Telescope Automation and Remote Observing System (TAROS)

Greg Wilson, Adam Czezowski, Gary R. Hovey, Mark A. Jarnyk, Jon Nielsen, Bill Roberts, Kim Sebo, Dione Smith, Annino Vaccarella, and Peter Young

Research School of Astronomy and Astrophysics, Mt Stromlo Observatory, Australian National University, Canberra, Australia

Abstract. TAROS is a system that will allow for the Australian National University telescopes at a remote location to be operated automatically or interactively with authenticated control via the internet. TAROS is operated by a Java front-end GUI and employs the use of several Java technologies - such as Java Message Service (JMS) for communication between the telescope and the remote observer, Java Native Interface to integrate existing data acquisition software written in C++ (CICADA) with new Java programs and the JSky collection of Java GUI components for parts of the remote observer client. In this poster the design and implementation of TAROS is described.

1. Introduction

TAROS is a remote observing system currently being implemented at the Mt Stromlo Observatory. The approach has been to use proven technologies and to aim for platform independence. TAROS will allow for telescopes at a remote location to be operated in fully automated, remote interactive, assisted remote interactive or local interactive observing modes.

- Automated observing is an observing mode in which the telescope and instruments are fully controlled by computer and observations are scheduled by the computer. Monitoring will be possible from SSO and remote sites.

- Remote observing is an observing mode in which an observer at a remote location has full control of the telescope and instrument, with no assistance from the staff.

- Assisted observing is an observing mode in which an observer, usually at a remote location, has full control of the telescope and instrument, with limited assistance provided by an observing assistant at the telescope site.

- Local observing is an observing mode in which an observer at the telescope site has full control of the telescope and instrument.

TAROS will provide for the integrated control of the telescopes independently of their mode of operation. From the observer’s point of view there is no difference between local and remote observing. TAROS will also control the flow and archiving of data obtained from instruments attached to the telescopes. It will
provide fully automated environmental monitoring to ensure that telescopes are only operated when appropriate. In what follows we provide an overview of the overall TAROS architecture and then focus on two of the more mature components of TAROS - the status and message server and the displayer sub-system.

2. Architecture

2.1. Technologies

TAROS is implemented using the programming languages C, C++ and Java. It is a distributed, multi-threaded and multi-process system. Processes communicate with each other using standard and reliable protocols. Technologies that are employed in TAROS include:

**Java Web Start** - for distribution and launching of the observer’s GUI.

**JMS** - Java Message Service, underpins a significant proportion of inter-process communication within TAROS, both point-to-point and broadcast;

**JNI** - Java Native Interface, allows Java and non-Java code to work together;

**JSky** - is a collection of reusable Java components for use in astronomy;

**RPC** - Remote Procedure Calls, provide a means for a process on one machine to call a process on another such that the programmer can treat both local and remote procedure calls in a like manner;

**XDR** - External Data Representation Standard, a standard used for the description and encoding of data across multiple hardware platforms;

**XML** - Extensible Markup Language, acts as the “specification language” for observation blocks.

2.2. Core Processes

The overall architecture for TAROS is shown in Figure 1. The main components are briefly described below:

- **Control** - this process starts the system and ensures that processes are still running.
- **Monitor** - this process monitors the health of sub-systems.
- **Gateway** - this process authenticates TAROS users and relays TAROS commands from the user to the other TAROS components.
- **Status, Health and Message Server** - provides a central repository of system status and health information which will be available to all other TAROS components. Both “message queue” and “subscriber-provider” models of message/data distribution are available.
- **Scheduler** - accepts requests for observation from external schedulers, including the interactive GUI. It interacts with the various TAROS sub-systems to adjust observation priorities and to initiate observations.
• **GUI** - Java based GUI that can be used from anywhere on the Internet.

• **Displayer** - supplies images taken by the data acquisition system to the observer’s image display tool. Images may be compressed to a level determined by the observer.

