# SOFTWARE DESIGN OF STORE AND FORWARD PAYLOAD FOR A MICROSATELLITE

**V.Vaidehi, S.Muthuselvam, R.Srinivasan, Praveen Kumar, V.Nirmal Kumar**
**Dept. of Electronics Engineering, Madras Institute of Technology, Anna University, Chennai-600044, India**

**Abstract**

Store and forward via satellite is an esteemed method of communication. This paper focuses on the design of software modules for store and forward payload for a polar microsatellite developed by Anna University, the ANUSAT. This paper describes several aspects of software design for Store and Forward payload, which includes memory manager, file manager, scheduler, I/O handler and Data Link layer protocol modules. This paper also discusses the use of a multi tasking real time kernel for task scheduling and development of mail client software for the ground users.

## 1. Introduction

ANUSAT is a polar micro satellite developed at Anna University. The primary payload of ANUSAT is store and forward, used to transfer electronic mail among ground terminals in the footprint of the satellite. In such a system the originating ground station sends a message to the LEO satellite, the satellite stores the message in an on-board storage system, and delivers this to the destination ground station in a later time. If source and destination terminals are situated within the footprint at the same time, then message delivery can be made almost immediate. But if they are not in the same footprint then message delivery time can vary from a several minutes to hours depending on user's location and satellite's orbit. Between the storage and the retrieval of the message, the LEO satellite moves around its orbit and the Earth rotates on its axis. These movements change the location of the satellite's footprint, and the satellite effectively carries the message from one station to the other. On a successful transfer, the mail is deleted from the memory. If not, it is retained on the memory to provide it again at the next visibility cycle. It should be noted that the source and destination users can be anywhere on the earth provided they are in the foot print of the satellite. The significance of this method is that it is one of the best modes for non real-time communication of information. Applications which can tolerate delay, but which require inexpensive global networking are well-served by this system. It provides a communication link which would be otherwise very expensive or impossible to establish. It enables to form a store and forward network which can be assembled quickly and inexpensively. It is particularly attractive in areas in which conventional terrestrial communication systems were not available. Some typical contexts are communicating with military units abroad, geographical expedition teams, disaster relief teams, etc.

In this paper section 2 introduces the protocol stack of ANUSAT. Section 3 briefs the various Modules of S & F Payload Software. Section 4 presents the memory management scheme used in the design. Section 5 discusses the results.

## 2. Protocol Stack of ANUSAT

ANUSAT protocol suite is specially designed for highly transparent error free transmission. The protocol suite is looked as a three layer network model. The block diagram of the layer model is given in the figure 1.It is necessary to look at the overview of various layers before going into the design of the S & F payload.

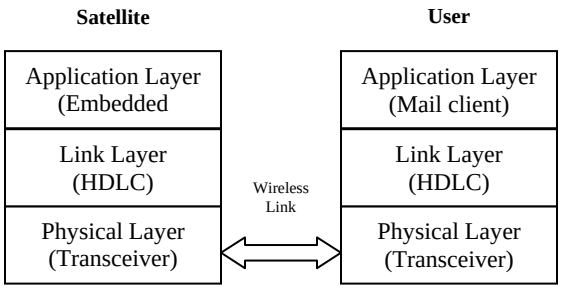


**Figure 1 Laver model**

### 2.1 Physical Layer

The physical layer is the layer which takes care of the communication medium, mode of communication, speed, etc. The communication medium used is free space. The mode of communication is full-duplex. The raw data rate is 9.6 kbps. The modulation scheme used is FSK.

### 2.2 Link Layer

The link layer is responsible for fragmentation of data, flow control, error control, etc. on a point to point basis. The flow control and error control is achieved through sliding window technique and Go-back-N ARQ respectively [2]. The medium access control protocols adopted at this layer are polling and reservation scheme. The link protocol used at this layer is HDLC.

### 2.3 Application Layer

The functionality of all the above layers is same for both satellite and ground terminals. But the application layer behaves differently at the satellite and the ground terminals side. At the satellite, the application layer has embedded software to handle the operations concerned with store and forward. In the user side, the mail client application running on a PC enables the user to send and receive mails between the satellite and user.

