommand() : This method takes in an AX25Telecommand frame as the argument and writes it into the AX25Telecommand table of the SQL database.
- ArchiveAX25TeleMetry() : This method takes in an AX25Telemetry frame as the argument and writes it into the AX25Telemetry table of the SQL database.
- retrieveAX25Telemetry() : This methods takes in two timestamps as its arguments and retrieves a list of frames which falls into that time range.

5.8 TMTC Frontend

The package **TMTCFrontEnd** implements the final TMTC Frontend module. It integrates all the sub modules like TC Transmitter, TM Receiver etc, handling communication between them and controlling the overall operations of the entire TMTC Frontend. The package consists of a class MissionConstants, which stores various constant values associated with the module like addresses of the satellite and ground station, number of resend attempts etc. The main class which intergrated other moduels together is FrontEnd.

5.8.1 FrontEnd

This class implements the logic behind the functioning of TMTC FrontEnd. It integrates different modules of the TMTC Frontend. It consists of aan object of the class TC Transmitter and another of the class TM Receiver. It also contains a replay controller object. The important methods of the class are discussed below.

Methods

- Start() : This method starts the operations of the TMTC Frontend. It starts various thread associated with the system. The various thread are MCSListener, GSListener,ControllerThread, TransmitterThread and ReplayControllerThread.
- ListenToGS() : This method listens to the ground station and receives packets from the ground station. It listens to ground station continuously in a thread (GSListener).
- ListenToMCS() : This method listens to the MCS and receives packets from the MCS. It listens to MCS continuously in a thread (MCSListener).
- ControlOperations() : This method controls the main operations of the TMTC Frontend. It is responsible handling communication between receiver and

transmitter. This is the method which the ControllerThread runs continuously.

- TransmitterSwitch() : This method switches the TMTC Frontend between transmission and reception modes after fixed time intervals. This switch happens continuously in a thread (TransmitterThread).

CHAPTER 6

TESTING

This chapter discusses the testing carried out on the TMTC Frontend module. Unit tests were carried out on different sub modules. The entire software module was tested with the help of GS and MCS simulators.

6.1 Unit Testing

This section discusses the unit tests carried out on various sub modules or units.

6.1.1 AX.25 Telecommand

Test to ensure that AX.25 telecommand encoding/decoding is working properly.

Procedure

1. Create an object of the class AX25Telecommand by providing values for each field.
2. Obtain the byte array representation of the frame.
3. Create another object of the class AX25Telecommand, this time using the constructor which takes in a byte array as the argument and parses it to get the values of each field.
4. Compare the values of the fields of the new object with that of the original one.

Comments

Matching of values of all the fields indicates that AX.25 Telecommand encoding/decoding is working properly.

Result

Test status : Success.

6.1.2 AX.25 Telemetry

Test to ensure that AX.25 telemetry encoding/decoding is working properly.

Procedure

1. Create an object of the class AX25Telemetry by providing values for each field.
2. Obtain the byte array representation of the frame.
3. Create another object of the class AX25Telemetry, this time using the constructor which takes in a byte array as the argument and parses it to get the values of each field.
4. Compare the values of the fields of the new object with that of the original one.

Comments

Matching of values of all the fields indicates that AX.25 Telemetry encoding/decoding is working properly.

Result

Test status : Success.

6.1.3 Replay Controller

Test to ensure that Replay Controller functionality is working properly

Procedure

1. Make entries in AX25Telemetry table in the date base for one particular virtual channel. Note the timestamps.
2. Create an object of ReplayController class and call Display() function to view the GUI.
3. Enter the virtual channel ID and start and end timestamps for the entries which were made in the database and press Replay button.
4. Observe the text box in the GUI to see if expected packets are reassembled.

Comments

Reassembly of expected number of packets indicate that replay controller is working properly.

Result

Test status : Success.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Current State

With the available requirements and specifications, TMTC Frontend for IITMSAT was developed in Java. All specifications up to now are implemented. Transmitter was developed according to the current expected scenario. Similar is the case with receiver. Replay Controller functionality was also implemented. A small GUI was also created for the replay controller. With the available details and specifications, the system was tested and was found to be working.

7.2 Future tasks

- Run more tests by creating MCS and GS simulators as and when more specifications about their working are available.
- Discuss and incorporate changes suggested by ISRO panel during design review .
- Decide on the internal time representation.
- Implement time correlation report.
- Integrating with MCS and GNU Radio (GS) once they are completed .
- Testing the complete GS software system after integration. This includes load testing over a period (for example : one week).

APPENDIX A

ASSUMPTIONS

Various assumptions made during the development of TMTC Frontend are :

- Beacon decoder is not a part of TMTC Frontend. It is assumed that beacon is decoded separately , analysed and TMTC Frontend will be triggered on positive beacon.
- Connection between ground station and satellite is half duplex. The system is almost fixed to be half duplex. As of now switch between transmitter and receiver happens after fixed time in TMTC Frontend. It can be modified easily to trigger the switch between two externally.
- SDR (in this case GNU radio) is responsible for bit stuffing and adding flags at the end of each frame.
- TMTC Frontend receives individual frames from SDR and not a continuous sequence of frames.
- Communication with TMTC Frontend can be through simple sockets or more advanced message queues. The function which communicate with other modules are left open. When a communication method is decided, it can be implemented by editing those functions without any other changes.
- Various parameters like maximum resend count, GS and satellite address etc are stored in MissionConstants class. Default values are assumed, which can be changed easily by changing the values in the class.
- MySQL is the database used. It is assumed to be installed on the machine. A seperate setup script will be provided, which will create database and add necessary tables. This has to be run once at the beginning on a new machine .

REFERENCES

[1] Beech, William A., Douglas E. Nielsen, and Jack Taylor. AX.25 Link Access Protocol for Amateur Packet Radio , 2.2nd ed. Tucson, Arizona: Tucson Amateur Packet Radio Corporation,July 1998.

[2] Packet Telemetry. Recommendation for Space Data System Standards, CCSDS 102.0-B-5. Blue Book. Issue 5. Washington, D.C.: CCSDS, November 2000.

[3] Ground systems and operations Telemetry and telecommand packet utilization,ECSS–E–70–41A. Noordwijk,The Netherlands.:ESA Publications Division,January 2003.

[4] Yann Voumard. TM/TC Front End Phase C,S3-C-S E-1-3-TMTC Front End Report .Issue 1.Noordwijk,The Netherlands,ESA,June 2008.

[5] Florian George and Benoit Cosandier. SwissCube Ground Segment Phase B,S3-B-S E-1-5-Ground Segment.Issue 1.Noordwijk,The Netherlands,ESA ,June 2008.