中国科学院大学
University of Chinese Academy of Sciences

博士学位论文

基于LAMOST的观测控制系统研究

作者姓名：田园

指导教师：赵永恒 研究员 中国科学院国家天文台

学位类别：理学博士

学科专业：天文技术与方法

培养单位：中国科学院国家天文台

2018 年 6 月

The Research of the Observatory Control System based on LAMOST

A thesis submitted to the
University of Chinese Academy of Sciences
in partial fulfillment of the requirement
for the degree of
Doctor of Natural Science
in Astronomical Techmology and Method

By

Tian Yuan

Supervisor: Professor Zhao Yongheng

National Astronomical Observatories, Chinese Academy of Sciences

June, 2018

中国科学院大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明或致谢。本人完全意识到本声明的法律结果由本人承担。

作者签名：田园

日　　期：2018.5.28

中国科学院大学
学位论文授权使用声明

本人完全了解并同意遵守中国科学院大学有关保存和使用学位论文的规定，即中国科学院大学有权保留送交学位论文的副本，允许该论文被查阅，可以按照学术研究公开原则和保护知识产权的原则公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：田园　　导师签名：

日　　期：2018.5.28　　日　　期：2018.5.28

摘 要

LAMOST（大天区面积多目标光纤光谱望远镜）是我国九五大科学工程之一。它结构复杂、技术先进，大口径与大视场的结合使其拥有极强的光谱巡天能力，在大视场天文学研究上居于国际领先的地位。从2011年先导巡天开始到2017年发布DR5为止，已经观测、处理和发布了恒星、星系和类星体等天体光谱数据超过900万条，使之成为世界上光谱获取率最高的天文望远镜。

LAMOST拥有众多软硬件子系统，子系统各司其职以保证整个望远镜的天文观测。其中，观测控制子系统(OCS)是LAMOST软件控制的核心。OCS系统的主要任务是管理、协调和控制其它子系统的操作，使整个望远镜有条不紊地、按计划、有步骤地进行天文观测。

LAMOST当前OCS系统于2000年左右开始研制，2008年基本完成。对于国内天文学界而言，这是第一次在复杂的大型望远镜上研制观测控制系统。目前，OCS已稳定运行十年，为LAMOST光谱巡天计划的实施提供了重要的保障。但是，随着巡天计划不断推进，为了进一步提高观测运行效率，LAMOST对OCS提出了更高的要求，即：实现整个望远镜观测控制的自动化。现有OCS系统无法满足LAMOST观测控制自动化的新需求。

本文在深入分析当前OCS系统设计运行原理基础上，尝试将小型望远镜自动化性能优异的远程望远镜系统第二版(RTS2)软件框架引入到LAMOST的观测控制领域中，从而改善OCS的自动化观测控制能力，提高望远镜整体观测效率。具体内容如下：

首先，本文简单回顾了天文望远镜的发展历程，讨论了国内外大口径光学天文望远镜及其观测控制系统的特点。

然后，着重分析了LAMOST当前OCS系统的设计原理以及软件组织结构、运行机制、通信机制和所采用的关键技术，并归纳了其自身的优点与不足。在充分调研小型望远镜程控自动化发展历程的基础上，选定了自动化能力出众的RTS2软件框架，并对其运行机制进行了深入研究。

之后，设计研发了基于Python协程技术的LAMOST控制节点分布状态采集与监视系统，并将之映射为RTS2框架中的传感器设备模块。还对现有多相机集群控制软件进行重构，将之映射为RTS2框架中的相机设备模块。

最后，在完成了LAMOST观测计划文件信息入库以及光纤定位软件接口

向RTS2滤光片设备映射工作的基础上，搭建模拟测试环境，完成了基于RTS2软件的LAMOST观测控制实验。

实验的成功不但初步验证了将RTS2软件引入OCS系统从而提高LAMOST自动化观测控制能力的设想。而且，为下一步RTS2软件的全面引入打下了坚实的基础。

本文主要有以下几个创新点：

1）在深入分析LAMOST当前OCS系统与RTS2系统之间异同的基础上，首次提出了基于LAMOST的大型望远镜设备虚拟化的概念。这一概念涉及对天文望远镜通用操作流程的抽象、大型望远镜子系统(或设备)信息的分层与内聚、复杂软件系统重构等一系列重大问题的思考。这一概念也为LAMOST的OCS系统扩展与升级指出了明确的道路。

2）首次将自动化性能优越的RTS2系统引入大型望远镜观测控制领域。在将LAMOST多个子系统或软件映射成RTS2设备模块的基础上，搭建测试环境进行模拟自动化观测控制实验，从而验证了基于RTS2框架的LAMOST望远镜自动化观测控制的可能性。

3）设计和研发了基于Pyhton协程技术的LAMOST控制节点分布状态采集与监视系统。这一系统填补了LAMOST运维相关领域的空白。将Python协程这一新型技术引入望远镜观测控制领域也属于国内首创。

关键词： LAMOST，观测控制系统，RTS2，自动化控制

Abstract

The Large Area Multi-Object Fiber Spectroscopic Telescope (LAMOST) is one of major-science projects in the National Ninth Five-Year Plan period (1996-2000) of China. It has a complex structure and using advanced technologies. The combination of large aperture and large field of view gives LMAOST a strong spectral survey capability and makes it to take an internationally leading position in the fields of large scale and large sample astronomy and astrophysics.As the telescope having the highest spectrum acquisition rate in the world, LAMOST has already successfully observed, processed and released more than 9 million targets, from the pilot sky survey in 2011 to the DR5 release in 2017. These targets include stars, galaxies and QSOs.

LAMOST has many hardware and software subsystems. These subsystems perform their duties to ensure the astronomical observation of the entire telescope. Among them, the Observation Control Subsystem (OCS) is the "nerve center". The main mission of the OCS is to manage and cordinate all these subsystems, and make the astronomical observations, using LAMOST, are more methodical, planned and systematical. This is the first time in the Chinese astronomy history that a research and development project on the OCS of large scale telescope has been conducted.

The current OCS was developed around 2000 and was basically completed in 2008. The OCS has worked for nearly adecade, providing an important guarantee for the LAMOST's spectrual survey. However, with the development of the survey project, in order to improve the efficiency of observation, LAMOST has put higher demand on the OCS, namely, to realize the automation of astronomical observations. Obviously, the existing OCS cannot meet this new requirement.

Based on the in-depth analysis of the design and operation principles of current OCS, this thesis attempts to introduce the Remote Telescope System - 2nd version (RTS2) software framework, which has excellent performance on automatic observation of small telescopes, into the domain of observation control of the LAMOST, in order to improve the performance of the OCS in automatic observation control and improve the observation efficiency of the LAMOST. The main works of this thesis are listed as follows:

First, this thesis reviewed the history of the development of astronomical telescopes, and discussed the characteristics of large scale optical telescopes and their corresponding observation control systems.

Then, we analyzed the design principles, software organization structure, operation mechanism, communication mechanism of the current OCS system of the LAMOST, and key technologies used are analyzed, and its own advantages and disadvantages are summarized. After fully investigating the development history of small telescope automation, the RTS2 software framework with excellent automation capability was selected and its operating mechanism was deeply studied.

After that, we designed and implemented LAMOST control node distribution state acquisition and monitoring system based on Python coroutines technology, and mapped it as a sensor device module in the RTS2 framework. We also reconstructed the current multi-cameras cluster control software of the LAMOST, and mapped it as a camera device module in the RTS2 framework.

Finally, based on the completion of plan file information storage and the mapping of the fiber positioning interface to the filter device of RTS2 framework, a simulation test environment was established, and the experiment of automatic observation of the LAMOST based on RTS2 is basically completed.

The success of the experiment not only preliminarily verified the idea that we had previously introduced the RTS2 into the OCS to improve the automatic observation capability. Moreover, it laid a solid foundation for the next step of work in the further.

The innovations in the thesis as bellow:

(1) The concept of large scale telescope device virtualization was proposed , which based on an in-depth analysis of the similarities and differences between the current OCS and RTS2. This concept comprehensively considers a series of major issues, such as the abstraction of the general operational flow of telescopes, the layering and cohesion of large scale telescope's informationg and subsystems (or device), and the reconstruction of complex software system. This concept also points out the clear path for extension and upgrade of LAMOST OCS system.

(2) For the first time, the RTS2 system with superior automation performance was introduced into the observation and control field of large scale telescopes.Based on the mapping of LAMOST subsystems or software into RTS2 device modules, a test

environment was set up to simulate automated observation and control experiment. And the experiment preliminarily verified the possibility of automatic observation of the LAMOST based on RTS2.

(3) Designed and implemented the LAMOST control node distribution state acquisition and monitoring system based on Python coroutines technology. This system fills the gaps in the operation and maintenance related fields of the LAMOST. And it was the first time in China that the Python coroutine technology has been introduced into the telescope observation control field.

Keywords: LAMOST, OCS, RTS2, Automatic Control

目 录

图形列表

表格列表

第1章　绪论

天文学作为自然科学六大基础学科之一，主要研究探索宇宙及其所包含的所有天体的本质，包括天体在宇宙空间的位置、分布、物理状态、化学组成、运动和演化过程等。它是一门有着悠久历史的古老而常新的学科，是随着人类生产实践活动的需求而产生和发展的。

天文学与其他自然学科不同之处在于，天文学的主要实验方法是被动的天文观测。通过天文观测来收集天体的各种信息。因而，对观测方法和观测手段的研究始终是天文学家努力的方向，也是推动天文学发展的动力和源泉。观测天体的重要工具是天文望远镜。可以毫不夸张地说，没有望远镜的诞生和发展，就没有现代天文学。随着望远镜在各方面性能的不断改进和提高，天文学也正经历着巨大的飞跃，迅速推进着人类对宇宙的认识。

1.1　天文望远镜的发展

1609年，意大利天文学家、物理学家伽利略发明了人类历史上第一台天文望远镜，它是有一片平凸透镜和一片凹透镜组成的折射式望远镜，如图1.1所示。伽利略利用天文望远镜先后发现了月球表面的山和谷、木星的四颗卫星、土星光环、金星盈亏、太阳黑子、银河系是由许多恒星组成的一系列重要的天文成果。从此，人类对世界和宇宙的认知走入了全新的时代。

图 1.1 伽利略折射式望远镜

此后的400多年中，随着自然科学技术各个分支的突飞猛进，天文望远镜所采用的相关技术也不断发展和更新。特别是以计算机技术、信息技术、电子技术、空间技术、材料技术为代表的新科技革命的进行，对天文望远镜技术起到了极大的推动作用。

总体上来讲，现代天文望远镜技术的进步主要体现在以下三个方面(苏定强, 2008)：

一、口径越来越大，观测能力越来越强。更大的口径意味着更强的光接收能力，也就意味着能够观测到更深、更暗的天体。因此，从最初伽利略制作的4.2厘米口径望远镜开始，人类就在不断尝试增大天文望远镜的口径。但望远镜口径会同时受到光学系统相差、面形误差、大气视宁度、衍射极限以及设备自重等多个方面的制约(苏定强 等, 2001)。在不断尝试新技术、新材料、新方法的基础上，人类已经建成了多台十米级天文望远镜，并为研制和建造30 - 50米级光学望远镜创造了可能。当前已建成或在建的大型光学望远镜如表1.1所示。

表 1.1 大型光学望远镜一览表

名称	口径(米)	国家(组织)	建成时间(年)
Gemini	2 × 8.3	美国、智利	2000, 2002
SALT	11	南非	2005
LBT	2 × 8.4	美国	2005, 2008
GTC	10.4	西班牙	2006
LAMOST	3.6 4.9	中国	2008
VLT	4 × 8.2	欧南台	1986～2012
LSST	8.4	美国、智利	建设中
GMT	21.4	美国、澳大利亚等	建设中
TMT	30	美国、加拿大等	计划中
E-ELT	39.3	欧南台	计划中

