Astronomical Data Analysis Software and Systems XIX
ASP Conference Series, Vol. 434, © 2010
Y. Mizumoto, K.-I. Morita, and M. Ohishi, eds.

A Distributed, Real-Time Data Monitoring System as Ground Support Equipment for Balloon-Borne Astronomy Experiments

C. M. Hubert Chen, Wayne H. Baumgartner, Walter R. Cook, Andrew J. Davis, and Fiona A. Harrison

California Institute of Technology, Pasadena, CA, USA

Abstract. We present a real-time data-monitoring software suite that we developed for the High Energy Focusing Telescope (HEFT). HEFT was one of the first projects to develop focusing mirrors and detectors for hard X-ray astronomy. We deployed these new technologies on the scientific ballooning platform. During a balloon flight, this so-called 'ground support equipment' (GSE) allows us to monitor the physical condition of the payload, and to inspect preliminary science data in real time, through displays of tables of frequently updated quantities and their averages, time-series plots, histograms, spectra, and images. Unique from previous implementations of GSEs for other experiments, our system is a server-client network that utilises TCP/IP unicast and UDP multicast to enable multiple, concurrent and independent display clients. Most of the code is in Java, and thus platform-independent. We verified that the software suite works on Linux, Mac OS/X and Windows XP, deployed it in two flight campaigns for use during on-site calibration, pre-launch practice drills, and an observation flight of 24 hours. This system, and individual ideas of its implementation, can be adapted for use in future experiments requiring sophisticated real-time monitoring and data display.

1 Context

Scientific ballooning is a platform widely used in experimental astronomy to develop and test telescope and remote-sensing technologies targeted for future space missions. Each payload of 4500 lb maximum gets lifted to the top of the atmosphere (45 000 m altitude) for about 30 h/flight, twice a year maximum, by a zero-pressure balloon of the size of a football field. Obviously, real-time, remote communication between the payload and the ground crew is crucial to successful flight operation. To this end, experimenters write custom software, commonly known as ground station equipments (GSEs), to monitor the physical condition of the payload, display preliminary science data, and possibly perform data analysis, all in real-time. In this paper, we present the Science GSE of the High Energy Focusing Telescope (HEFT).

HEFT (Harrison et al. 2005) was one of the first projects to develop focusing mirrors and matching detectors for the hard X-ray band (10–100 keV; for HEFT, 20–70 keV). Focusing makes direct imaging in hard X-ray possible,providing 1000 times sensitivity gain, and 10 times better spatial and spectral resolutions of $1'$and 1 keV. Science data from a focusing telescope such as HEFT are events, each corresponding to a hard X-ray photon detected on the focal plane of the instrument. Each event contains information on the location (pixel coordinates), energy and arrival time of the detected photon, as well as some other variables about the condition of the detector at the time of detection. We structure the serial science-data stream into one-second frames of variable length depending on the data rate. Each frame consists of a header section to record housekeeping data such as voltage-, temperature- and pressure-sensor readings, followed by a data section made of a sequence of events detected during that second.