## 3. Modules of S & F Payload Software

The software for S & F Payload is alienated into two modules, the data link layer module and the other is the Application layer module. The data link layer module is responsible for data transfer, flow control and error control between the satellite and the users. The application layer module is responsible to perform the following tasks such as memory management, file management, scheduling, and I/O handling in a multitasking environment.

### 3.1 Data link Layer Design

The data link layer design consists of a MAC layer and a logical link control layer. The MAC layer in the design allows to access the link on a reservation based polling scheme. Where as the logical link layer is organized into three different phases they are Directory Broadcast Phase, Allocation Phase, Data transfer Phase. The data link layer protocol that is adopted for the design is HDLC [5]. The communication from the user to the satellite and vise versa is through exchange of frames organized according to the different phases of operation.

#### 3.1.1 Directory Broadcast Phase

Once the satellite comes to the visibility of a group of users, a directory broadcast frame (DBF) frame is broadcasted to all users. This frame informs the mail status of the users in the on-board memory. Each bit in the information field of the DBF frame is allotted for a single user. By setting or resetting this single bit the design informs the user about their mail status on – board.

#### 3.1.2 Service Request Phase

After receiving the DBF, the users have to send a request frame to request for service. The uplink is shared between multiple ground terminals in the same footprint, so the satellite polls individual users in the footprint to receive their request. In response to the poll each user will send a request frame. The request frame content will inform the satellite whether the user has only mails to be received or have only mails to be sent or have mails to both send and receive or neither send nor receive mails. It
also gives the information about the file size the user wants to send. This information will be used in the satellite for schedule preparation in the reservation phase.

#### 3.1.3 Reservation Phase

In the reservation phase a schedule is prepared based on the information available in the satellite and information collected from the Request frames received from contenting users. Following preferences are taken into account while preparing the schedule; the ground station is given the highest priority. The user with both mails to send & receive will be given the next highest priority. The mail size to be sent from the satellite is the next priority parameter as more free space can be obtained on – board. The visibility period is the next priority parameter. The user with only mail to send will be given least priority.

#### 3.1.4 Allocation Phase

In this phase the satellite sends an allocation frame with the user address for whom the time slot is allocated. On receiving the allocation frame the specific user responds with data transfer.

#### 3.1.5 Data transfer

The user who has received the allocation frame is allowed to use the channel for data transfer to and from the satellite. This phase continues either till the data to be transferred is completed or till the maximum allocated time per user is exceeded.

### 3.2 Process Scheduler

The various phases discussed above are all grouped into a set of tasks for real time scheduling. Tasks are scheduled on a preemptive priority based real time kernel using µcos – II RTOS [4]. Each task is assigned a priority ID.

The task with the lowest numbered ID is given the highest priority. Based on the above scheme the design assigns the priority as shown in Table 1.

The 'memory initialization' task is executed, if it is the first time of S&F being switched ON or when a 'reset memory'

**Table 1: Priority order**

| Name of the Task | Allotted Priority Values |
|---|---|
| Memory initialization | 0 |
| Send DBF | 1 |
| Schedule | 2 |
| Allocation | 3 |
| Data Transfer | 4 |
| Housekeep | 5 |

Telecommand (TC) is received. Then this task is deleted. So the next priority task 'SendDBF' is scheduled and complete the Directory Broadcast phase and get suspended by itself. The task 'Schedule' with priority 2 is invoked after SendDBF task gets suspended. After completion of schedule task, it delete by itself. Now the task 'Allocation' with priority 3 gets executed. After the completion of this task, it suspend by itself. So the next higher priority task, 'Data transfer' with priority 4 is scheduled.

When data transfer task completes execution, it resumes the Allocation task.