二、从单一可见光向全波段观测发展。天文望远镜初期发展全部集中在可见光波段。直到1931年，美国新泽西州贝尔实验室无线电工程师央斯（Karl Guthe Jansky）用定向天线研究越洋无线电话在短波上受到的静电干扰时意外地发现了来自银河中心方向的天体的无线电波。从此，为人类天文观测领域打开了一扇新的窗口。随着新技术的不断应用，天文观测频段不断扩展。如今，天

文观测以发展到由光学波段向高频的紫外、X射线、γ射线，以及低频的红外、射电等全波段观测，从而为人类更全面的观测宇宙内在面貌提供了可能。

三、从地面观测发展到空间观测。天文望远镜发明以来，地面天文观测始终是其唯一的方式。地面观测过程中，除了要受到自然气候影响之外，大气扰动以及大气的辐射和吸收等问题也影响最终的观测效果。1957年，苏联成功发射第一颗人造地球卫星，开创了空间观测的新时代。以1990年美国发射了著名的哈勃空间望远镜（Hubble Space Telescope，HST）为例，该望远镜口径2.4米，由于运行于近地轨道，不受大气层影响，因此效率更高、性能更好。经过历年来多次维护与升级，该望远镜已经获得了大量暗弱天体数据，在宇宙年龄、恒星形成演化、黑洞、暗物质等许多研究领域发挥了重要作用。下一代詹姆斯韦伯太空望远镜（James Webb Space Telescope，JWST）由美国航天局、欧洲航天局和加拿大航天局合作开发，目前已处于最后组装和测试阶段。作为哈勃空间望远镜的继任者，该望远镜质量（6.2吨）仅为HST的一半，而口径达到6.5米，接收能力为HST的5倍以上，预计将于2020年5月发射升空。相信届时该望远镜会为人类深入的了解宇宙概括和细节带来极大的帮助。

1.2 国外大型光学天文望远镜及观测控制系统

如上文所述，随着科学技术的不断进步，天文望远镜正在朝着大型化和复杂化方向发展。而对于天文望远镜的控制，也经历了由人工操作到计算机辅助，最终向自动化智能化方向发展的不同阶段。

最初，望远镜控制均由操作者手动完成。在整个观测过程中，每个步骤都要人工操作。由于口径不断增大，望远镜需要的操作人员也越来越多，而且操作者之间的协作也存在诸多限制，因此该方式效率很低。随着计算机技术的不断发展，计算机辅助控制开始引入到大型望远镜控制领域。这不但减少了操作人数，还极大地提高了运行效率和控制精度。早期的望远镜计算机辅助控制软件功能单一、结构松散，望远镜每个部件都有单独的软件提供给观测者来进行操作。望远镜整个观测过程就是由观测者和不同控制软件之间各种交互来完成的。随着望远镜逐渐巨型化，其软硬件也开始逐渐变得复杂，因此，早期功能单一的计算机辅助控制软件也开始变得捉襟见肘。为了进一步提高观测效率，人们急需要一种建立在辅助控制软件之上的总体控制软件，用它来来协调和调度各种辅助控制软件，从而完成望远镜整体运行。这套软件系统甚至可以部分或全部代替人工来完成复杂情况分析判断。大型天文望远镜的观测控制系统

（Observatory Control System, OCS）应运而生。

1995年，Gemini望远镜的设计者们在ADASS（Astronomical Data Analysis Software and Systems）第5届年会上首次明确提出了观测控制系统的概念(Gillies et al., 2007)。近年来，随着国际上一些大型天文仪器的建成和巡天计划的开展，OCS也获得了长足的发展和进步。简单来说，OCS的主要目标就是以计算机技术来完成大型望远镜观测中的常规任务，例如：观测目标的选择、观测步骤的协调、观测中所涉及到的望远镜各组成部分的控制、数据的采集存储与处理、环境的监控等。利用OCS可以极大的提高望远镜使用效率和科学产出能力。随着望远镜越来越复杂化以及计算机技术的不断发展，OCS已成为望远镜系统中必不可少的一部分。

下面将简要介绍几个大型光学望远镜及其观测控制系统。

1.2.1 SUBARU望远镜控制系统

SUBARA望远镜由日本国家天文台建造，坐落于夏威夷毛纳基山的山顶，海拔4139米。主镜直径8.2米，焦距15米，采用报价薄镜面、主动光学和自适应光学技术。1999年建成，总耗资3.7亿美元。如图1.2所示。

图 1.2 SUBARU望远镜

SUBARA望远镜的观测控制系统采用C++和Python语言混合编写，是一套轻量级基于任务的面向对象的分布式系统。系统提供任务抽象基类，用户（操作者）继承并实现具体任务类。任务类实例之间可以相互嵌入，并支持串行和并行执行方式，从而实现整套系统的调度和控制能力(Jeschke et al., 2008)。由于采

用面向对象设计，因此用户对基本任务具有较高的扩展能力。

1.2.2 LBT望远镜控制系统

LBT望远镜坐落于美国亚利桑那州东南格雷厄姆山（Mount Graham），海拔3190米，是美国亚利桑那大学格雷厄姆山国际天文台的一部分。它使用两个紧邻的8.4米口径圆形反射镜，等效通光口径11.8米。2004年10月第一反射面出光，2008年1月开始双反射面同时工作。如图1.3所示。

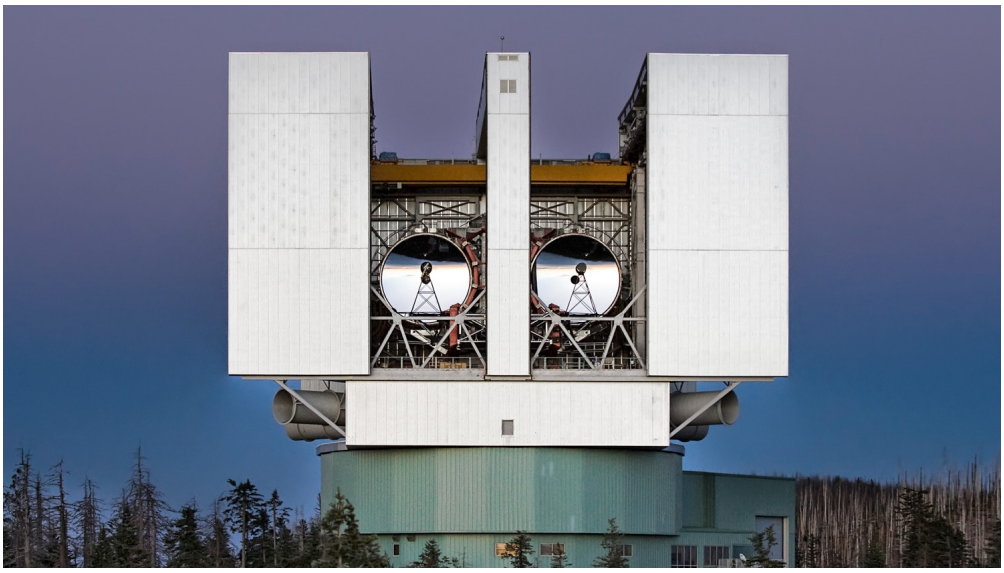

图 1.3 LBT望远镜

LBT观测控制系统采用网络分离技术，根据任务不同将分离不同子网，如：控制网络、存储网络、测量网络、仪器网络等。

底层架构中，系统支持一系列服务和结构。系统命令、变量、属性信息均采用系统数据字典形式存储，各子系统状态采用网络共享内存方式提供给整个望远镜(Axelrod et al., 2004)。各子系统状态转换、异常等均产生统一格式的事件，通过订阅分发机制有需求的子系统可以收到关注的事件并进行相应处理。整个系统采用多进程方式实现，命令控制图形用户界面（Graphical User Interface, GUI）与命令序列器通过远程过程调用方式实现进程间通信。GUI负责人机交互，命令时序器负责对请求有效性进行校验和分发请求。

1.2.3 LSST望远镜控制系统

LSST望远镜坐落于智利北部帕穹山伊尔佩恩峰上，海拔2682米。它是一个广视场反射式巡天望远镜，采用最先进的32亿像素科学级CCD成像相机，并采用新颖的三镜设计，主镜口径8.4米，可以获得3.5度视场。该望远镜目前正在建

设过程中，其设计图如图1.4所示。

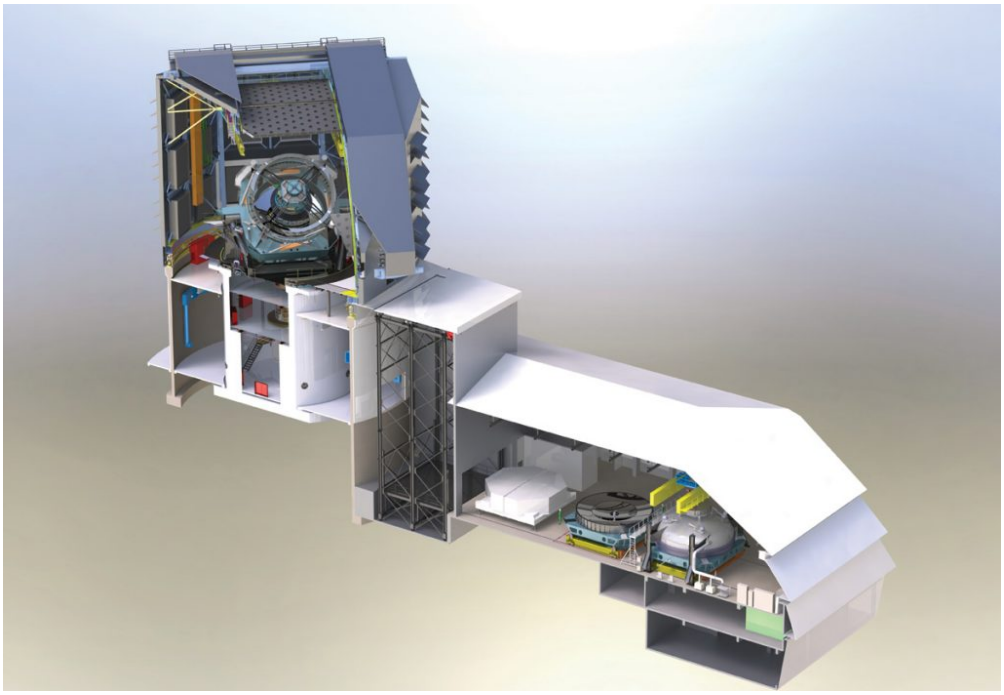

图 1.4 LSST望远镜构想图

LSST望远镜观测控制系统采了用数据分布服务（Data Distribution Service, DSS）中间件作为系统消息总线来连接各个子系统，具体使用开源OpenSplice软件实现，如图1.5所示。

采用OMG定义的SysML5语言建模，支持发布/订阅架构。多子系统实现中，支持多种语言（C/C++、Java、Python、LabVIEW等）运行时支持机制：系统以XML定义核心抽象对象（命令、事件等），利用通用解析器和通用库生产器将XML转换成本地子系统应用库(Daly et al., 2016; Mills et al., 2016)。

