A FULLY IMPLEMENTED SEMI-AUTOMATED GROUND-CONTROL SYSTEM FOR THE TERRIERS SATELLITE

Nicolas Groleau
Recom Technologies/NASA Ames Research Center
Moffett Field, California

Larry Kiser
Caelum Research/NASA Ames Research Center
Moffett Field, California

Forrest Girouard, Allen Hopkins, Tom Morgan,
UC Berkeley, Center for Extreme UltraViolet Astrophysics
Berkeley, California

Supriya Chakrabarti, Timothy Cook, Daniel Cotton, Paul Dell
Boston University, Center for Space Physics
Boston, Massachusetts

Abstract

In a collaborative effort, NASA Ames Research Center, the University of California Berkeley, and Boston University have developed ground software for a small satellite. This paper describes the ground communication software, the data archiving and serving system, and the data display and satellite scheduling in sufficient detail to allow prospective small satellite operators to explore its possible use or extension.

Introduction

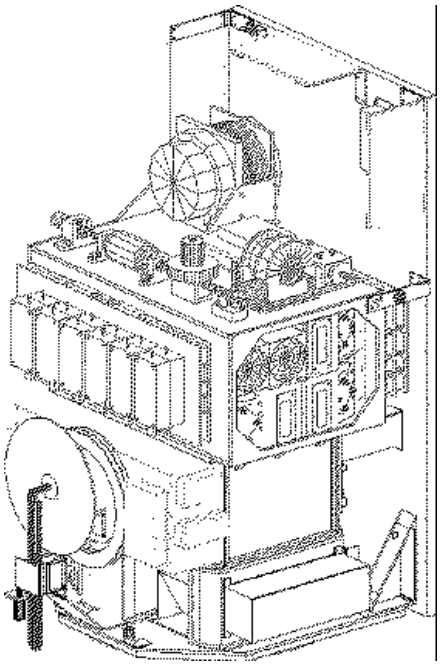
In a collaborative effort, NASA Ames Research Center's[4] (ARC's) Computational Sciences Division (IC), the University of California Berkeley's Center for Extreme Ultraviolet Astrophysics[5] (CEA), and Boston University[3] (BU) have developed ground software for BU's TERRIERS* satellite (see Figure 1). TERRIERS' primary goal is to, for the first time, demonstrate meridional 2-D (latitude-altitude)and global 3D imaging of the ionospheric electron density and thermospheric photo-emission profiles using EUV emissions and tomographic techniques. Secondary goals of TERRIERS focus on the study of several ionospheric and thermospheric phenomena through the use of this novel combination of techniques and observations. As a tertiary objective, TERRIERS will test the utility of long term solar EUV irradiance measurements of the solar EUV, one of the primary sources for the upper atmosphere.

The TERRIERS OPerationS (TOPS) system software will be used to automate the TERRIERS Payload Operations and Control Center (POCC). The TOPS philosophy is

*TERRIERS is the Tomographic Experiment using Radiative Recombinative Ionospheric EUV [Extreme Ultraviolet] and Radio Sources.

general and extensible in design. The general layout of the data processing and archiving scheme were inherited from the EUVE mission. The specific programs and configurations for the TERRIERS mission are tightly coupled to the requirements of the BU science teams and the AeroAstro[2] (AA) spacecraft bus. Many of the components were written as libraries with standard interfaces and use standard Unix communication protocols. A block diagram of the TERRIERS end-to-end system is shown in Figure 2.

Figure 1. TERRIERS Satellite



The following functional requirements were addressed by the TOPS software components.

- a system for saving incoming data on secondary media,
- a system for converting TERRIERS schedules into Inter-Process Protocol (IPP) messages required by the spacecraft,
- a system for archiving the TERRIERS data and providing access to it in a manner that is appropriate to their processing needs,
- a means of transferring files between the BU Sun workstation and a Macintosh computer at the ground-station,
- a means of displaying satellite data to a human operator, and
- a means of manually scheduling satellite operations.

There are many different ways to logically present the TOPS software. The writing of this paper was itself a collaborative venture. The next three sections describe the ground communication software, the data archiving and serving system, and the data display and satellite scheduling respectively.

Ground Communications Software

NASA ARC developed software to automate the transfer of TERRIERS telemetry data and command files between the AA Ground Station (GS) (located in Virginia) and the TERRIERS facilities (located in Boston, Massachusetts). Telemetry data files are transferred from AA to BU using the File Transfer Protocol (FTP) over the Internet; Command files are transferred from BU to AA using standard telephone service and the zmodem file transfer protocol.

The next two sections discuss the TERRIERS Telemetry File retrieval software and the Command File transmission software that were developed at NASA ARC.

