Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

An Overview of the Large Binocular Telescope Control System

T.A. Axelrod and M.D. De La Peña

Large Binocular Telescope Observatory, University of Arizona, Tucson, AZ 85721

Abstract. The Large Binocular Telescope (LBT) consists of two 8.4-meter mirrors on a common mount. This configuration provides the light gathering power equivalent to an 11.8-meter telescope and the resolving power of an 22.8-meter telescope. Due to the binocular nature of the telescope, there are unique requirements imposed on the telescope control system (TCS) to ensure the health and safety of the telescope, while also enabling observations in both independent and interferometric mode.

This paper presents an overview of the design of the TCS, from the graphical user interface (GUI) and the plotting and analysis capabilities at the highest interaction level to the low-level interfaces for the software.

1. Overview of the System

A *logical* view of the LBT TCS subsystems, represented by rectangles, and the controllers, represented by rounded-corner rectangles, is shown in Figure 1. Subsystems are autonomous entities which control attached telescope hardware through one or more interfaces. In contrast, controllers are autonomous entities without attached hardware. The connecting lines in Figure 1 represent the major communication paths between components.

From the physical point of view, the TCS has three support (control, global memory, and telemetry) and an instrument network, all of 1 Gb/s capacity. The *control* network is used for shipping commands through the system. The *global memory* network is used to maintain a globally accessible data structure which contains complete state information for the system. This global or shared memory is replicated on each platform, and therefore, is robust to individual subsystem failures. Telemetry streams can originate in any subsystem and are stored in the telemetry archive. Certain subsystems can require high bandwidth (MCS servo loops, AO loops) which cannot be accommodated on the other networks, raising the need for a dedicated *telemetry* network. The *instrument* network services the requirements of the instruments and storage of the acquired data.

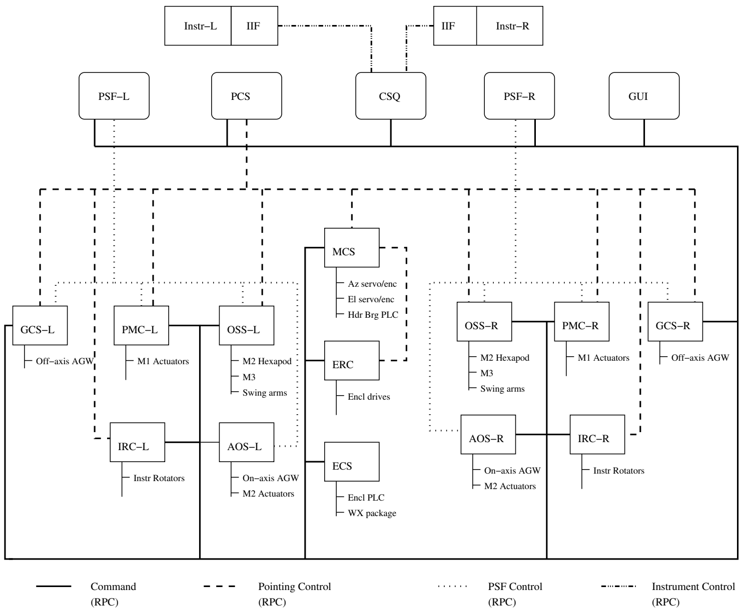

Figure 1. TCS Logical Overview of Subsystems and Controllers (Axelrod 2003)

2. Low-Level Infrastructure: Common Software Layer

This software layer provides the fundamental foundation upon which the TCS is constructed and functions. This layer is comprised of a series of services and structures which are accessed by the higher levels of the system.

System Data Dictionary and Network Shared Memory All system commands, variables and their corresponding attributes, and ancillary information are defined in a system data dictionary which enforces the concept of a single information source, allowing for a greater level of system integrity. Telescope status is organized by subsystem and maintained in a segment of network shared memory specifically designated for the subsystem. Collectively, the subsystem segments comprise a "telescope-wide shared memory" where each segment is propagated to all TCS platforms for a contemporaneous update of all the platforms.

Event Subsystem Events are the means of recording actions, as well as errors or abnormal conditions in the system. This mechanism provides a means to both generate events and to be notified via a callback when a particular event has occurred elsewhere in the system. Notification of events is only sent to subsystems which have subscribed to receive such events; the subsystem decides on the appropriate action to handle the event. All events are stored in a central logging archive for diagnostic purposes.