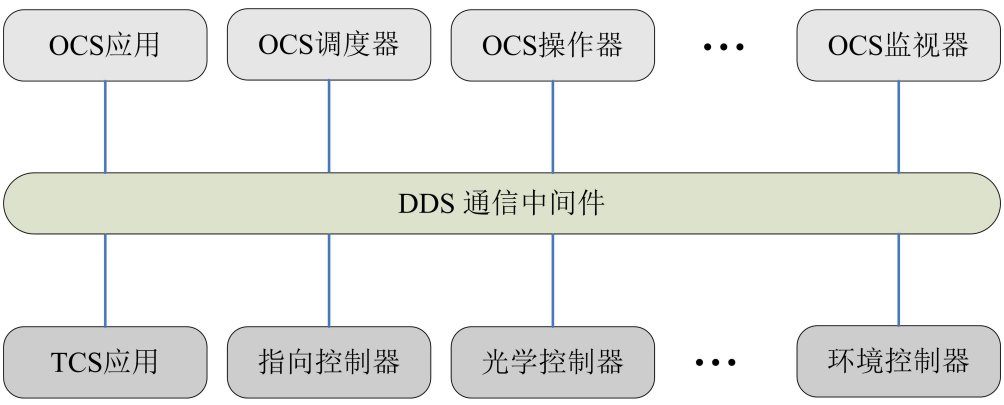

图 1.5 LSST观测控制系统简图

1.3 LAMOST与观测控制系统

LAMOST（Large Sky Area Multi-Object Fiber Spectroscopy Telescope）望远镜是由中国科学院承担的国家九五大科学工程。LAMOST坐落在国家天文台兴隆观测站，是一架横卧南北方向的中星仪式反射施密特（又称王-苏型）望远镜(Wang et al., 1996)。其光学系统包括由24面子镜拼接而成的反射施密特改正镜MA、由37块子镜拼接而成的球面主镜MB以及直径1.75米的焦面三个部分(Shi, 2015)。而主动光学技术的引入使天体暗弱光线能够完美地汇聚到焦面上。在焦面板径向2/3处对称安装着四个导星相机用于为望远镜的指向和姿态提供校准信息，当望远镜发生指向偏差或焦面失焦情况时，望远镜控制系统会根据校准信息进行精确地修正，从而达到最佳跟踪效果。LAMOST光路示意图如图1.6所示。

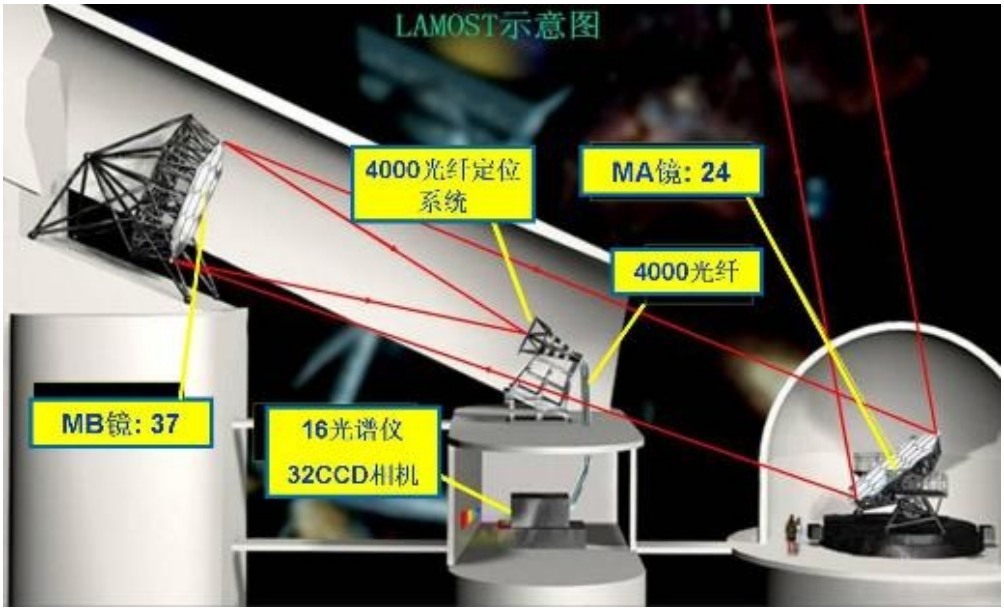

图 1.6 LAMOST光路示意图

LAMOST采用先进的并行可控方案以及双回转光纤定位单元技术使安装在焦面板上的4000根光纤快速准确的指向各自的观测目标。所有光纤分成16个组（每组250根）分别引入16台光谱仪。光纤出射光经过光谱仪分光镜后分成红蓝两束，再分别经过光栅和改正镜改正后投射到4096×4136像素科学级CCD上。红蓝两端的波长覆盖范围分别为3700至5900埃和5700至9000埃，光谱分辨率分别为2.5埃和4埃左右，相当于在5000埃处和7000埃处的分辨率约为1800(崔向群, 2001)。

LAMOST有一套自己的输入星表，观测计划由巡天战略系统的软件自动产

生产生。进行分配的。巡天源按r星等被分为亮、中、暗三种天区。每个天区一般会曝光三次来获得足够的信噪比和去除宇宙线，一般来说亮、中、暗源的每次曝光时间分别为600秒、900秒和1800秒。除对目标源的曝光之外还有其他类型的曝光，包括：本底流量用来消除CCD读出电路噪声；天光平场用来消除光纤效率差异；汞镉氖氩波长定标灯用来对进行波长定标。

LAMOST具有明确的科学目标(赵永恒, 2015; Q. et al., 2012)，主要包括三点：第一个是宇宙和星系，包括星系红移巡天和星系的物理特性研究，用于研究星系的形成和演化、宇宙大尺度结构以及暗物质和暗能量等前沿课题。第二个是恒星和银河系的结构特征，通过观测更多更暗的恒星来了解银河系中恒星的分布和运动情况以及银河系的结构。第三是天体光谱的多波段证认，通过LAMOST 获得的光学波段光谱来确认其他波段比如射电、红外、X射线、γ射线发现的天体，共同研究单一波段无法研究的前沿课题。

LAMOST采用巡天的工作模式，巡天周期为5年。2011年10月开始进行了为期一年的先导巡天之后，2012年9月开始正式巡天(Zhao, 2014)。迄今为止，最新的版本数据集是2017年发布的DR5，包含了901.7万条天体光谱。

从以上介绍可以看出，LAMOST是一套精密而又复杂的大型望远镜，因此其观测控制系统的设计也十分巧妙。

LAMOST观测控制系统运用面向命令的控制方式，采用松散的分层体系结构设计，并使用模型-视图-控制器（Model-View-Controller）模式，将用户界面和实现相分离。观测控制系统从下到上分为功能层、应用层和界面层。功能层实现了观测控制系统的底层功能，包括有通信、日志记录、出错处理、数据库读写、文件读写、进程间通信等功能；应用层使用功能层的功能实现了基本的组件，主要有消息总线、命令执行器、系统代理、日志记录器、状态分析器等，并定义了组件之间的协作关系；界面层提供了用户界面，方便了观测人员的使用。详情参见第2章。观测控制系统目前已稳定运行超过十年，为LAMOST稳定观测和运行做出了重要贡献。

随着计算机技术的不断进步，以及巡天计划的不断推进，LAMOST对观测控制系统提出了更高稳定性、更强大的自动化观测等新的要求。

1.4 RTS2简介

RTS2（Remote Telescope System - 2nd version）是一套完整的集观测计划制定、观测任务调度、观测控制、数据生成与发布为一体的自主天文台管理软

件(赵永恒, 2012)。它最初应用于BOOTES系列望远镜上，可以说是为BOOTES项目量身定做的软件系统。目前，RTS2系统已经应用到全世界20多台中小望远镜上，遍布亚、非、欧、南北美各个大洲。前文提到的LSST项目的CCD测试工作目前也在RTS2系统下进行(Kubánek et al., 2006)。软件采用C++开发，模块化设计，在Linux系统下安装和运行，并且源代码对外开放。RTS2自2001年左右开始开发，经过十多年发展，积累了大量的实际经验。代码的反复修改、锤炼，使得其设计目标进一步明晰，运行状态更加稳定。相比于其它软件系统，RTS2更适合中小型自主天文台，经过适当的二次开发，也能够应用于大型天文项目。

RTS2最初始的设计目标就是为中小型自主天文台提供一套能够观测γ暴现象的软件系统。因此项目从一开始就有其自身的特点：必须完全的自动化；必须对最新的γ暴警报做出最快反应；必须在没有γ暴观测时能够执行其它的观测任务; 必须模块化设计，以便硬件设备切换等。随着项目的推进，额外的功能需求不断增加。尤其当RTS2系统移植到其它项目的望远镜时，产生了一些新的技术要求：系统必须具有足够强的鲁棒性，即使在一些非关键性单元崩溃时，系统也能够连续的运行；系统必须能够被观测人员远程控制；系统必须可以完全用配置文件来配置; 系统必须对当前正在工作的部件提供清晰的状态描述信息；设备启动失败时，必须能够向观测者提供一份故障清单；代码应该包括虚拟设备驱动，用来对软件进行没有硬件情况的测试(Kubánek et al., 2012)。可以看出，在实际的开发过程中，随着经验的积累，RTS2本身功能也在不断的完善。

RTS2秉承“即插即用”的设计理念，所有设备在接入RTS2所在服务器时，都可以随时地开启和关闭。模块化设计使得系统的各个部分彼此独立，这为每个设备的单独控制和调试带来了方便。另外RTS2被设计成基于Linux用户空间的模式，不涉及到操作系统的内核部分，有利于开发和安装时避免过多代码库的依赖(李建 等, 2013)。

从 RTS2 的代码上看，RTS2具有逻辑清晰的类继承关系，所有天文台管理行为只定义为几个基础的类。其他的所有硬件驱动、守护进程等都从这几个基本的类出发，继承相关特性。通过这种类继承的关系，RTS2的控制命令可以共享参数，达到一定程度的内部通信的目的。由以上RTS2介绍可以看到，RTS2所具备的许多内在优点非常适合于对LAMOST现有观测系统自动化性能提升。而RTS2本身提供的二次开发接口也为将其移植到LAMOST这类大型天文望远镜的观测控制系统提供了可能性。

1.5 本文的结构

本文的主要内容是研究基于RTS2软件框架的LAMOST观测控制系统的实现。第一章对研究背景和方向做了简要介绍。第二章详细分析了LAMOST现有观测控制系统的实现以及RTS2软件框架的机制。第三章描述了基于RTS2软件框架的LAMOST集群计算机硬件资源监控的实现方法。第四章描述了基于RTS2软件框架实现LAMOST的多相机集群控制的实现方法。第五章介绍了基于RTS2软件框架的LAMOST观测计划入库以及光纤定位控制接口实现方法，并进行了基于RTS2软件的LAMOST自动化观测模拟测试。第六章是对本文的总结和未来的工作展望。

第2章 LAMOST观测控制系统与RTS2软件框架

LAMOST是一个高度自动化的、也是相当复杂的大型系统。这个大型系统是由多个子系统组成的，只有将各子系统有机地组织起来，协调一致地执行子系统的功能，才能可靠和有效地完成观测任务。因此各子系统之间需要有一个能联系、协调和控制各子系统操作，使望远镜系统有条不紊地、按计划、有步骤地进行天文观测的观测控制系统（Observatory Control System简称OCS）。

OCS是LAMOST观测运行的神经中枢，是一个高度自动化的、复杂的大型望远镜观测控制系统。它的主要任务是管理、协调和控制各子系统操作，使整个望远镜系统有条不紊地、按计划、有步骤地进行天文观测。同时，OCS还是一个通过子系统之间、运行模块之间的软硬件接口构成的多层次的、集中与分散相结合的观测控制系统(赵永恒, 2000)。

在OCS的控制下，LAMOST从SSS系统制作好的众多观测计划中选择适合的观测计划文件，驱动望远镜同时跟踪和观测4000个目标，并获取观测天体的光谱，其数据量巨大（每夜数十G字节）。如此强大的观测能力要求OCS系统不仅应具备使望远镜的各子系统协调一致对所观测的4000个目标进行定位和跟踪的能力，同时还应该具备同时记录4000个目标的位置信息、光谱数据、修正数据、观测条件等数据的能力。除此之外，OCS还需要具备根据观测环境的变化、观测设备状态的变化以及出现的意外事件进行相应的实时处理的能力。因此，OCS作为LAMOST软件系统的重要组成部分，在观测运行和数据管理方面都具有着重要地位。