Telemetry File Retrieval

NASA ARC developed a suite of ftp utilities that is being used to automate the transfer of TERRIERS telemetry files across the Internet. This software is resident on a Sun Workstation located at Boston University.

The **ftp_from_gs** module is one component in a suite of automated ftp utilities. It was developed using a standard Unix Scripting language (the Unix Bourne shell) and is responsible for retrieving and deleting telemetry data files that are on the GS Macintosh Computer. The automated ftp tool suite consists of the software shown in Table 1.

Communications Blackout

During a satellite pass, the GS Macintosh Computer is unable to simultaneously process ftp requests, and ingest telemetry data files or uplink command files. It is necessary to prevent ground-based data transfers to or from the GS Macintosh Computer during a satellite pass. Therefore, a few minutes prior, during, and a few minutes after a satellite pass, the TOPS software is prevented from communicating with the GS Macinstosh.

Table 1. Autoftp Software Modules

ftp_from_gs	Manages the retrieval and deletion of remote telemetry files. It is periodically executed by a **watchd**** program in combination with the Unix **cron** and **at** commands.
ftpget	Retrieves files from a remote host.
ftpdelete	Deletes files on a remote host.
ftplist	Lists files on a remote host.
ftpput	Transmits files to a remote host.

Telemetry File Retrieval Process

The telemetry file retrieval process used by **ftp_from_gs** software module is listed below:

1. **ftp_from_gs** uses the ftplist command to obtain a list of remote files on the AeroAstro GS Macintosh computer.

2. **ftp_from_gs** polls the tape backup system to determine the local status of each telemetry file contained on a specified folder on the remote GS Macintosh computer.

3. All remote files that have been stored locally (i. e. on the Sun computer) on tape will be deleted from the GS Macintosh computer using the **ftpdelete** command.

4. Remote files unknown to **tbackup** will be retrieved from the GS Macintosh computer using **ftpget**.

---

watchd was developed by CEA/UCB. It monitors user-specified directories for the existence of specific files and executes user specified Unix (Bourne Shell) commands.

5. Once **ftpget** successfully retrieves a file named "**filename**", it locally creates a new file named "**filename**. **complete**". This signals other TOPS elements that a file is ready for processing and prevents the processing of incomplete files.

6. Newly retrieved files are automatically added to the **tbackup** index, but are NOT automatically stored on magnetic tape. A person must physically load a magnetic tape and execute the proper command to actually save the files on magnetic tape.

7. Any remote file that is stored on Sun Computer but is not on tape will be untouched by **ftp_from_gs**.

Telemetry File Retrieval Options

The following are file retrieval options:

1. The log file options for errors, failed transfers, and successful transfers.

2. The number of retry attempts for recovering from unsuccessful attempts to establish an ftp session.

3. The full pathname on the local host where incoming telemetry data files should be stored.

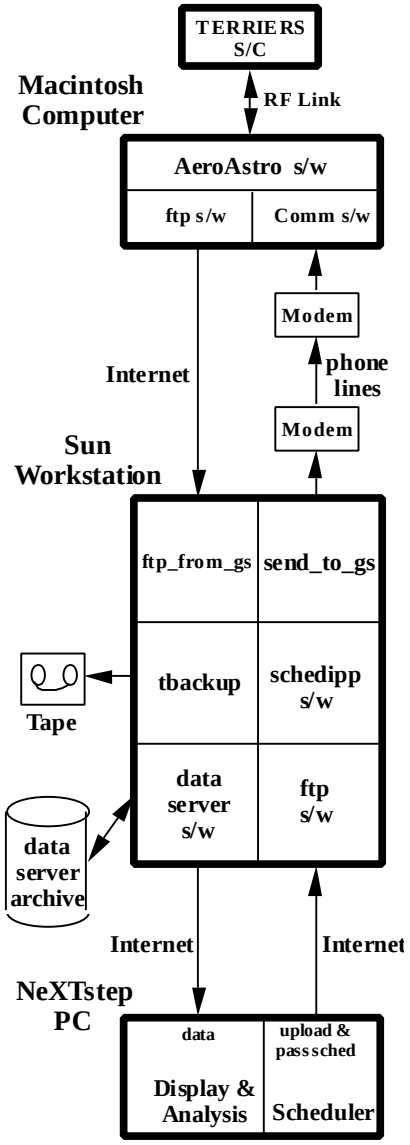


Figure 2. TERRIERS POCC System

Autoftp Modules

To use the **autoftp** modules, the user MUST specify the remote host name, and the corresponding username and password on that host. This information can be specified in one of three ways: command line arguments, the "**.autoftprc**" file, or environment variables. In case of conflict, there is a set order of precedence for these parameters. The optional **autoftp** parameters is shown in Table 2.