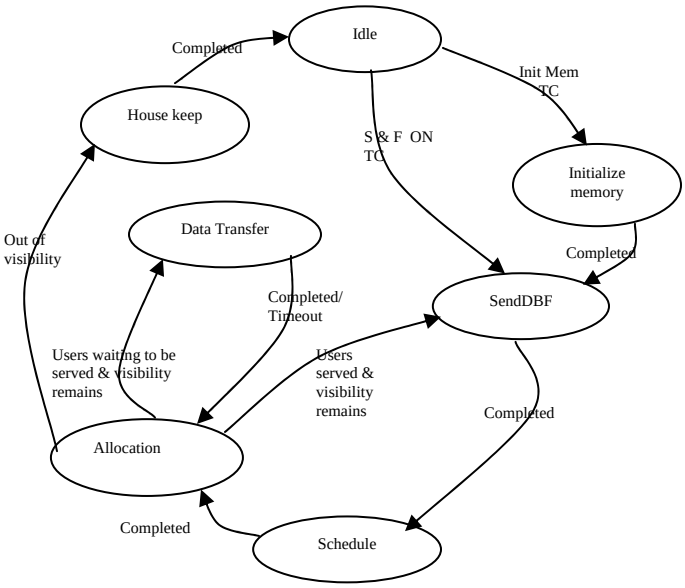


**Figure 2 Task Flow Graph**

The allocation task sends allocation frame if there is a user waiting for service and suspend itself to schedule the data transfer task. But if there is no users left and if the visibility time remains then it resumes the task SendDBF. Else if the satellite is out of visibility then it delete the SendDBF task and delete itself. Then the next higher priority task 'Housekeep' is scheduled, it construct the Telemetry Frames which carry the performance parameters of Store-and-forward payload. The system goes to the idle state when the S&F off telecommand is received.

## 4 Memory Management

The on board memory is divided into three major regions. They are File table area, free block manager area, Message storage area.

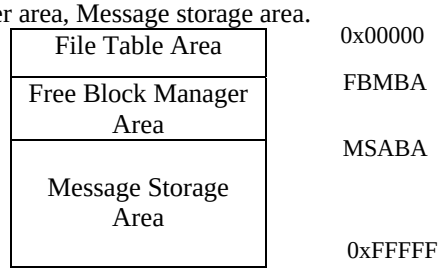


**Figure 3 On-Board Memory Split up**

### 4.1 User Addressing

In any type of network, there should be an addressing scheme which would `uniquely identify the users on the network. It is needed here also. ANUSAT will serve at the maximum 255 users on a whole. In ANUSAT 1 byte is used for addressing which allows 256 address spaces which are sufficient for addressing all the users in the network. Out of these 256 addresses, 2 addresses are used to convey special meaning as broadcast mail and ground control station. Address"11111111" i.e. 0x255 is allotted for broadcast address and the address"00000000" i.e. 0x00 is assigned for the ground control station. So, the addresses from 1 – 254 can be assigned to the users.

### 4.2 Block Access Method

The entire memory is divided into various zones as mentioned earlier. The message storage area is divided into blocks of equal size. The block size in this design is 256bytes. The simplest unit that can be accessed in the message storage area is a block. So, the addressing for the message storage area is in block addressing mode. It has to be noted that if actual addressing mode is used then an address would need 22bits. Instead, if we go for block addressing, then an address size is 16bits. This will help us to reduce the overhead

size. As a message may be more than 256 bytes in size, it may need multiple blocks to store that message. So, a method should be provided for accessing the files in multiple blocks. This design makes use of the linked list structure for this purpose. The first 254 bytes in the block is used to store the message file's segment and the last two bytes are used for chaining on to the next block. These two bytes store the address of the next block if any or a null address if it is the last block of that message file.

A translator is needed to translate the block number to physical address. The expression used by the translator is

$$PA = BN * 256 + MSABA \quad (1)$$

Where

PA is Physical Address.
BN is Block Number.
MSABA is Message Storage Area Base Address

This translator translates only the block addresses and an explicit offset value is needed to access any location in the memory. So, the offset value is maintained explicitly and summed with the physical address generated by the translator to get the exact location in the memory.

### 4.3 Bit Vector Technique

The bit vector technique is used for free block management which is one of the most efficient and successful scheme. In this method, each storage unit is mapped to a bit. The bit value is '1' if the storage unit is occupied and the bit value is '0' if the storage unit is free. In this design, the storage unit is a block. Each block has a corresponding usage status bit indicating the usage statistics of the block.

If there are N blocks in the message storage area, then the number of bytes needed for the free space manager area (M) is given by the formula

$$M = ceil\ (N/8) \quad (2)$$

Where *ceil* (x) is the smallest possible integer which is greater than or equal to x. The bits in the bit vector are addressed in normal addressing i.e. starting from 0, and they are in the order of LSB to MSB incremental. The addressing of bits is in a zigzag fashion. The main idea behind mapping is to find out the address of the bit corresponding to a block address and vice-versa. This job is done by the free block manager mapping unit. The block address (BA) corresponding to a bit in the bit vector is achieved by the following formula

$$BA = ByA*8 + BiA \quad (3)$$

Where

ByA is the byte address of the location of the bit from the FBMBA.