2.1 LAMOST软件体系结构

LAMOST当前的OCS系统为2000年至2012年由国家天文台与中国科技大学共同研制开发。2009年LAMOST望远镜开始试运行后，软件框架基本稳定，细节功能随着巡天计划的不断深入而逐渐完善。现行OCS为保证巡天计划顺利进行做出了重要贡献。

要理解现行OCS软件设计和实现，首先需要了解整个LAMOST软件体系结构，以及OCS软件在其中所起到的主要作用。

由于LAMOST本身的复杂性导致其子系统软件众多，且由于底层子系统与硬件结合比较紧密，因此，各个子系统由不同单位来负责开发与维护。这种现

实情况为设计和开发OCS软件带来了一定的难度。但是，由于望远镜从设计伊始就紧跟国际前沿，开发团队已意识到OCS对于大型望远镜的重要性，因此已预留了OCS的设计位置。各个子系统开发单位在底层软件开发过程中，经过大量的协商后，均采用了统一的对外通信协议和相似的接口，因此为OCS的设计和实现提供了基础和可能性。

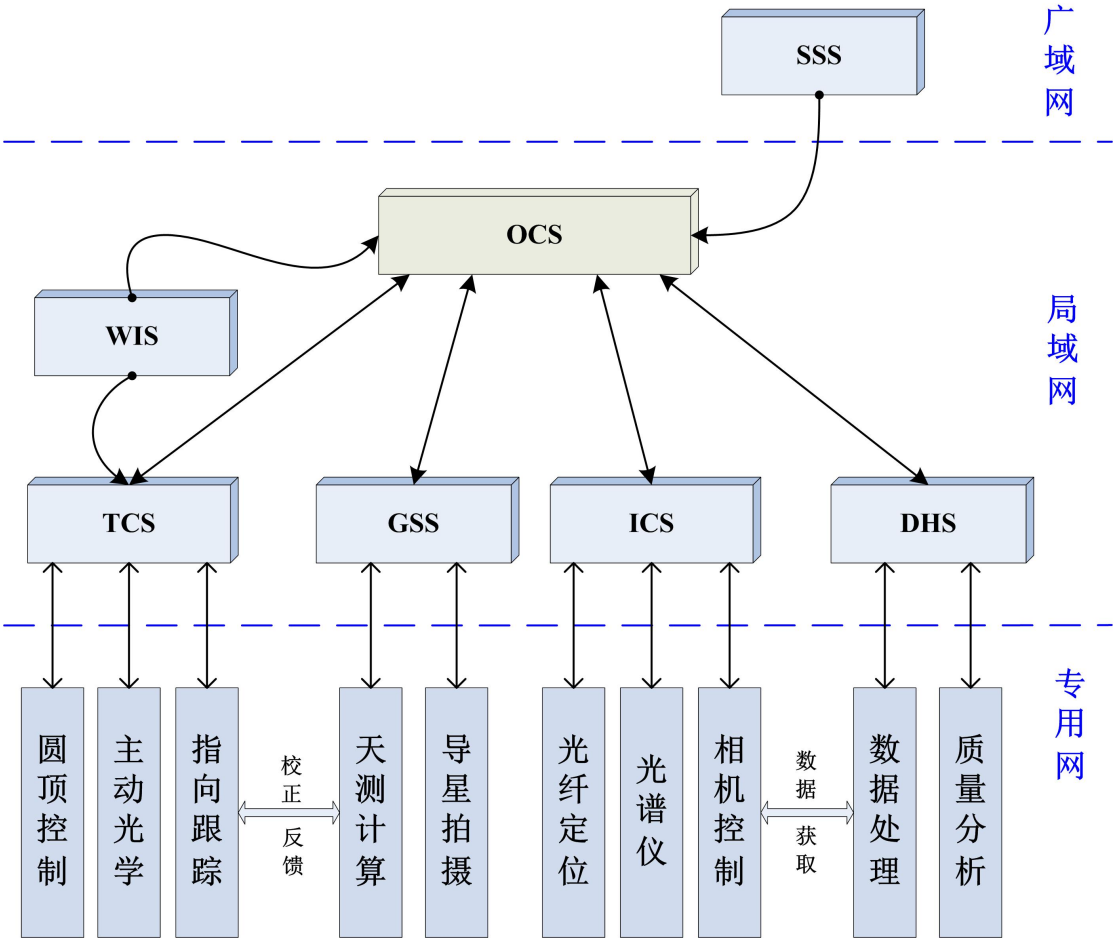

图 2.1 LAMOST软件体系结构示意图

图2.1描述了当前LAMOST运行过程中所涉及到的子系统和软件。从图中可以清晰的看出，整个LAMOST软件体系采用了分层设计思路，各子系统层次比较清晰，分工非常明确，这些有利因素为实现OCS带来了很大的便利。

在图2.1中，各子系统功能如下所述：

望远镜控制系统（Telescope Control System，TCS）主要负责圆顶控制（包括圆顶和焦面门）、望远镜的指向和跟踪（MA机架控制和焦面机构姿态调整）、主动光学控制和矫正（MA子镜MB子镜力促动器位移促动器的控制和主动光学矫正）。

导星系统（Guiding Star System，GSS）负责控制八台导星相机拍摄照片并进行天测计算从而为望远镜指向跟踪提供实时校正服务。

焦面仪器控制系统（Instrument Control System，ICS）负责分别控制光纤定位子系统、光谱仪控制子系统以及CCD相机集群控制。

数据在线处理系统（Data Handling System，DHS）提供存储光谱数据、在线的数据初步处理和离线的光谱质量初步分析。

巡天战略系统（Survey Strategy System，SSS）负责根据LAMOST巡天总体规划选择观测天区和观测对象并提供每次观测需要使用的具体观测计划文件。

气象环境辅助系统（Weather Information System，WIS）负责控制小圆顶气象站设备，检测当前环境信息并向OCS和TCS提供监测的环境数据以供参考。

如图2.1所述，在网络结构方面，TCS、GSS、ICS、DHS等子系统内部使用专用网，以尽量减少网间干扰。TCS、GSS、ICS、DHS、WIS和OCS之间使用LAMOST局域网。SSS系统目前采用离线生成观测计划列表的方式运行，其主机并未坐落于LAMOST局域网内，因此OCS与SSS之间采用广域网方式连接。

从图中可以看出OCS在LAMOST软件体系结构中的核心地位，整个系统的协调运行必须通过OCS来完成(罗阿理 等, 2012)。基于LAMOST的特点，OCS它必须具有如下的几个特点：

分布性：由于望远镜本身的复杂性以及各子系统需要完成对许多操作必须要并行处理，因此，OCS软件相应的也应具有空间和功能上的分布性。

实时性：LAMOST观测过程中无论是指向跟踪、主动光学、导星校正还是数据采集均需要很高的空间精度和时间精度要求。因此作为整个系统的调度者和协调者，OCS也必须具备较高的实时性。

可靠性： LAMOST作为国际领先的大型光学望远镜，其稳定高效运行必须得到保障，任何无谓时间的浪费都是不应该出现的，而任何故障性停机其后果更是不堪设想的。

开放性：一是，OCS的设计要有一定的前瞻性，考虑未来需求。二是，望远镜的寿命通常比电子产品更新的周期长得多。为了保证现有的投资能够延续使用，在设计时必须优先考虑采用具有开放性和可扩展性的技术和电子产品。

考虑到以上特点和要求，基于计算机软件控制技术，为了实现LAMOST观测控制以及巡天战略的顺利实施，OCS的设计与研发立足于当时的软硬件技术水平提出了总体设计实现思路：

以软件复用、模块化、松散耦合等思想为原则，以工业标准的公共对象请求代理体系结构（Common Object Request Broker Architecture，CORBA）中间件技术为框架，以C++语言为基本实现语言，采用了通用的通信库Socket进行网络编程，主流的图形界面设计软件QT3编写人机交互界面，并且以MySQL作为数据库(王坚 等, 2005)。

2.2 OCS软件层次与框架

如上一节所述，由于LAMOST自身的复杂性，且OCS设计中需要尽量满足分布性、实时性、可靠性和开放性，因此，OCS软件本身设计的较为复杂，充分利用了当时流行的各种先进的计算机技术。

为了尽量完美的实现OCS设计需求，内部采用组件组合的方式来设计和实现软件。从功能上讲，每个组件独立负责一项功能，组件之间采用CORBA中间件作为交互渠道，这样设计既便于各组件的独立开发，又为后期维护升级减少了很多工作量。

从实际运行情况上讲，每个组件作为一个独立运行进程，这些进程可以分布于同一台服务器或不同服务器上，从而提高了整套软件的稳定性和可扩展性。

从用途上讲，各种组件又可以分为前台交互式组件层、后台应用组件、后台服务式组件以及底层代理组件等(王坚, 2003)。

图2.2描述了整个OCS软件的内部组件体系架构。在这个构架中，采用分层的体系结构，以提高系统的松散耦合性，并提高系统的可扩展性和可维护性。

在负责人机交互的界面层，为了实现用户接口和领域实现的分离，采用了先进的模型/视图/控制器（Model/View/Controller）模式框架。考虑到视图/控制器经常在一起，所以在图2.2中把视图/控制器（V/C）合在一起作为最上层和用户通讯，构成系统的界面层。在界面层下面是OCS系统的应用层和服务层，而同各个子系统的通讯则由代理层来完成。由于采用分层结构，整个OCS软件系统设计变得目标清晰。

层与层之间仅以逻辑功能加以划分，而并非以物理接口加以区别。所有组件仍以CORBA中间件作为通信层，理解了CORBA中间件运行机制以及两个组件之间通过CORBA进行信息交互的机制后，推而广之便能顺利掌握整个OCS运行原理。