Table 2. Optional Autoftp Parameters

---

tbackup was developed by CEA/UCB. It partially automates the tape backup process. Also, it can provide status information regarding archived files.

-p **command line flag**	Prompt the user for a password.
-l **command line flag**	For ftplist only, long directory listing format. The default is the short listing format.
-d **command line flag**	Full pathname of the remote directory where files are to be listed, retrieved, transferred, or deleted. The default is the user home directory on the remote host.

Autoftp Sample Usage

% **ftplist** -h pancake.bu.edu -u terriers -d "/home/terriers/data" -P secret

filename1
filename2
gold.1
gold.2
gold.3

In this example, "pancake" is the remote host, the username is "terriers", the remote directory is "home/terriers/data", and the user password is "secret". **Ftplist** produces the directory listing for the specified remote directory.

Command Transmission

For increased security, TERRIERS command files are transferred from the Sun Workstation to the GS Macintosh Computer across telephone circuits using callback modems. To perform this function, NASA ARC developed two software modules (**send_to_gs** and **modem_send**), whose functions are described in Table 3.

Communications Blackout

To prevent computer resource conflicts on the Ground Station Macintosh Computer, the same communications blackouts apply to **send_to_gs** that apply to **ftp_from_gs**.

Send_to_gs

The **send_to_gs** command is executed periodically using the **watchd**, Unix **cron**, and Unix **at** commands. The **send_to_gs** module was developed using a standard Unix Scripting language (the Unix Bourne shell) and is responsible for transmitting command files from the Sun Computer to the Macintosh Computer.

Send_to_gs Transmission Process

The transmission process for **send_to_gs** is listed below:

1. Check for file existence of the files that are to be sent to the Macintosh Computer.
2. Transmit existing files using **modem_send**.
3. Upon successful file transmission, delete the file(s) from the appropriate local directory.

Table 3. Command Transmission Software Modules

send_to_gs	Transmits TERRIERS upload files to the GS Macintosh Computer.
modem_send	Automated utility for transferring files across telephone connections using the zmodem file transfer protocol.

Command File Transmission Options

The following are user options for **send_to_gs**:

1. The directory name from which files are to be sent. The default directory name is set in the configuration file.
2. The log file options for errors, failed transfers, and successful transfers.

modem_send

The **modem_send** application is an automated utility that supports file transfers. It uses the **zmodem** file transfer protocol to exchange files between two computers. **Modem_send** is configured for transferring files from a Sun Computer to a Macintosh Computer (hereafter referred to as the local host and the remote host, respectively).

The local host initiates the file transfer to the remote host, which is presumably unattended. The remote host must use a callback modem that has its callback security feature ENABLED. Also, the remote modem must be configured to place a return telephone call to the local modem's telephone number (after the correct modem password has been entered). The local host must use a modem with callback security DISABLED and configured to NOT answer incoming telephone calls. This is critical to prevent unauthorized persons from connecting to the local modem.

modem_send Process

The following is a summary of the event sequence that will occur during a **modem_send** file transfer:

1. An outbound telephone call is initiated by the local host.
2. The incoming telephone call is answered by the remote host's modem.
3. The local host supplies a password to the remote modem.
4. The remote modem immediately terminates the current telephone connection.
5. The remote modem places a return telephone call to the local modem.
6. The return call is answered by the local modem in response to an "**ata**" modem command.
7. A **zmodem** transfer is initiated for each specified file.
8. This process is repeated for each specified file until finally, the telephone connection is terminated.
9. All errors, failed transfers, and successful transfers are saved in log files.

modem_send Usage

To use **modem_send**, the user must specify the local modem device name, remote modem telephone number, and the remote modem password. This information must be specified in one of three standard Unix ways: command line arguments, the

".**modem_sendrc**" file, or environment variables. In case of duplicate parameter assignments, there is a specified order of precedence.

Optional modem_send Parameters

Table 4 describes the user options for **modem_send**.

Table 4. Optional modem_send Parameters

device name	Specifies the local modem device name.
-v flag	Indicates invokes the verbose mode, which provides detailed output that can be used for diagnostic purposes.
Redial Attempts	If specified, this determines the number of redial attempts when the remote modem's telephone number is busy. The default value is 10.
Redial Spacing	If specified, this determines the spacing (in seconds) between redial attempts, The default value is 10 .

Sample Modem_send Usage

Example 1

```
% modem_send -d "/dev/cua2" -n
    "9,(703) 555-1212" -P secret gold*
Successfully transferred "gold.1".
Successfully transferred "gold.2".
Successfully transferred "gold.3".
```

This specifies a modem device named "/dev/cua2", telephone number of "9,(703) 555-1212", and transmits all files on the current working directory of the **LOCAL** host that begin with the characters "gold". The output indicates that three files named "gold.1", gold.2", and "gold.3" were successfully transferred.