BiA is the bit address of the bit starting from the LSB.

Similarly, a reverse mapping unit is needed to reverse map a block address to a bit in the bit vector. The formula used for reverse mapping to find the bit address corresponding to a block address is given below.

$$ByA = floor\ (BA / 8) \quad (4)$$

$$BiA = BA \bmod 8 \quad (5)$$

Where, *floor* (x) is the greatest possible integer lesser than or equal to x.

x mod y is the remainder of the division of x by y.

### 4.4 Managing File Table

Any operating system which uses file should have a file table which stores the details of the files present in the disk. This design also needs a file table for this purpose. But it is different from the conventional operating systems in the sense that, in conventional operating systems, the files are stored in the secondary memory of the system as some magnetic disks or so. But in our case, the files are stored in the semiconductor memory.

A key assumption is made while designing the file table structure. It is that a user doesn't have more than twenty concurrent mails in the memory. This is decided by taking into account the average traffic for a user and also the transfer speed and visibility time. If this is wanted to be changed in future, the design is so flexible to accommodate the change into it.

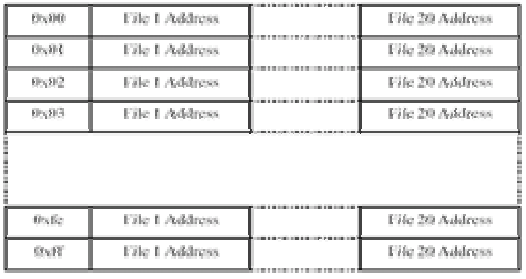


**Figure 4 File table Area**

The file table of this design is a fixed sized one. It is composed of file table entries. Each user has

a file table entry corresponding to him. Each entry is capable of holding 20 file addresses in block addressing mode. The first slot is of 1 byte size which stores the user address which the file table entry corresponds to. The next 20 consecutive slots are of size 2 bytes each which is used to store the block address of the first block where the file is being stored in the memory. If there is no file exists for the user, then a slot should store a null address which indicates that the slot doesn't store any file's address. As 0x0000 is a valid block address in this design, we can't use this as null address. But the address 0xffff is not possible in this design. So, this is used as the null address here. If a slot has 0xffff in it, then it is an empty slot and it can be used to store a new file's address. Each user has a separate file table entry. So, we need 256 file table entries each of size 41 bytes. It should be noted that there is an entry for the broadcast address also. So this makes the size of the file table area as 256 * 41 (10496) bytes.

Thereby each user's file table is accessed by using the formula UA*41. This will give the starting address of the file table entry. The address of the slots for files 1, 2, 3… 20 is given by UA*41+1, UA*41+3, UA*41+5… UA*41+19

The slots are used in a queue model. The first file address is stored in the first slot, second file address in second slot and so on. When a message arrives, for storing the block address of the file, the first free slot is found i.e. the slot with null address and the block address is stored in that slot. Similarly when deleting a file, the block address is overwritten by the block address of the next slot and the other slots are moved one step ahead. This makes that the first came message is available in the first slot and so on.

### 4.5 File Handling

When a user sends mail to other user or broadcasts a mail, a file creation is needed. File creation can be of two kinds based on the mail type. It is different for single destined messages and broadcast mails.

The file creation in this design falls into two categories they are normal file creation and broadcast file creation. Normal mails are the mails destined to a single user. The process of creating normal mails is given as follows. When a new mail arrives, the destination address of the mail is looked up from the mail. The first free block is found using the routines from the free block manager. The address of the first free block is added to the file table of the corresponding user. The method for adding a new file to the file table entry is as follows. The first slot for the given address is examined. If it is not null, then it implies that there is already one or more files are present for that user. Then the file table entry is traversed to find the first free slot i.e. slots with 0xffff and the file's address is stored in that slot. If no free slot exists for a given user, then an exception is raised. Once the file entry is made, then the mail is started to store in sequence. A counter is kept which is incremented for each byte transfer. Once the 254 bytes are stored in that block, a new block should be allocated and the new block's address is chained in the previous block's chaining bytes. This is repeated till the mail ends. If a new block can't be allocated in the meantime, then no space exception is raised. And also if the transfer is incomplete, then incomplete transfer exception is raised.