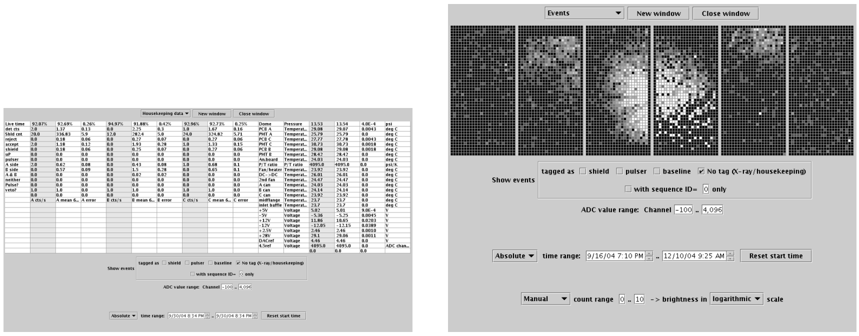
2 Objectives

The requirements of the HEFT Science GSE are to log, parse and process the raw data stream from the payload, and then display in real-time the housekeeping data in tables, and event data as hard X-ray images. It should also provide capabilities for users to interactively select subsets of display data in time, energy, and space (ie, pixel IDs), to display time series (i.e., light curves), spectra, and other specialised charts upon user request, and to retrieve archived data saved in previous experiment runs. The system must be stable over the duration of a balloon flight and any pre-launch calibration experiments. Raw data logging must be impeccable, as an unsuccessful landing of the payload is a very real possibility, in which case, the raw data logged by the Science GSE could be the only surviving copy. The user interface should be fast, responsive, and well-designed. Components of the GSE should be scalable. Specifically, the system must allow telescope modules to be added incrementally due to a long hardware production schedule; experiment conductors may want to add new, derived variables for monitoring; it is also desirable to have multiple displays, without affecting reliability and performance, so that monitoring tasks can be divided among the launch crew. We designed the HEFT Science GSE with these requirements in mind; the following sections detail the final design.

3 System Architecture

The HEFT Science GSE is a network application consisting of servers for logging, processing and serving data, and clients for displaying the data and receiving selection requests from users. During operation, the payload sends the science data stream to an EIA-422 serial line either directly (for ground testing) or via radio telemetry (in flight). Because the integrity of the logged data is paramount, we split the EIA-422 line into two in hardware, and direct one stream into a server (the 'data logger') dedicated to logging the data (into a RAID-1 disk array) and nothing else. The second data stream goes into the 'data server', which parses the stream, breaks frames into individual events and housekeeping data items, immediately broadcasts these data items as packets to display clients via a UDP socket, inserts them into efficient data structures in memory for subsequent retrieval upon client request, and also saves them individually into disk for long-term retrieval. The data server listens through a socket to requests from display clients for specialised data streams, such as time series of a specific housekeeping data item or spectra of events at a specific pixel. Upon such a request, it spawns a new thread to serve the requested data from memory via a dedicated TCP/IP connection. If a client requests data recorded before the start of the current session, the data server retrieves the requested data previously saved from disk to memory before serving.

The client side of the GSE consists of multiple, visually distinct components to display the data in different parameter spaces. The main applications are widgets that each shows either a table of the latest housekeeping sensor values, or maps of the focal-plane detectors displaying per-pixel event data, such as event count or the latest current reading (see Figures 1a and 1b). In the housekeeping data tables, each cell is alarmed—when the latest sensor value goes out of a preset valid range, the cell background turns red and the client machine optionally sounds a beep. In detector maps, when a user hovers the pointer within a pixel, a small pop-up box appears to show the pixel coordinates and displayed value in numbers. Individual elements (table cells and pixels) within both types of widgets are clickable; upon clicking, it spawns a new widget to display the time series (or light curve) and histogram (or spectra) of data at the selected cell (or pixel). These time- and energy-space plots can zoom in and out. At the time of its maiden launch in 2005, the HEFT payload carried three telescope modules, co-aligned but rotated at different angles relative to each other. Also, all modules are rotated 180° when pointing moves between targets below and above 70° elevation. Due to these complications, we added an additional display client to show the