Example 2

```
% modem_send -d "/dev/cua2" -n
    "9,(703) 555-1212" -P secret
    silver.1 silver.2
File silver.1" does not exist.
Successfully transferred "silver.2".
```

This specifies that "silver.1" and "silver.2" are to be transferred. A warning message is generated to indicate that "silver.1" does not exist and that "silver.2" was transferred".

Example 3

```
% modem_send -d "/dev/cua2" -n
    "9,(703) 555-1212" -P secret
    silver.2
File "silver.2" does not exist.
No valid files specified!
```

This specifies that one file named "silver.2" is to be transferred. The warning message indicates that "silver.1" did not exist and that no files were transferred.

Modem_send Caveats

On the local host, the user must have read and write privileges for the both current working directory and the specified file(s); otherwise, the local copy(ies) of the specified file(s) will not be deleted. Also, if the user attempts to transfer a file named "**filename**" that does NOT have a corresponding file named "**filename. complete**" in the current working directory and **modem_send CANNOT** create it with the Unix "**touch**" command, then "**filename**" will NOT be transferred.

However, other files with corresponding "**.complete**" files will be transferred.

The Macintosh is a "weak" link in the TERRIERS ground system. If an active **zmodem** file transfer is interrupted by the Sun Computer or by an unexpected loss of the active telephone connection, the Macintosh will require resetting. If such an interruption occurs, the Macintosh will generate an alert box advising that the host has canceled the transmission or that the Macintosh cannot connect to the sender. No future **zmodem** transmissions can proceed until a person closes the Macintosh's Alert box.

Data Archiving and Serving

The TERRIERS requirements for data archiving and serving were decomposed into 4 modules:

- **tbackup**
- **ds** (archiver)
- **tds** (data server)
- **schedipp**

FTP and Remote Process Control (RPC) are used to communicate between the different platforms while the **watchd** utility is used to automate the processing on the SUN workstation.

The description of the functions and portability of each module comprises the following section.

Data Archiving and Serving Modules

These modules were designed and constructed with varying degrees of portability. The **tbackup** utility resides at the portable side of the spectrum while the **schedipp** module is very specific to the TERRIERS application. Compromises between mission specific and completely portable were made early in the design decision. A six month schedule forced many decisions to be made in favor of mission specific design.

Generally mechanisms on the BU Sun workstation communicate with each other by using the **watchd** utility and dropping files into directories. Using the **watchd** utility and these directories provides a great deal of flexibility in terms of notifying and/or invoking other processing agents. RPC is used as a communication mechanism between the data server and the data server interface (located on the NextStep PC).

Data Archiving (tbackup)

It was determined a backup function was necessary as early in the data flow as possible to guarantee data integrity. Budgetary constraints dictated a simple, inexpensive and automated solution to transferring the raw telemetry to inexpensive media. The backup process takes place on the BU site since it is manned. The IPP protocol guarantees transmission from the spacecraft to the ground-station. FTP guarantees transmission and data integrity from the ground-station to BU.

There were several subtleties inherent to the relative cost of the system which influenced the final design. Most locations on media can be specified with the following quintuple: media type, volume, file number, partition, and pathname. These five values along with a unique file identifier can be used to specify a media independent backup location. The backup scheme also involves confirmation that retrieving the file from the backup location results in the identical content as the original. For example, when using the Unix **tar** command to write to tapes the file should first be written to the tape, and then

extracted and compared to the original (using **cmp**). Once confirmed, the file is marked for removal from the Macintosh GS. 8mm cartridge tapes were specified as the media but the full capacity of the tape (2 Gbytes or 5 Gbytes) could not be used as it takes a very long time to move through that much data when it is in small pieces. The length of time to retrieve files from tape is a combination of the physical location on the tape and the number of files occurring before it on the tape. This means that writing many small files to a tape still makes for slow retrieval even though only a small fraction of the tape capacity is in use. Currently the number of files on a tape is limited to 300 and the capacity is limited to 1.25 Gbytes including the End Of File (EOF) markers. Each EOF marker consumes 2 Mbytes. The backup naturally involves an operator to load, unload, label and store tapes.

The resulting solution is a simple, generic set of Bourne shell programs using an ASCII database file for indexing. The **tbackup** program is invoked when data is returned by the **ftp_get** routine and dropped into the incoming box. Each file that is backed up is cached under a directory named after the date and time of the last tape write. The backed-up file's name is recorded in a special file, **files.list**, along with its size and the name of the current cache directory. When the cache is written to tape the entire current cache directory is written to tape in a single **tar**(1) file, a new cache directory is created named for the current date and time, and a symbolic link named "current" is created to point to the new cache directory. The name of the just-written cache directory is recorded in a second special file, **cache.list**, along with its size, the tape identifier (a unique integer), and the file position on that tape.