As mentioned the mails that are larger than 254 bytes in size are stored in multiple blocks and are kept track using the chaining bytes in the block. While allocating a new block, the chaining bytes are nullified by storing the null block address 0xffff to it. This helps us to avoid the garbage chaining if any value is already present in those bytes.

As discussed earlier, the broadcast mails are the mails that are destined to all the users in the satellite. So, a single copy of the mail is kept in the memory and entries are replicated in all users file tables. Hence, the procedures are almost same as for normal mail creation. The main difference is that for these mails, apart from adding an entry to 0x255, entries are made to all users. This makes that every user has reference to that mail and only one copy of the mail being physically existing in the memory. Files should be retrieved when the user requests them. In this design, the files can be retrieved only in the order that they are arrived. When a user requests his mail, the file table entry of the user is located and the first slot is examined. If it is not null, then it means that the user has one or more mails. The first slot's value gives the block address of the first mail. The physical address is got by using the address translator. The offset value is initialized to 0. Byte by byte transfer of data is done by incrementing the offset. Once the offset is reached 254, the next block is accessed from the chaining information. This is done till the entire message is accessed by looking for the EOF. File deletion is done when the mail is

successfully delivered to the destination and an acknowledgement is received. File deletion also falls under two categories. Deletion of normal files and deletion of broadcast files. When normal files are to be deleted, the entry in the file table as well as the bit in the free block manager corresponding to the message in the memory is deleted. To delete a mail, first all the blocks containing the mail are marked as free by traversing through the blocks. Then the entry is removed from the file table entry. Broadcast file deletion is same as normal file deletion except that the mail is not removed from the memory instead only the file table entry is removed. When a user x acknowledges a broadcast mail, only the entry for the user x is removed and the broadcast mail still remain in the memory.

## 5 Results and Discussions

The entire design is implemented in C, which is easier to migrate to other environment and also easy to maintain. The entire design is simulated in the personnel computer environment by using a portion of the RAM as the satellite's memory. The communication between the user and the satellite is achieved using the com ports available in the Personnel computer. An RTOS is used for achieving a multitasking environment and to ensure the timeliness of the system. All possible logical paths are exercised and the design is tested for stability. Both the satellite and the client are modeled as per the real-time specifications and a wired link is used as the communication medium.

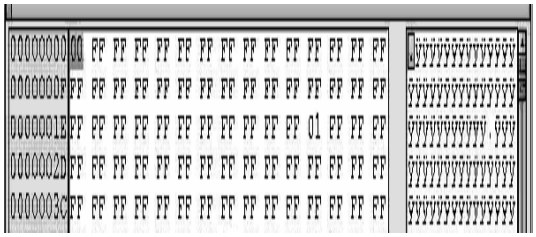

**Figure 5 File Table Area after Memory Initialization**

The figure 5 shows the file table area after initialization of the on board memory. The first byte (element) in the above figure has a value '00' indicating the address of user zero. The next consecutive 40 bytes are used as sizes of 2 bytes each to store the block number of the first block of each mail being stored in the memory. These locations are initialized to 0xFF FF to indicate that there is no mail for user '0'.Similarly '01' in the third row indicates the address of the user one and its mail status.

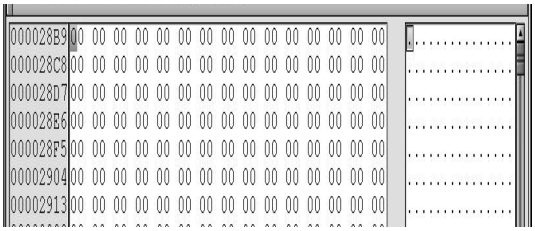

**Figure 6 Free Block Manager**

The figure 6 shows the memory dump of free block manager on memory initialization. The '00' in all the locations indicate that no memory block is occupied in the memory storage area.