Live time	92.07%	92.69%	0.26%	94.97%	91.88%	0.42%	92.96%	92.73%	0.25%	Dome	Pressure	13.53	13.54	4.0E-4	psi
det cts	2.0	1.37	0.13	0.0	2.25	0.3	1.0	1.67	0.16	PCB A	Temperat...	29.08	29.07	0.0043	deg C
Shld cnt	20.0	336.83	5.9	12.0	282.4	5.0	24.0	324.82	5.71	PMT A	Temperat...	25.79	25.79	0.0	deg C
reject	0.0	0.18	0.06	0.0	0.27	0.07	0.0	0.27	0.06	PCB C	Temperat...	27.77	27.78	0.0043	deg C
accept	2.0	1.18	0.12	0.0	1.93	0.28	1.0	1.33	0.15	PMT C	Temperat...	30.73	30.73	0.0018	deg C
shield	0.0	0.18	0.06	0.0	0.25	0.07	0.0	0.27	0.06	PCB B	Temperat...	29.08	29.08	0.0018	deg C
uP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	PMT B	Temperat...	28.42	28.42	0.0	deg C
pulser	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	An.board	Temperat...	24.03	24.03	0.0	deg C
A side	2.0	0.62	0.08	0.0	0.43	0.08	1.0	0.68	0.1	P/T ratio	P/T ratio	4095.0	4095.0	0.0	psi/K
B side	0.0	0.57	0.09	0.0	1.5	0.28	0.0	0.65	0.1	Fan/heater	Temperat...	23.92	23.92	0.0	deg C
A & B	0.0	0.0	0.0	0.0	0.02	0.02	0.0	0.0	0.0	DC->DC	Temperat...	26.01	26.01	0.0	deg C
neither	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2nd fan	Temperat...	24.47	24.47	0.0	deg C
Pulse?	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	A can	Temperat...	24.03	24.03	0.0	deg C
veto?	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	B can	Temperat...	24.14	24.14	0.0	deg C
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	C can	Temperat...	23.92	23.92	0.0	deg C
	A cts/s	A mean 6...	A error	B cts/s	B mean 6...	B error	C cts/s	C mean 6...	C error	midflange	Temperat...	23.7	23.7	0.0	deg C
										inlet baffle	Temperat...	23.7	23.7	0.0	deg C
										+5V	Voltage	5.02	5.01	9.0E-4	V
										-5V	Voltage	-5.36	-5.25	0.0045	V
										+12V	Voltage	11.86	10.65	0.0203	V
										-12V	Voltage	-12.05	-12.15	0.0389	V
										+2.5V	Voltage	2.46	2.46	0.0010	V
										+28V	Voltage	29.1	29.06	0.0011	V
										DACref	Voltage	4.46	4.46	0.0	V
										4.5ref	Voltage	4095.0	4095.0	0.0	ADC chan...
												0.0	0.0	0.0	

ADC value range: Channel -100 .. 4,096

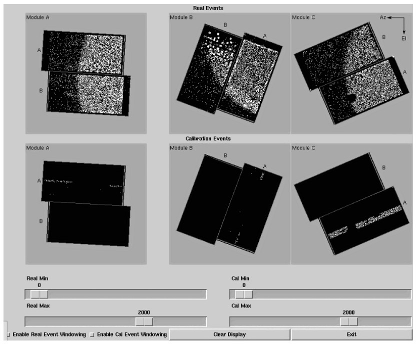
time range: 9/30/04 8:34 PM .. 9/30/04 8:34 PM

ADC value range: Channel -100 .. 4,096

time range: 9/16/04 7:10 PM .. 12/10/04 9:25 AM

count range 0 .. 10 -> brightness in logarithmic scale

Figure 1. Left: (a) Housekeeping-data client. Right: (b) Detector-map client.



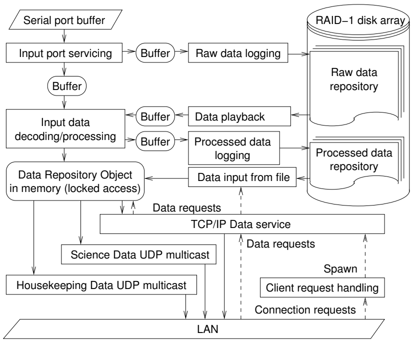


Figure 2. Rotated-detector-map client. Figure 3. Architecture of the data server.

detector maps rotated in the right orientations (Figure 2). Where appropriate, clients of all types contain control elements such as buttons and text boxes for user input to select and filter the displayed data. When a user specifies data filtering and redisplay through the controls, the client initiates a dedicated TCP/IP connection to the data server and retrieves the requested data. To receive new data, clients tune into the preset UDP multicast addresses and ports; each chooses to discard any items according to filtering criteria in effect at the individual clients.