These uniquely-named cache directories remain on disk in a special directory until a tape write is done (**-w option**), at which point cached files are removed, oldest first, until the cache disk usage reaches the maximum allowed size.

The two special files, **files.list** and **cache.list**, can be queried to find the tape location of any backed-up file, and to keep track of the amount of data written so far to the current tape to avoid exceeding the maximum tape usage. The files are organized like relational database tables and could conceivably be replaced by a relational database with a minimum of impact to the code.

The archiver (ds)

The EUVE archiving philosophy was largely replicated for the TERRIERS archiving mechanism. Raw IPP packets are the format of the archived telemetry data. All products are recreated for each type of query from the original binary IPP messages. This approach was used for two reasons:

1. the binary format is very compact and easy to archive
2. processing is done only as necessary and for the specific data requests.

The processing cycles and disk space saving for these two reasons were well proven by the experience with EUVE.

The same **watchd** process that runs **tbackup**, invokes **ds** on the new telemetry files (IPP messages) and adds them to the data server archive. The data-server archive consists of three sets of files. First is the set of data files being stored; second is a set of database files used to access the stored information; third is the single latest-value-table file. The locations of each of these is identified by environment variables : one each for the archived files, the database files, and the full pathname of the latest-

value-table file. The data files are organized in hierarchical directories by the year, month and day they were entered into the archive.

The **ds** program offers coalescing for optimization index search efficiency for particular data by reducing the number of data repetitions. Intervals are initially identified in the database when **ds -p** is run, one interval is created per IPP packet. Periodically, **ds -c** can be run to coalesce these many one-packet intervals into much larger ones, each from one file, representing all the data of a particular type covering a particular span of time. This greatly decreases the number of intervals that must be kept in the database and simplifies access to the data. This coalescing is done as soon as it is certain that no more data is expected from within that range of time, which would cause large intervals to have to be split up to accommodate the newly arrived data.

In normal operations, the **-a**, **-p**, and **-c** options are used regularly to add files, add indexing information and coalesce indexing information, respectively. The **-f** and **-i** options can be used to get information about the contents of the archive.

The User Interface to the Data Server (tds)

In order to support flexibility in the hardware which would not comprise the TERRIERS LAN the design for the interface to the data-server adopted the SUN RPC standard. This interface decision allows the data-server to be queried by virtually any type of hardware. All data conversions between data types are handled by the RPC standard. This proved to be an important decision for the TERRIERS mission since the late addition of a DEC Alpha to the list of machines which need access to the data-server proved transparent.

The **tds** program provides for an interface between the client making the request for specific telemetry types and the data-server. The queries are made in the form of C-data structures and information is returned via C-data structures.

Schedule Creation (schedipp)

The **schedipp** software was designed to be as accessible to a human operator as possible. The input for **schedipp** is a human readable ASCII file with TERRIERS schedule requests. The requirements for the input are very minimal and can be created by any software that can output ASCII. The knowledge of specific TERRIERS commands and configurations are all inherent to the **schedipp** program. The result being a great deal of flexibility for the scheduling station selection and design.

The command uplink portion of the data flow begins at the scheduling station with a science request. All resource allocation and constraint checking for the request are performed on the scheduling station NextsStep PC and result in the output of a named TERRIERS schedule file. The schedule file is FTPed to the SUN machine and deposited in a special directory. A watchfile in this directory directs **watchd** to run **tbackup** to add the schedule file to the tape backup, **ds** to add it to the data-server archive, and **schedipp** to create scheduled upload and unscheduled upload files of IPP format. These IPP files are placed in a holding directory. When they appear in that directory, **watchd** consults the pass schedule to see if it is OK to send them and, if OK, runs **send_to_gs**, which sends them and deletes them from the holding directory. If it is not OK to send, i.e., if during or close to a spacecraft contact time, the files are not sent until the black-out time has ended.

Commands in the input which are scheduled for execution at some future time generate IPP messages which go to an upload-scheduled file. IPP messages for commands which are to be executed

immediately go into an upload-unscheduled file

The properly formatted IPP messages containing the scheduled upload and unscheduled upload files are then copied to the **send_to_gs** directory where the **modem_send** software transfers it to the ground-station.

Data Display And Satellite Scheduling

Satellite operations scheduling is achieved by packing IPP messages in a batchfile format and uplinking the batchfile. Messages scheduled within a batchfile contain all the timing information including time to dispatch the message, number of times to the repeat message, and repeat delay interval. The batchfile facility provides for timing information and allows one to group related messages. The batchfile is given an identification number (ID) on the satellite which is downlinked. This batchfile ID can later be used by the operator or the ground software to cancel the further execution of events contained in the batchfile. Therefore one can upload nominal operations which can then be canceled and a new schedule can be executed in its place.