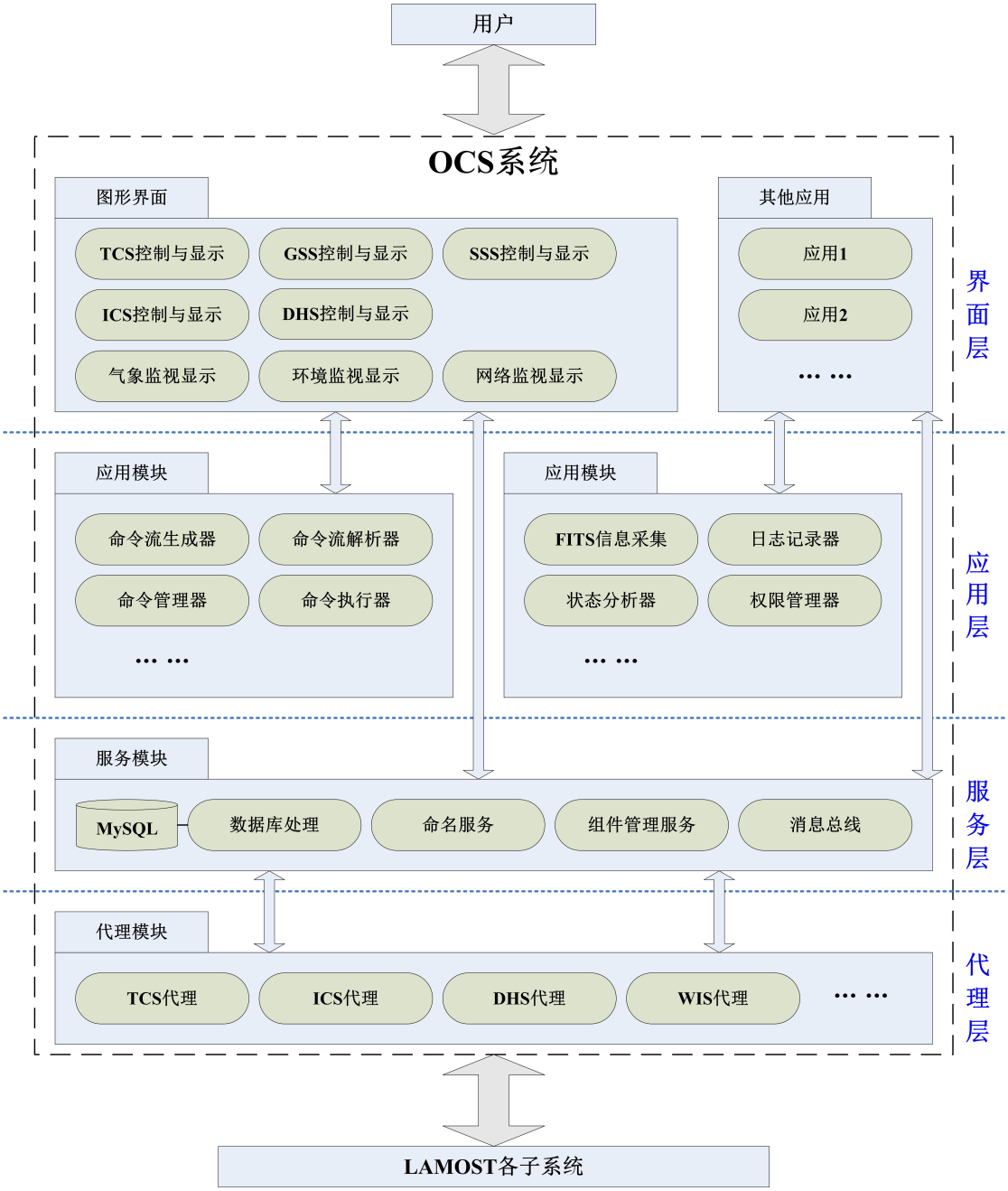

图 2.2 OCS系统的组件架构图

2.3 OCS软件关键技术

通过了解OCS软件组件体系结构可以使我们更好的理解其整体设计和实现思路。下面，将进一步介绍OCS各组件在设计和开发过程中所采用的先进技术，从而彻底理解现行OCS运作机制，并为下文所论述的基于RTS2框架对现行OCS进行改造的课题打下基础。

2.3.1 CORBA中间件与OCS组件

如前文所述，OCS所有组件均已CORBA中间件作为交互介质，这样的设计可以屏蔽组件之间的差异，简化各组件的设计。

CORBA是对象管理组织（Object Management Group，OMG）组织制定的关于分布式对象计算的标准，它和微软公司推出的组件对象模型/分布组件对象模型标准（Component Object Model/Distributed COM，COM/DCOM）标准以及SUN公司推出的EJB（Enterprise Javabean）标准并称三大面对对象组件技术。CORBA可以做到与实现语言无关，与运行平台无关，因此被称为“异构网络环境下的软件总线”(OMG 等, 2002)。

CORBA充分考虑分布式运行环境，基于CORBA技术的组件（服务）既可以在单机的模式下运行，也可以分布式环境下运行，因此，OCS系统中的所有组件可以在同一机器上运行，也可以在不同机器上运行。

为了适应不同语言环境，CORBA提供了接口定义语言（Interface Definition Language，IDL），IDL为描述性语言，用于抽象地描述对象接口，与具体实现语言无关。确立了IDL对各种对象接口的描述文件后，利用CORBA提供的各种语言翻译器将IDL文件翻译成相应源码。以图2.3为例，在完成IDL接口文件后可以利用CORBA的C++翻译器分别为客户端和服务器端翻译出Stub和Skeleton源码，再结合具体编程环境即可获得可运行程序。

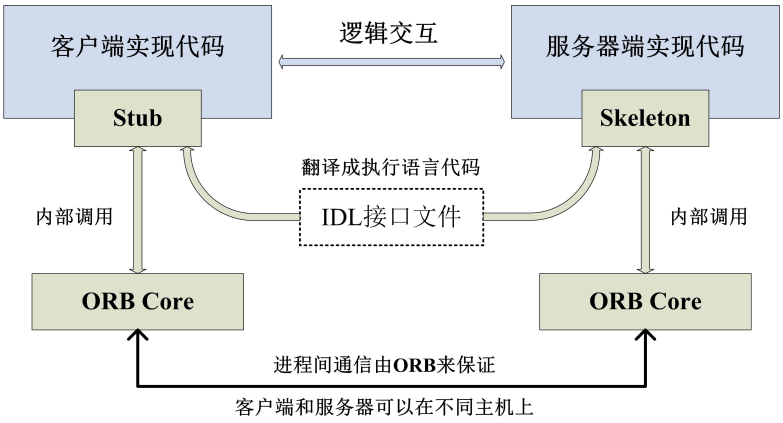

图 2.3 CORBA运行机制图

为了适应异构网络环境，CORBA提供对象请求代理（Object Request Broker，ORB）来屏蔽操作系统和网络差异。如图2.3所示，用户编写客户端和服务器端代码时，只需考虑逻辑交互即可。利用CORBA技术，用户将精力集中于代码业务逻辑，而实际通信过程由Stub/Skeleton与ORB核心库之间交互来完成(Henning

et al., 1999)，从而极大的简化了复杂系统的开发难度。而且，当扩展和维护单个组件时，只要接口文件不变，可以用一个新的实现替换旧的实现而不影响整套软件的运行。

CORBA作为一个标准，存在多种实现，既有商业版本，也有非商业版本。现行OCS采用的是非商业化的开源版本TAO （The ACE ORB）。TAO是基于可移植通信环境库（The Adaptive Communication Environmen，ACE）的一个ORB实现，其性能相对较高。

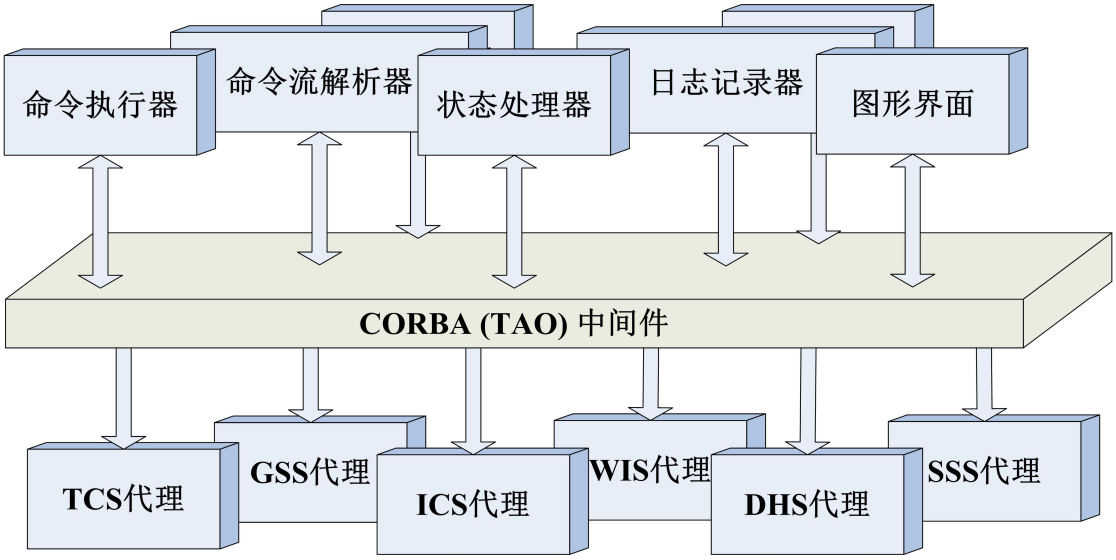

图 2.4 OCS组件采用CORBA(TAO)中间件交互示意图

图2.4是OCS各组件基于CORBA中间件交互的示意图。如图所示，OCS系统中CORBA中间件像一个多孔的大型插座一样将各自独立的OCS组件粘合成一个整体，具有独立功能的组件如同一个个独立的插头，只需要将其插到CORBA中间件上就可以与其他组件实现逻辑交互。这种实现方式中，各个组件实现得到了极大的简化，而整个系统的稳定性取决于TAO软件库的稳定性(姚仰光 等, 2008)。

2.3.2 TAO命名服务和消息总线

如上文所述，OCS众多组件安插在CORBA(TAO)中间件上，由于组件较多，组件间建立通信时采用硬编码方式并不合适。因此，采用了TAO提供的命名服务（Naming Service）来解决组件寻址问题(Schmidt, 2005)。

TAO命名服务的作用可以类比于网络通信中的域名系统（Domain Name System，DNS）服务，简化理解如图2.5所示。命名服务提供一个众所周知的通信端口以供其他组件使用。当某个充当服务器角色的组件启动时，通过命名服务端

口将自己以字符串形式(例如“ServerA”)向命名服务进行注册，步骤（1）。命名服务内部维护着一个列表，用来记录服务名及相应的对象引用（Interoperable Object References，IOR）。当某个组件以客户端角色启动时，客户端可以向命名服务提交服务名字符串查询，命名服务搜索内部列表向客户端返回查找到的IOR，步骤（2）。客户端获得IOR后，就可以利用ORB库将IOR转换成服务器实际通信端口，并向服务器提出请求并获得结果，步骤（3）（4）。

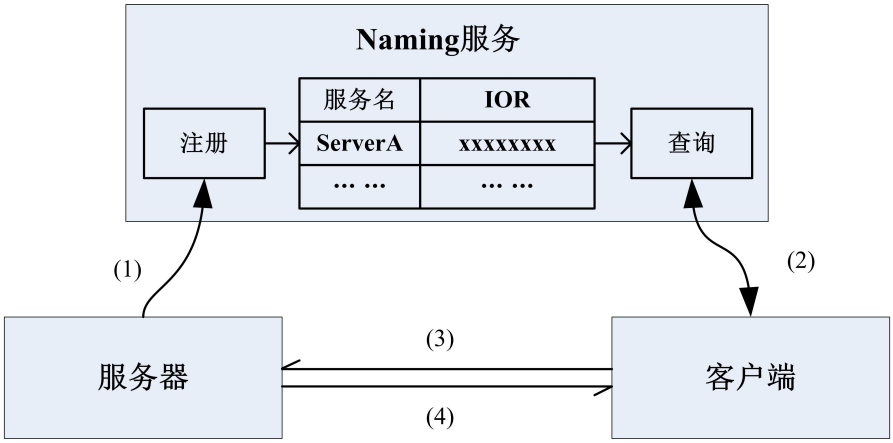

图 2.5 OCS命名服务简图

使用命名服务后，OCS各服务器角色组件可以任意修改内部实现，只需保持服务名不变即可保证整个系统的可运行性，从而为修改、替换和扩展组件带来了极大的便利。

另一方面，OCS利用TAO提供的结构化事件以及通知服务进行组合从而形成了消息总线。OCS消息总线采用的是推模式(王坚 等, 2006)，具体实现简图如图2.6所示。