• **Telescope Control System** - The TCS will have its own internal mechanisms for startup and error recovery. It is responsible for all telescope hardware, handling telescope commands and maintaining status information such as interlocks, position and motion status (see Jarnyk & Hovey 2004 for more details).

• **Meteorological Station** - is also an independent subsystem that provides meteorological data. This information will be obtained by TAROS subsystems via the Met Client.

• **Data Distributor** - performs data “trickling” to remote observers and data archiving on various media types.

• **CICADA** - CICADA (Young et al. 1997, Young et al. 1999) is the current data acquisition software used at the ANU telescopes. TAROS has been
designed so that this software can be reused with a few modifications as possible.

3. Status, Health and Messages

In implementing a remote observing system it is necessary for the remote client software (and observer) to be informed of changes in telescope parameters as well as messages from the telescope control system and data acquisition software. TAROS provides this functionality. Sets of system parameters are defined using a subset of the Remote Procedure Call Language (RPCL - we do not include pointers or programs). This allows for parameters to be of any standard RPCL types such as arrays, structures, unions, and enumerated types. A shared memory database is used to store parameters in machine independent XDR format. Parameters are converted from XDR format to native machine format as they are retrieved.

The system requires a master database to be configured on one machine. Slave databases running on other machines can send a copy of the entire contents of their database or any structure within their database as an atomic update to the master database through the transfer of a block of XDR formatted memory. In this way it is ensured that telescope control system parameters are always kept as a self-consistent block in the TAROS master database.

The underlying database implementation is in C++, using RPC as a means of transferring parameter updates from slave databases to the master database. A Java Native Interface is also provided to allow full access to the database from Java applications running on the same machine as the master database. To facilitate access from remote Java clients (such as the GUI) a server application has been developed in Java. It receives incoming commands via a JMS queue and provides parameter and message updates via JMS topics. Parameter values can be explicitly requested from the server or the client application can elect to receive updates automatically whenever any relevant parameters change in the database.

4. Image Compression and Display

Real time image display can be a difficult problem when setting up a remote observing system. Depending on bandwidth and image size it can be impractical to download every image before the next exposure (for instance, an image of 140 MB\(^1\), would take over an hour to download using a 256Kb/s broadband connection). TAROS will use compression to enable users to view some/all of their images. The H-Compress algorithm (White & Percival 1994) is used to perform the compression. While this form of compression is lossy for compression ratios of around 20 and greater there is no significant effect on the astrometric and photometric properties of the uncompressed image. This enables observers to use the compressed image to measure seeing and line widths etc. It should

\(^1\)140 MB is the size of the largest image that our instruments generate currently, but future instruments will generate larger images.
be noted however that the images saved by the displayer client only represent quick look data. The observer receives the uncompressed images via download some time after the observation run.

The TAROS displayer is composed of two main components - an image server and a client:

The display server The server is mostly written in Java. It interfaces with the CICADA data acquisition system via JNI. The JNI interface receives the images in chunks as the image is being read out. Each chunk is compressed to the desired level and then forwarded to the client via JMS. Due to the nature of JMS this allows more than one client to receive the real time display images just by subscribing to the real time display topic.

The display client The displayer GUI is built from the JSky Java components. It is closely based on the JSkyCat tool. Essentially the client subscribes to the display JMS topic and receives the image chunks as they are published. The client then decompresses the image, updates the display and writes a copy to disk.

As well as the JSky Java package we have made use of several existing Java packages to simplify the implementation of the displayer code. To perform the H-Compression TAROS uses the Java H-Compression package provided by Jim Dowler\(^2\). The Java FITS package written by Tom McGlynn\(^3\) is also used.

5. Final Remarks

Currently the system is fully designed but only partially implemented. The full system is due to be completed in early 2006 in time for the new SkyMapper telescope. When completed TAROS will provide a reliable, maintainable and easy-to-use automatic and remote observing system.

References


\(^2\)http://cadcwww.dao.nrc.ca/software/hcompress/

\(^3\)http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java