Spacecraft Communications Protocol

Message handling within the satellite subsystems and between the satellite and the ground are all handled with the IPP developed by AeroAstro Corp (Herndon, VA). This protocol is being utilized by both the MIT HETE[1] and Boston University TERRIERS spacecraft missions. Each IPP message contains a header, see Table 5 for values, and a 0-1024 byte length data field.

Data Display

The data archive is queried to retrieve engineering and science messages from the time period of interest. Queries to the data server specify a time period and a data type. Optionally the query may indicate specific messages of interest or priority levels.

Table 5: Inter-Process Protocol

Header Field	Description
Magic	Sync word
Type	Identifies the function of the message as well as the format of the data field.
Subtype	Further identifies the function or data format.
Priority	Specifies the priority of the message. Is used on the satellite for priority based memory management and to order messages to be downlinked.
Length	The number of IPP words (16-bit) contained in the data field.
Sequence	Indicates the relative order in a sequence of messages with the same originating process and shared data field, i.e. the data field is greater than 1024 bytes therefore data field is broken into a series of IPP messages ordered by the sequence number.
Destination	The IPP name or address of the destination process.
Source	The IPP name or address of the originating process.

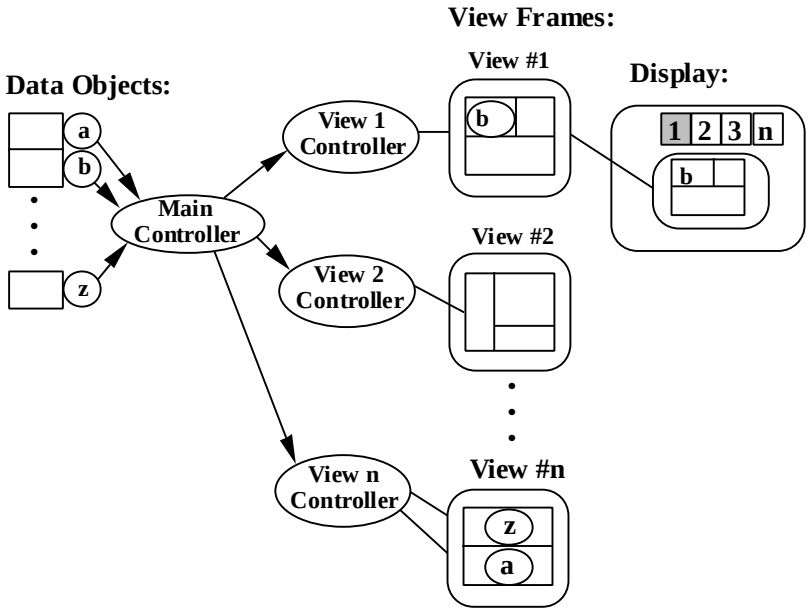
Timestamp	Bits 24...39 of the 64-bit of the spacecraft time[à] at the time the message is sent.
Checksum	Flag to determine if optional checksum is to be used for this IPP message.

Querying the message archive will either return null if no messages match the query or will return a pointer to the request structure if the data server found matches. A second call is made referencing the returned request address and the first in the series of messages is returned. The series of messages that match the query are returned as a linked list of summary data. The linked list is sequentially scanned. Each message may be passed over or retrieved. When a message is retrieved, local memory must be allocated, and a separate function is called to cast the message into the memory structure on the local machine. When the end of the query is reached, a call is made to free the previously requested list of objects. At this point the data server is available for another query request.

The process of retrieving & loading information from the data archive is displayed in Figure 4. When data of interest is encountered, a local data object is allocated and initialized with the correct header values. As shown in Figure 3, the newly created object first copies the header information, then checks for information in the data field. The data field information is copied into a field which is binary data streams. Then the object is either immediately displayed or in most cases the object is passed on to the proper view.