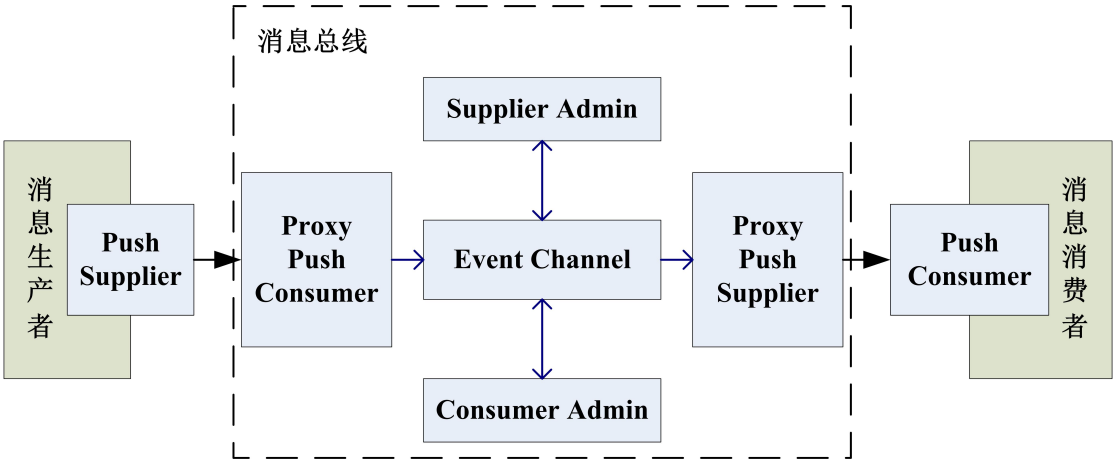

图 2.6 OCS消息总线结构简图

图2.6中，事件通道(Event Channel)是由TAO提供的可执行文件，生产者管理器（Supplier Admin）、消费者管理器（Consumer Admin）、推模式生产者（Push Supplier）、推模式生产者代理（Proxy Push Supplier）、推模式消费者（Push Consumer）、推模式消费者代理（Proxy Push Consumer）均可以有IDL接口文件翻译产生。OCS中需要使用消息总线的组件可以引用生产者或消费者代码来实现自己的逻辑功能。在消息总线上传输消息是CORBA规定的结构化事件。由于采用了TAO通知服务，因此可以进一步实现事件条件过滤以及服务质量控制（Quality of Service，QoS）(黄鲲 等, 2007)。

消息总线提供了多对多通信能力，极大的简化了OCS组件通信难度，例如：当获得一条子系统状态时，状态获得者可以将之转化成符合消息总线要求的结构化事件，并将之推送到消息总线上。而状态获得者本身无需关心事件的后续传递和处理。

2.3.3 通信协议

OCS内部各组件之间以及OCS与其他子系统之间的交互采用了统一的自定义通信协议，由各子系统开发单位共同协商而得。该套协议是在充分考虑了各子系统实际工作原理、子系统功能实现，并保证协议格式统一的基础上制定而成的。

LAMOST通信协议采用XML格式，XML是一种用来描述不同系统或应用程序间消息传递扩展的、推广的标记语言。XML本身来源于标准的通用标记语言（Standard Generalized Markup Language，SGML），XML和HTML相似，只是可以通过使用分层标记对数据进行自我描述，使web页面更接近于程序。XML 允许消息的发送者将内容与表述方式分离，因此，它非常灵活，可用来为各种规范定义新的、专门的标记语言。XML支持多种语言，因为它使用了8位的统一的字符编码（Unicode Transfer Format 8 bits，UTF-8）标准集。XML的这种自描述的、分层次的标记结构使得可利用的数据访问比数据库中可获得的更加丰富。它最主要的用途是对各种数据的统一的格式化表达，系统与相似信息的不同存储格式之间的信息交换。因此，XML可以跨平台。

从功能上讲，LMAOST通信协议分为：命令、状态和命令执行状态三种。其中命令又可以分为单条的基本命令以及由基本命令组合而成的命令流。

图2.7描述了三种消息在OCS系统中所完成的功能。其中，基本命令或观测命令流由观测者发送给OCS用以完成人机交互；而OCS驱动各子系统时发送的

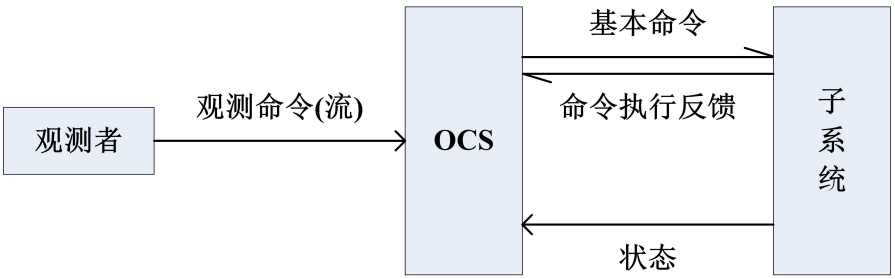

图 2.7 OCS通信协议流

均为单步的基本命令（OCS不会向子系统发送命令流）；子系统在接到单步命令后会发送给OCS命令执行反馈（包括：开始、执行中、完成、出错等）；而子系统的一般状态则根据子系统实际运行情况发送给OCS，由OCS来统一的进行接收和处理。命令状态和执行反馈格式见附录A。

2.3.4 单步命令执行与命令流执行

OCS系统采用命令驱动模式，即整个OCS的运转是由一条一条的命令执行带动的。如前文所述，命令又分为单步的基本命令和由基本命令按串并联方式组合而形成的命令流。

OCS系统命令处理采用同步异步相结合的方式：对于短时完成任务采用同步处理方式，这类命令不会出现阻塞问题。对于需要一定时间才能完成的命令，则采用异步处理方式(朱利平 等, 2007)。

从一般意义上讲，异步通信必然涉及到请求/结果分离、已发出请求的状态缓存、请求完成结果查询三方面的设计(Tanenbaum et al., 2007)。

具体到OCS中，首先，由于采用了附录A所述的命令协议和命令执行反馈协议，因此达到了命令通道和状态的反馈通道分离。其次，命令执行反馈本身恰好可以作为请求状态缓存。最后采用有限状态机的方式可以解决如何标注请求完成的关键步骤。经过巧妙的设计就能完美的实现异步通信机制。

在设计OCS伊始，区分一般状态协议和命令执行反馈协议的主要目的就是为了将命令执行到何种程度这一重要信息从大量的一般状态信息中彻底分离出来，从而为实现命令异步处理打下基础。命令的执行状态有：开始（Start）、正在执行（Active）、完成（Done）、出错（Error）、中断（Interrupted）(姚仰光, 2008)。

图2.8描述了单步命令异步执行时状态机。OCS向子系统发出异步命令后，这条命令的执行状态会按照图中所示方式在五种状态之间进行切换。以正常

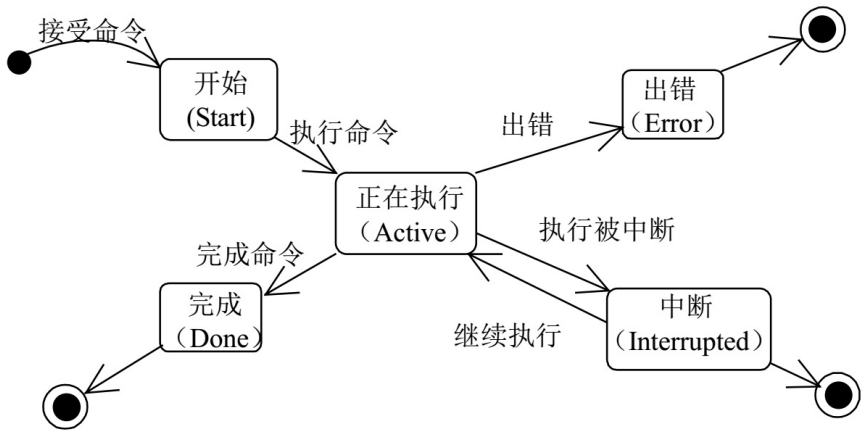

图 2.8 单步命令执行状态机

执行为例：命令发出后会接收到Start执行状态，然后是Active执行状态，最终是Done执行状态。OCS发出命令后即可执行其他业务逻辑，而不必阻塞等待该命令的完成。执行状态的接收和命令完成的判定等工作则由其他线程（甚至是其他组件）来完成，从而实现了命令的异步处理。

在理解了单步命令执行方式的基础上，才能进一步理解由单步命令组合而成的命令流的执行原理。

图2.9为命令流解析器内部结构。命令流解析器接收来自用户界面控制器所发送来的观测命令流（箭头1），按照观测命令流的要求并行或者串行的驱动命令执行器（箭头2）。同时监控命令是否在规定的时间内执行完成（箭头3），并把该命令的执行情况通过消息总线返回给用户界面控制器（箭头4）；同时通过消息总线接收命令执行反馈状态（箭头5）以判定命令实际执行情况；如果该命令没有在规定的时间内完成，命令执行器也会通过消息总线将信息返回给用户界面控制器（箭头4）。

命令流执行控制逻辑模块：负责处理命令的分发，同时启动已发命令计时模块对所分发的命令进行监控。用户界面控制器调用其发送命令流或者给出对超时命令的处理意见；消息总线通过消息接收器提供已经执行完成命令的信息，命令流执行控制逻辑模块据此来决定是否执行下一条命令。

已发命令计时模块：对每条基本观测命令的发送和执行时间进行监控，将命令执行的时间信息通过消息总线提供用户界面控制器。其中命令流执行控制逻辑模块给出时间监控的起始时间，消息总线消费者模块给出结束时间。

消息接收器（总线消费者）模块：消息总线的接收接口，从消息总线获取命令在子系统中的实际执行信息。

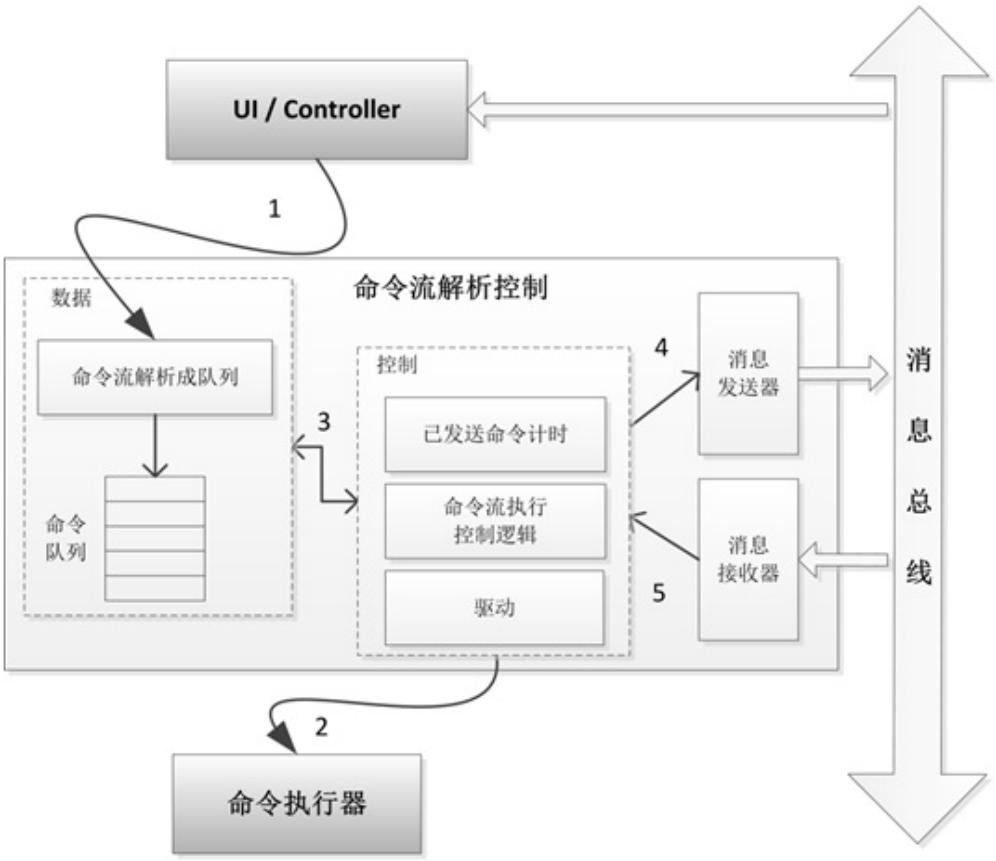

图 2.9 命令流解析器结构简图

消息发送器（总线生产者）模块：消息总线的发送接口，向消息总线推送当前正在执行命令的相关信息。

OCS命令流解析执行算法是整个OCS调度能力的核心保障。如上一节所述，OCS系统将逐条基本命令通过串联或并联节点嵌套形成XML树状结构。执行基本命令的工作由命令执行器组件来完成，从本质上讲该组件仅是通过命令的CUID来确定命令发送对象并经命令发送出去，因此不再累述。而解析和执行这颗XML树，将之按照时序和逻辑拆分成一条条基本命令并传递给命令执行器的工作由命令流解析器来完成。可以说，命令流解析器是OCS调度各子系统运行的核心模块。正是由于命令流解析器的数据结构和算法保证了OCS以命令驱动的方式有条不紊的驱动各子系统，从而完成整个LAMOST观测任务。