4 Implementation Details

Server. We write the server code in Java, and run it in RedHat Enterprise Linux on machines with RAID-1 disk arrays for additional data protection, and with EIA-422 ports on PCI extension cards for data input. To interface with the EIA-422 ports in Java, we call the third-party library RXTX (http://www.rxtx.org/, which implements, in Linux, the JavaCOMM standard of Sun Microsystems for serial port communication.

The server code is multi-threaded to avoid latency and data loss due to, e.g., waiting for disk writes. Dedicated threads service the input serial port, log raw data, process raw data, log processed data (i.e., data items), multicast data items, listen to client requests, and serve requested data items. Figure 3 shows the server architecture, with each rectangular box representing a separate thread. In fact, the data logger and data server share the same multi-threaded code; they differ only in the specific threads activated.

Similarly, if desired, one can run multiple data loggers and/or multiple data servers simultaneously simply by changing the listening port and restricting UDP multicast to a single server. One can also play back data only using the same code, by running the data server without serial line input and making a data request from a client.

The core of the data server is a central data repository, which contains one table (ie, 2D array) per detector, plus one more for housekeeping items. We devised an abstraction to handle science and housekeeping data using the same framework. This greatly reduces code complexity and allows for easier extension. Each detector pixel or housekeeping item occupies one table cell, which contains two copies of the data, one as a balanced tree ordered by energy (or sensor value) and the other as a doubly linked list ordered by event (or frame) time. This duplication enables efficient data selection and filtering in both time and energy/value space. Note that new data are guaranteed to have later time stamps but arbitrary energy/values; thus the differing implementation of the two copies. All table cells are equipped with semaphores to guarantee coherent insertion and reading. The low event rate ($<$ 2 event/s on average) of our instrument means that we can store the data repository entirely in memory for the duration of the balloon flight for fast insertion and retrieval. Instead of saving this ever-changing repository to disk in one piece, we save processed data items individually to disk as compact serialized Java objects. This also provides a common abstraction for inserting items into the repository from the serial port and from disk.

Client. We wrote the rotated-detector client in perl, and all other display clients in Java. The Java clients are also multi-threaded, all sharing a common back-end for data download, server communication, and time-keeping.

Just like their server-side complements, we also wrote the two main widgets that display detector maps and housekeeping data, respectively, using a common framework of Java/Swing components, both as JTables. They differ only in having different TableCellRenderers—housekeeping table cells are simple text boxes with mutable background colour (for alarms); detector pixels are greyscaled squares whose values determine the brightness, with tool tips showing pixel coordinates and values in text. This choice of abstraction substantially reduced code duplication and complexity by eliminating the need of any line- and shape-drawing package. On the other hand, the JTable implementation restricted the maps to be upright, making rotation impossible, which was a need unforeseen until very late. As a result, we had to implement separately the rotated-detector client. At the end, the two detector-map clients complement each other, with very different feature sets and being useful in different occasions.

To build the time- and energy-domain plots in the user-requested clients, we call a third-party plotting library called Ptplot (http://ptolemy.berkeley.edu/java/ptplot/, which provides zooming capability.

5 Performance Evaluation

We ran this GSE daily in two 2-month long campaigns for pre-launch practice drills, a 110-h continuous calibration run (wired input directly from the focal plane), and a 25-h balloon flight (input from radio telemetry with data dropouts). The software functioned within specifications. This system, and individual ideas of its implementation, can be adapted for use in future experiments requiring sophisticated real-time monitoring and data display. We welcome discussions of collaboration and code reuse.

Acknowledgments. HEFT was funded by the NASA Space Science "Supporting Research and Technology" (SR&T) programme under award number NAG5-5398.

References

Harrison, F. A., et al. 2005, Experimental Astronomy, 20, 131–137