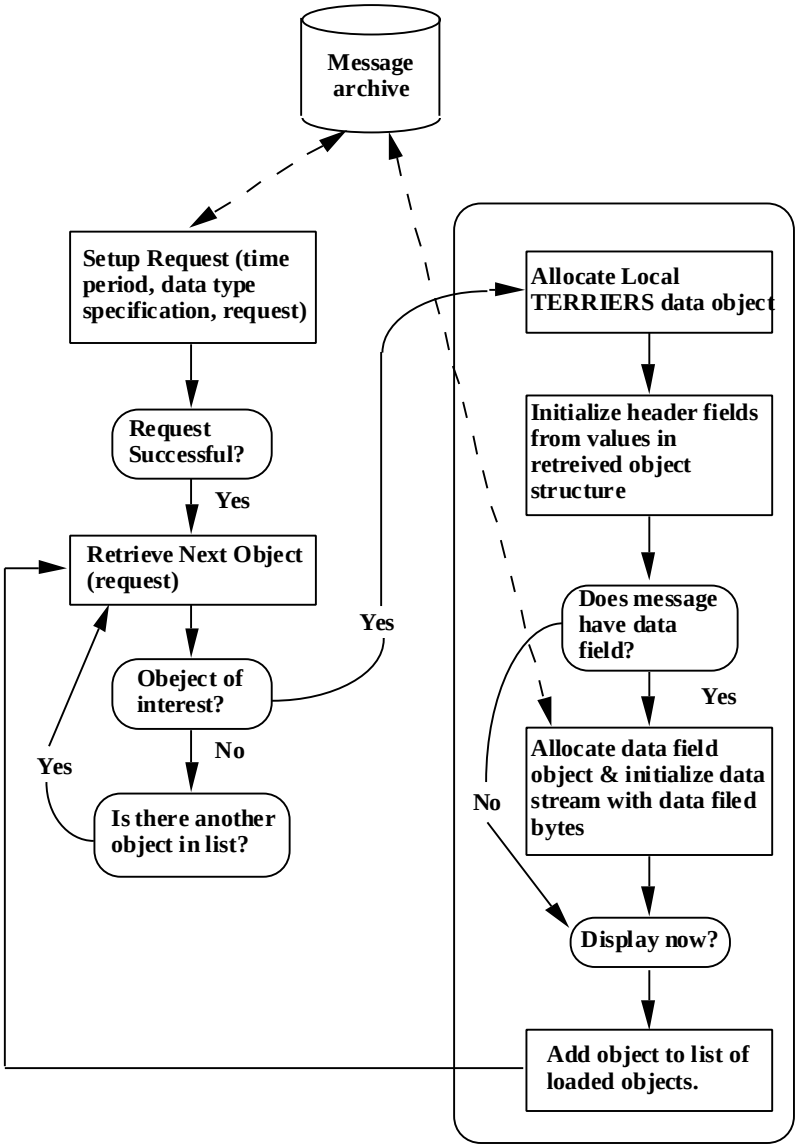
[à]The ground-station Macintosh has independent calculation of the predicted spacecraft contact times. A ranging procedure on the ground-stayion will determine the spacecraft-GMT correlation.

Figure 3. Displaying Data Objetcs Using Views



After the object(s) from the query are loaded, the objects are assigned to a user interface view. . Figure 3 shows the process for displaying the data. Because of the large amount of data and various data types and priorities, the visualization screen consists of two current status displays that cannot be removed from the screen and a main display which toggles between different data views. The loaded data objects need to be associated with the proper subviews & screen elements. This is achieved by first filtering the data objects through the Main Controller object which associates the data object to the correct View Loader. Next the View Loader adds the object to its own list of loaded objects and assigns the object to the proper screen object. After the objects are loaded into the proper views, the user may select any subview and this triggers the display of data associated with that view.

Figure 4. Retrieving & Loading Information From The Data Archive



This approach to retrieving and displaying data allows the user to query for and examine the data of interest while minimizing the amount of data being transferred, delaying type casting until needed, allowing for incremental visualization, and allowing the examination of various events through the use of multiple data windows.

Satellite Scheduling

Scheduling from the operators standpoint is achieved in a simple three step process:

1. A message is constructed. This is done by selecting the IPP message type & subtype, setting the time to be executed and any timing for repeated events, and finally setting any parameters if the message contains user selectable parameters.

2. The event is added to the schedule by dragging the event icon onto the schedule.

   The first 2 steps are repeated to add any additional events.

3. The schedule is "saved" by the operator which parses and transmits the data to the archive.

Operation constraints are checked when the event is created and when the event is added to the schedule. In order to operate the satellite effectively and safely, messages are simple and error checking occurs on the ground and in the satellite. The satellite is designed to operate in five basic science modes. The algorithms for the mode operations reside on the satellite and are tested rigorously. The mode messages contain a number of parameters which will set the suite of instruments to use and set safety thresholds. Each mode may be assigned to be Day or Night Side. Modes that are designated as night side will not be executed on the day side, though there is no danger in running a day mode on the night side and this is permitted. On the ground, a Night mode message is not allowed to be scheduled for a time that is predicted to fall in the Day. Also on the satellite a Night side message is not executed until the sun sensor is checked to confirm that it on the night side. Other constraints checked on both the ground and satellite include battery level and timing conflicts with executing or previously scheduled messages. In order to reduce operator errors, user selectable parameters are limited by bounds set by the satellite management team and modifiable only by the computer system administrator.

Conclusion

This collaborative effort has produced a semi-automated Payload Operations and Control Center tailored for the TERRIERS satellite but relying as much as possible on standard hardware and software elements.