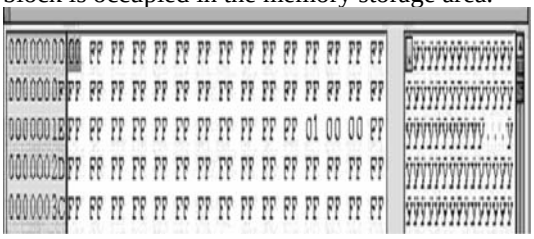

**Figure 7 File Table Area after storing a Mail**

The figure 7 shows the memory dump of file table area after saving a mail in the onboard memory for user 1. This can be viewed in the row with the starting address 0x0000001E. The values '0' in the next Consecutive 2 bytes indicate that the starting block number for the first mail of user 1 is 0.

The figure 8 shows the status of the free block manager after a mail is stored in the on board memory.

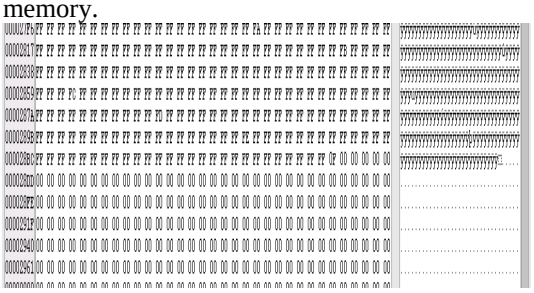

**Figure 8 Free Block Manager After Storing a Mail**

The row with the address 0x000028BC has an entry 0x0F (binary equivalent being 00001111) which indicates that first 4 blocks are occupied in the mail storage area for storing the mail received.

The figure 9 shows the mail storage area in the on board memory after storing the mail received. The row with the starting address 0x000030BA shows the starting of the mail storage in the hexadecimal format. The column in the right side of the figure 9 shows the exact mail content that is stored in the mail storage area of the satellite.

**Figure 9 Mail Storage Area after Storing a Mail**

The 14th and the 15th byte of the row with the address 0x000031C2 are the chaining bytes. The value shows the chaining

The figure 10 shows the status of the file table area after sending the mail for user 6 and a new mail received for user 7. After the mail has been received by user 6 the corresponding file table entry is reset to 0xFFFF. This can be seen in the row starting with the address 000000E7. Looking into the row with starting address 00000104 it is found that user 7 has a mail to receive and the starting block number is 4.

**Figure 10 File Table After Sending The Mail**

## Conclusion

In this paper the design and implementation issues of typical store-and-forward software system for ANUSAT has been outlined. This paper also provides explanation for using a simple yet powerful scheme for mail transfer, on board memory management and mail retrieval mechanism. The data link layer protocol of this design has adopted a reservation based MAC. The development team has made significant progress in developing a software design for Store and forward payload. This design balances simplicity with the performance required for University-based micro satellite mission. To date, the system has been implemented and functionalities are verified.


## Acknowledgements

Development of this software design for Store-and Forward Payload of a micro satellite has been made possible through sponsorship by the Indian Space Research Organisation (ISRO). Any opinions and conclusions or recommendations expressed in this materials are those of the authors and do not necessarily reflect the views of the ISRO. The authors wish to thank the experts of ISRO for their feedback in the development of this system. And also wish to thank all of whom provided wise opinions and astute suggestions.


## References


1 J.W. Ward “Microsatellites for global electronic mail networks”, Electronics and Communications Engineering Journal, IEE, December 1991, pp 267-272.

2 D. Friedman, “Error Control for Satellite and Hybrid Communication Networks, M.S. Thesis, Un. of Maryland, 1995, (CSHCN Techn. Rep. MS 95-l).

3 A. R. K. Sastry. Improving Automatic Repeat-request (ARQ) Performance on Satellite Channels under High Error Rate Conditions. IEEE Transactions on Communications, April 1975.

4 Jean J. Labrosse, “MicroCOS II : The Real Time Kernal”, CMP Books, 2002

5 William Stallings,” Data & Computer Communications”, Sixth edition, Prentice Hall, Nov 1999.

6 Kahn, R.E., and W.R.Crowther, ‘Flow Control in a Resource Sharing Computer Network,’ IEEE Transactions on Communication, 1972

7 Kamalini Martin, V.Venkatesan and U.N.Das, ‘Data Link Control in the LEO satellite store and forward network’, IEEE TENCON 2003 Conference, 2003