如前文所述，命令流中可以包含互相嵌套的串行节点和并行节点，而子节点的串行或并行完全取决于直接包含该节点的父节点的串行或并行执行方式。OCS 命令调度的基本思想是：在执行串行命令块内的命令时，OCS认为只存在一个待执行的命令队列；而在执行并行命令块内的命令时，存在多个待执行的命令队列。串行节点执行相对比较简单，因此不再累述。下面讲述并行节点如

何执行。

XML树的解析和遍历过程中，当前节点所产生的等待完成队列成为父队列，子节点所产生的等待完成队列成为子队列。当执行到当前节点的子节点为并行命令块时，父队列状态设为等待并记录子队列的数目（当前节点可能包含多个子节点）。每当一个字队列执行完成时，父队列计数减一。直到全部子节点执行完成(父队列计数已为零)时，父队列执行才算完成，父队列状态由等待改为完成，且父队列返回到其自身父队列中去(姚仰光 等, 2007)。

命令流本身是XML格式树状结构，而命令流解析器为了解析命令流其内部数据结构采用了队列和链表结构。图2.10描述了比较简单的命令流解析器运行过程中内存数据结构情况。

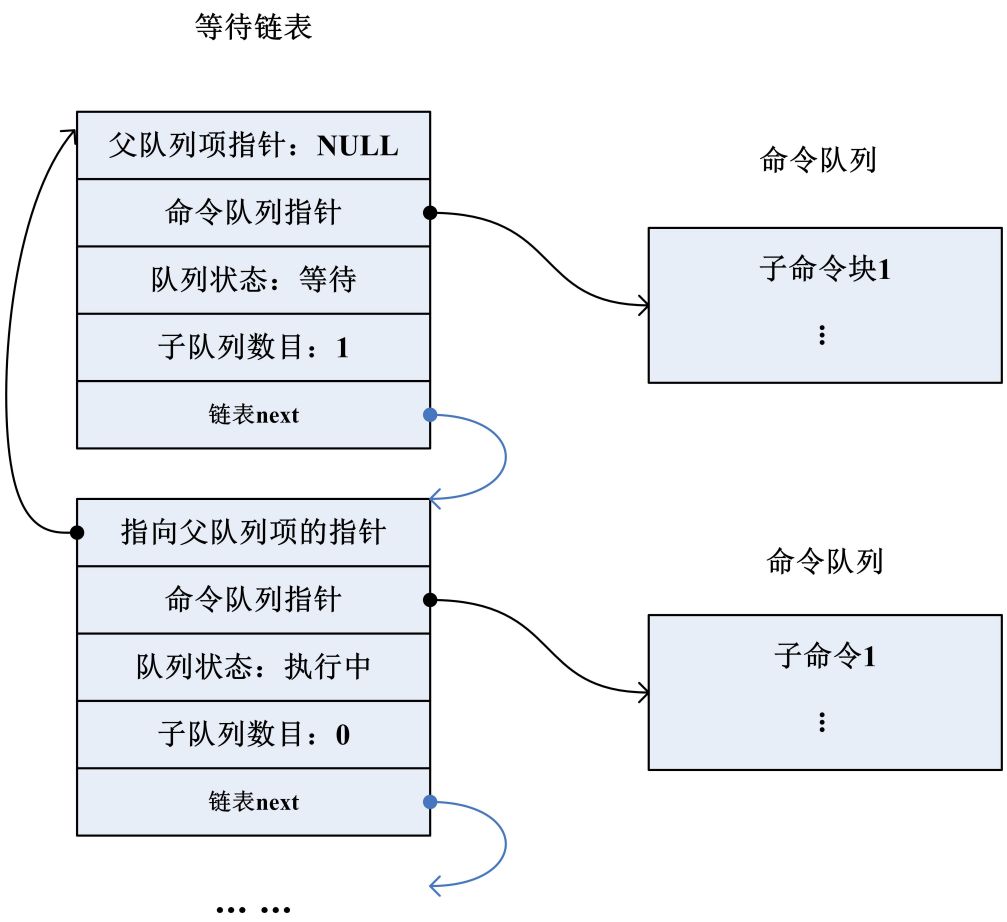

图 2.10 命令流解析器内部数据结构图

命令流解析器接收到需要解析执行的命令流之后，首先生成如图2.10所示的第一个链表项，然后递归遍历整个命令流XML树，逐渐生成各种队列并加入到链表中。之后按照图2.11所示内部流程逻辑从链表第一项开始执行。

当单个链表项中所指向的观测命令队列已为空时，说明该命令队列已经执

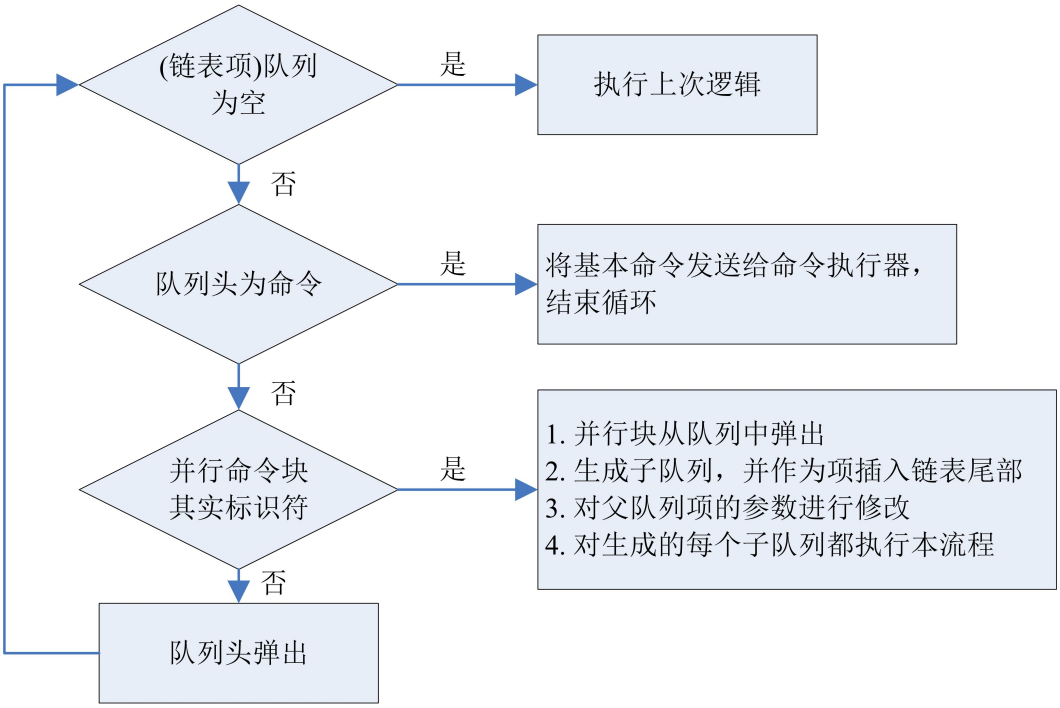

图 2.11 命令流解析器内部逻辑流程图

行完毕。此时命令流解析器会转换到图2.12所示外部流程逻辑开始继续执行。

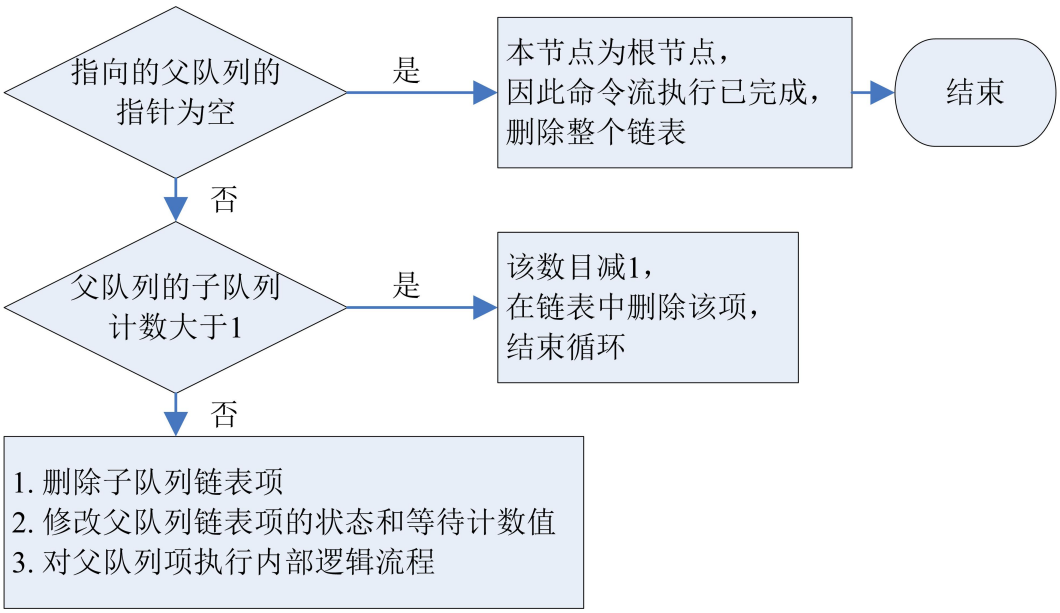

图 2.12 命令流解析器外部逻辑流程图

当有命令执行完成时，命令流解析器会进一步查找该完成命令属于哪个链表项的队列头，因为每条命令都有一个唯一的流水号，所以结果是唯一的。找到该队列后，把队列头的项弹出。若整个链表尚不为空，则命令流解析器会继续按照图2.11所示内部流程逻辑执行下一条链表项。若整个链表已为空，则表

示整套命令流已全部执行完成，命令流解析器进入休眠等待下一套命令流到来时被再次唤醒。

2.3.5 子系统代理机制

在LAMOST望远镜运行过程中，OCS需要与各个子系统进行频繁的信息交互，OCS系统设计中增加了实现子系统通信的代理层接口。在OCS系统内部设计子系统代理主要基于以下三方面考虑：

1）封装子系统差异性，使各子系统在OCS的命令执行器看来完全相同。

2）完成协议桥接，对CORBA信息（内部使用）与基于Socket连接的XML信息（OCS与子系统之间使用）进行转译。

3）使OCS与子系统进一步解耦，如果某个子系统必须修改或升级与OCS之间的通信方式，OCS只需简单调整相关代理组件而无需改动其他组件，从而增加了OCS的适应性以及可扩展性。

现行OCS主要面对的子系统有：观测战略系统（SSS），望远镜控制系统（TCS），焦面仪器控制系统（ICS），数据处理系统（DHS），导星系统（GSS）和气象信息系统（WIS）。因此，相应的子系统代理分别记作：SSSA，TCSA，ICSA，DHSA，GSSA，WISA。