In particular, software design was largely driven by reuse of existing software and libraries and strived to produce reusable libraries whenever possible.

This paper presented the overall architecture is sufficient detail to allow prospective small satellite operators to explore possible use or extension of TOPS.

References

1. HETE satellite built by AeroAstro and providing the bus architecture for TERRIERS: *http://space.mit.edu/HETE/*

2. The Aero Astro company responsible for the TERRIERS satellite bus and ground-station: *http://www.isso.org/Industry/AeroAstro/AeroAstro.html*

3. Boston University's TERRIERS home page: *http://net.bu.edu/terriers/terriers.html*

4. NASA Ames IC's TERRIERS home page: *http://ic-www.arc.nasa.gov/ic/projects/Auton-ops/TERRIERS/TERRIERS.html*

5. UC Berkeley's Applied Research Technology Group home page: *http://www.cea.berkeley.edu/~art*

Acknowledgments

The support for the TERRIERS mission supplied by the Center for EUV Astrophysics was funded by the following grants:

- NCC-902
- NAS4-89584

CEA wishes to express special thanks to Dr. Mel Montemerlo (NASA HQ, Code S) and Dr. Dave Korsmeyer (NASA Ames Research Center, Code IC).

Boston's University's research was supported under USRA contract number 1500-05.

Author Biographies

Dr. Nicolas Groleau

Dr. Nicolas Groleau is leading the Autonomous Operations Group at the Computational Sciences Division of the NASA Ames Research Center. He his a Sr. Computer Scientist and Deputy Program Manager with Recom Technologies, Inc.. He obtained his Ph.D. in Computer Science and Civil and Environmental Engineering from the Massachusetts Institute of Technology in 1992. His research interests focus on intelligent computer tools helping humans involved in space missions, including procedural training, scientific discovery, and satellite operations.

Larry M. Kiser

Larry Kiser has BS and MS in Electrical Engineering from the University of Virginia. He is currently employed by Caelum Research Corporation and works on site at NASA Ames Research Center. He has prepared specifications for the Space Station's Communications and Tracking System, analyzed Radio Frequency Interference for INTELSAT and for NASA, provided systems engineering support to NASA Headquarters' Tracking and Data Relay Satellite System (TDRSS) Project Office, assisted in the development of functional requirements for National Oceanic and Atmospheric Administration's satellite ground stations, and developed utilities to automate data transfers across the Internet and across serial modem connections.

Forrest Girouard

Forrest Girouard received his B.S. degree in Computer Science from the University of California, Berkeley in 1990. He has been a software engineer at IBM and is currently the system architect for the Information Systems Development group at the Center for EUV Astrophysics.

Allen Hopkins

Allen Hopkins received his B.S. degree in Computer Science from San Francisco State University in 1989. He has been a programmer/analyst for the Extreme Ultraviolet Explorer Project, and later, the Center for EUV Astrophysics at the University of California, Berkeley since 1985. His areas of concentration have been data storage and retrieval and the monitoring of payload health.

Tom Morgan

Tom Morgan received his BA degree in Philosophy from the University of California, Berkeley in 1993. He has been with the Center for EUV Astrophysics and manager of the Applied Research Technology group since his graduation. Areas of interest include information indexing and retrieval in large unstructured data bases and man/machine interfaces.

Dr. Supriya Chakrabarti

Dr. Supriya Chakrabarti received his Ph.D. from University of California at Berkeley. He has been involved in several space flight missions including U. S. Air Force's STP78-1, FAUST and EUVE. He has also developed many sounding rocket experiments and is presently a member of the team developing TERRIERS which is one of the STEDI missions being managed by USRA. Prof. Chakrabarti is a member of the Center for Space Physics, Department of Astronomy and the Center for Photonics Research at Boston University.

Dr. Timothy Cook

Dr. Timothy Cook, an astronomer with the Center for Space Physics at Boston University, specializes in developing new instrumentation for ultraviolet space astronomy. He received his Ph.D. from the University of Colorado in 1991 and joined the Boston University team in 1992.

Dr. Daniel Cotton

Daniel Cotton received his B.S. degree from the University of California at Davis and his M.A. and Ph.D. degrees from the University of California at Berkeley, all in physics, in 1986, 1990 and 1991, respectively. He continued on at Berkeley for a year as an Assistant Research Physicist. Presently, Dr. Cotton holds a research Professor position at the Center for Space Physics at Boston University. His current research interests are in experimental EUV and FUV aeronomy.

Paul Dell

Paul Dell, a Computer Science graduate student at Boston University, is the software developer for the ground based TERRIERS monitoring and control system. He received his B.S. in Physics at Gordon College in 1992 and is now pursuing Ph.D. in Computer Science.