Telemetry Streams Subsystems may register telemetry streams, sets of time-sampled variables, which are delivered over the telemetry network to an archive.

Customized RPC Library The library was created to support a multi-threaded environment. Since communication is accomplished via XML strings, only strings need to be accommodated by the library.

Time Service A GPS synchronized time server will be present on the LBT ethernet and will offer time services to any client via ntp. Since the system design is intended to keep the ethernet load light, ntp should easily provide time accurate to 1 msec.

3. High-Level Interface: GUI

Each subsystem has a dedicated GUI, created with Qt Designer, which is comprised of a main window and ancillary support panels and dialogs. The LBT TCS GUI guidelines are defined by De La Peña (2003). Each GUI is an independent executable and

- *never* communicates directly with its corresponding subsystem or to any hardware,
- *never* communicates with GUIs for any other subsystem,
- issues its commands as customized RPC calls to the command sequencer (CSQ),
- receives a handle from the CSQ which it uses to poll the status of the issued command,
- obtains subsystem state and other critical information from the network shared memory, and
- registers itself with the event handling system for direct communication of alarms.

The GUIs have been designed such that they are instantiated in their "design-time" form (*disabled*). Only when status is successfully obtained from networked shared memory is the GUI *enabled* for use. A GUI can be aborted and restarted without any interference to/from the TCS. Each subsystem GUI has exclusive use of a monitor, giving rise to a TCS display wall. The display wall provides individuals in the control room simultaneous state information for all of the telescope subsystems. Finally, a standalone Matlab GUI application provides capabilities for plotting telemetry streams.

4. Commanding and Sequencing

4.1. Communication

Command and control of the telescope is done primarily through a series of GUIs which communicate directly with a centralized controller or command sequencer (CSQ). Commanding is accomplished by sending XML strings using a customized remote procedure call (RPC) library. The customized library properly supports a multi-threaded environment and only needs to accommodate the transfer of strings. The controller which initiates the command maintains a handle from the receiving system which can be used to determine the status of the command. System status (e.g., running, stopped) and state (nominal, alarm) are read from network shared memory.

4.2. Command Sequencer

The primary responsibilities of the CSQ are to validate incoming requests and to route the requests to the appropriate subsystems. Since a single command can blossom into a series of necessary actions which need to be assigned to a variety of the telescope subsystems (e.g., mount control, pointing control), a petri net technique is employed by the CSQ to handle the sequencing. Petri nets are well suited to model concurrent and discrete events, and therefore, are ideal for control system specification.

4.3. Subsystems

While subsystems control specific hardware and unique characteristics, all subsystems share common properties. These properties consist of: autonomous operation, heartbeat, remote reboot, subsystem network interfaces, subsystem API, and subsystem template. A few of these properties are discussed here.

Autonomous Operation For testing purposes, each subsystem must be able to work autonomously. While the entire TCS depends upon the common software layer, a minimal version of the common software should be able to support autonomous operation of a subsystem. The minimal common software has the following capabilities: manage subsystem configuration via the data dictionary and configuration files, log events to local log files, collect telemetry streams to a local archive, and use customized RPC services to communicate with local or remote clients.

Heartbeat As the TCS is a distributed system, a software implemented heartbeat is used to monitor the health of all the subsystem nodes. Each subsystem provides a "heartbeat" function which returns with minimal latency a system health code. In the event of an abnormal health code, the controllers monitoring the heartbeat, GUI and CSQ, will alert the telescope operator and client (respectively) that the subsystem should be restarted.

Subsystem API The subsystem API consists of two major components: commanding and getting/setting values. Commands are issued as customized RPC calls, and a handle can be retrieved from the server such that the commanding subsystem can monitor the progress of the issued command and continue to perform other tasks. All subsystems support a minimal set of commands which accommodate an orderly startup and shutdown of the subsystem.

All values which can be configured and all read-only state values for a subsystem comprise the keyword interface. The keywords define the external state of the subsystem, and are stored in the LBT System Data Dictionary as discussed in Section 2. The configurable values are accessible via accessor or modifier functions.

References

Axelrod, T. 2003, LBT CAN 481s001

De La Peña, M.D. 2003, LBT CAN 481s010a