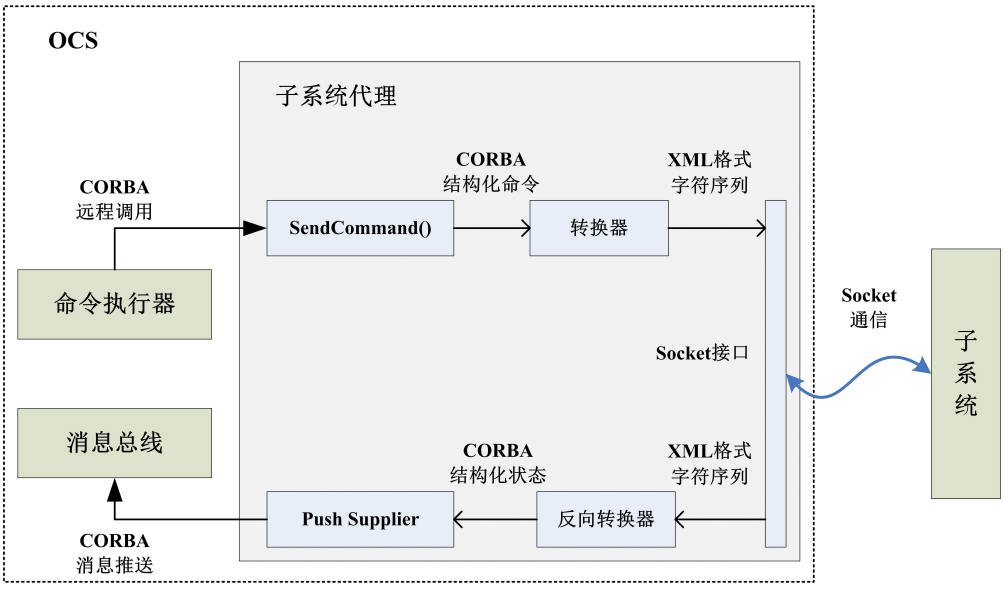

图 2.13 OCS子系统代理组件结构图

图2.13描述了各系统代理组件实现的结构以及与其他组件之间的交互。由于封装了子系统差异，各子系统代理实现的接口基本相同。

子系统代理组件主要提供的接口服务为 SendCommand()，这是由命令执行

器调用以发送基本命令的接口。该接口封装了与子系统的socket 连接与对话过程。对命令执行器来看，无论是哪一个子系统代理提供的接口方法，它只需按照相同的格式去调用，就能保证命令被准确无误地传递给相应子系统。

另一方面，根据命令驱动模型和命令执行反馈协议，子系统会将命令执行情况和状态信息数据返回给OCS。因此，子系统代理组件还需要作为推模式，消息生产者将接收到的信息推送到消息总线上去，以供其他的OCS组件接收和处理子系统状态消息。

2.4 OCS优点和不足

OCS软件是一套较为复杂的软件控制系统。其复杂性根源在于LAMOST的复杂性。既要负责多子系统控制、天文观测调度，还要兼顾状态信息接收、分析和展示。开发过程近十年，经过最初设计、原型实现、测试、最终定型、使用中不断添加和完善功能等诸阶段。它满足了最初的设计需求并稳定观测运行八年（从2010年软件验收算起），为提高观测运行效率做出了贡献。

根据上述章节对于OCS系统整体描述以及对于OCS所采用的关键技术的描述，我们可以粗略的总结出现行OCS系统具有以下几方面的优点：

1）采用异步通信机制（命令状态分离）设计，不但极大的提高了大型望远镜控制中复杂命令执行效率，而且这种机制也是现行OCS能够完成命令流串并联执行的基础。这种思路十分具有前瞻性。

2）前台图形界面组件与后台应用或服务组件的解耦。OCS系统在设计之初就充分考虑了分布式运行环境、组件化软件设计，以CORBA作为中间件的基础上，各个组件独立完成各自的功能。各种后台运行的应用组件和服务组件采用纯C++编写，作为负责人机交互图形界面组件采用了QT编写。这种分离设计更好的发挥了两种语言各自的长处，也减少了组件间直接耦合所带来的实现、修改和升级的困难。

3）命令流设计和解析能够完成复杂的控制流程（串行命令、并行命令、命令嵌套）。前面章节已对命令流解析做过介绍，这里不再赘述。需要强调的一点是，正是由于命令流的引入，才使现行OCS实现对于LAMOST这样包含复杂子系统的大型望远镜的控制、协调、调度成为了可能。

4）消息总线机制的采用。在现行OCS中，消息总线其实主要负责的状态的投递。总线上的消息只包括各子系统状态和命令执行反馈，并不包含命令。由于子系统较多，状态消息和命令执行反馈消息不但门类较多而且数量也很巨大。

另一方面由于OCS系统中组件较多，不同组件对特定消息的需求也不一样。因此，面对如此巨大的消息需求环境，引入消息总线机制极大的简化了OCS系统的设计与开发。此外，在消息总线实现过程中，由于直接采用了TAO提供的通知服务以及结构化事件的概念，因此进一步减少了编程量，并提高了消息总线乃至整个OCS系统的稳定性。

现行OCS系统最初产生设计理念在2000年前后，通过不断的努力，于2008年基本成型。立足于当时整个计算机技术的特点，正是由于其自身包含了以上的多方面的优点，使其能够稳定运行至今，为LAMOST望远镜巡天计划的执行做出了重要的贡献。

随着时代的不断发展，近十年间计算机领域新技术新方法层出不穷，望远镜自动化控制领域也取得了长足的进步。站在当今时代，结合十年来LAMOST望远镜巡天不断提出的新要求，重新审视现行OCS系统，其自身仍然存在一些不足之处：

1）子系统调度约束条件不够充分和强力。现行OCS以单步命令与命令流相结合为控制思路，提供了基本的子系统间的调度功能。但是并未实现子系统操作之间的硬性约束，这方面需要由人工或人工编写正确的命令流来保证。

2）自动化观测控制能力不足。现行OCS系统通过前文所述的基本命令、命令流解析器的方式初步实现了对大型复杂望远镜观测中各子系统的调度和控制。但是这些只体现在具体控制领域，对观测计划排序选择、环境监控等其他领域并未涉及。因此，整个望远镜观测控制并未实现真正意义上的自动化乃至智能化，而自动化和智能化的概念不只在望远镜控制领域在整个计算机技术中也是当今的热点话题。

3）具有错误处理设计，但实现并不完善。现行OCS中，出错处理更多的是实现为警告信息，并通过图形界面提交给观测者，由人工完成错误恢复工作。OCS本身几乎不干预或处理任何具体错误。在LAMOST研制之初，由于并无实物参考且国内对大型复杂望远镜软件设计研发经验不足，这种设计方式无疑是正确的选择。但是，其中也存在不少遗憾，例如，OCS设计中已注意到了各条命令执行时效性，但如果某条命令执行超时后，OCS仅仅是提供一条超时警告，并没有进行下一步处理的能力，不支持超时原因分析以及命令或事件回滚或丢弃等功能。这样的现状是的现行OCS在子系统异常时的自动恢复能力变得很差。随着巡天计划的不断深入，这种所有错误均需人工干预的错误处理方式越来越不适应当前的工程需求。

4）软件陈旧，很难维护和再开发。站在科技蓬勃发展的今天，任何一种开发运行近十年的软件系统显得相对陈旧也是必然的。具体到现行OCS系统中，这种陈旧性更为突出。这其中有部分客观的历史原因存在。例如，作为整个OCS系统骨架的CORBA技术是在1994年OMG提出CORBA2.0后进入了发展的黄金时期，主要应用于拥有成千上万节点的大型工业控制或金融领域。其自身结构庞大、内容繁杂、学习曲线陡峭。随着2002年Internet泡沫破灭，CORBA本身也受到极大的打击，逐渐被各种新型的更简单易用的轻量级中间件和消息总线所代替。目前由于复杂性和过于臃肿，CORBA几乎已经被业界弃用。另外，作为图形界面组件开发的主力语言，现行OCS采用的是当时流行的QT3，而之后的十年间QT先是被Nokia公司收购进行了重要调整推出了QT4系列，之后又一次被Digia公司收购在推出QT5。现在流行的QT5无论是在绘图质量上还是在底层实现上和QT3都存在着很大的不同。

5）开发语言完全采用C++编写，语言选择不够灵活。现行OCS系统后台服务组件完全采用纯C++编写，与CORBA相关的部分代码是经由IDL文件翻译成C++版本后融入程序中，图形界面QT3也是一种基于C++语言的开发框架，因此可以说整个OCS系统完全建立在C++语言基础之上。C++作为一门计算机领域的主要语言固然可以完成各种领域的工作，但是正是由于包罗万象导致了其编写复杂、调试困难。具体到OCS系统中，例如天文计算、文本处理等很多功能采用C++编写其实并不合适，代码很长而且编写很费时费力。近十年来随着Python等一系列易用型脚本语言的飞速发展，许多顶层业务逻辑方面的程序已经可以用这类脚本语言来完成。应该考虑多种语言混编，从而最大限度减低OCS开发效率和维护难度。

6）现行OCS系统设计中只考虑了客户端-服务器（Client-Server，C-S）一种框架。但随着Internet的不断发展尤其是WEB技术的逐渐完善，基于浏览器-服务器(Browser-Server，B-S)框架发展迅猛，B-S框架所带来的平台一致性、实现灵活性、交互友好性等许多优良特性也十分符合当前LAMOST观测控制系统的要求。因此，在保持C-S框架优良基础的前提下，应该考虑加入B-S框架为OCS分层次分等级的开发WEB功能。

以上对OCS系统不足的分析并非是对OCS作用以及现行OCS实现的否定。恰恰相反，正是由于OCS系统对于大型望远镜运行效率的提升起到了重要作用，我们才需要不断的总结和审视OCS软件，充分理解工程需求，深入理解运行机制，不断改进和提高软件，为大型望远镜稳定高效运行提供重要保证。

2.5 望远镜程控技术与RTS2软件

由于大型望远镜自身的复杂性，其观测控制系统的自动化乃至智能化难度非常巨大。当我们将目光转向小型望远镜控制领域时，获得了不少惊喜。由小型望远镜计算机控制技术发展起来的程控天文台技术已经在天文观测领域取得了很大的成就。

2.5.1 望远镜程控技术

望远镜程控技术是一种控制小口径结构相对简单的天文望远镜观测的技术，以计算机技术为核心，也经历了从无到有、从小到大的发展历程。但是由于其所要考虑到控制对象是小型望远镜，其控制复制难道相对较低，使其发展过程中更多的焦点集中于自动化控制以及多点组网领域上。在早期的建设中，其软硬件由大量功能单一的子系统构成，代价昂贵，功能有限。自20世纪80年代起，由于个人计算机的推出，多个程控望远镜计划开始酝酿，部分计划取得了成功，自此人们开始试图标准化程控望远镜的各个系统。进入21世纪之后，随着现代信息技术的突破性进步，大量的望远镜都实现了信息化的改造。近年来，人们逐渐认识到程控天文台技术不但开启了人类在非静态和暂现事件方面对宇宙进行观测的能力，而且由多点组网而成的小口径望远镜网络能极大地提高观测的空域和时域覆盖，从而极大的提高了观测性能和效率(崔辰州 等, 2013)。

2009年在西班牙召开的首届程控自主天文台国际研讨会上，Castro-Tirado 对望远镜程控化的历史做了全面的回顾，与会代表就程控自主天文台的重要历史发展阶段达成了共识，确定了程控技术发展的若干阶段(Castrotirado, 2014)：

1）自动控制望远镜（Automated Scheduled Telescope）：即一架可以执行预先编制好的操作流程而无需观测者实时干涉 (例如不需要观测者手动调整望远镜的指向和跟踪) 的望远镜。这个阶段大约为1968 -1975年间。

2）远程操作望远镜（Remotely Operated Telescope）：即一套按照观测者的观测要求执行远程观测的望远镜系统。这个阶段大约为1975 -1984年间。

3）程控自主天文台（Robotic Autonomous Observatory）：即一架能够执行各种远程观测并且能够在任务执行过程中在没有任何人为协助的情况下自主适应各种变化(天气监测等)的望远镜。这个阶段从1985年持续至今。由于Internet技术的飞速发展，21世纪开始，程控自主天文台技术由原先的专用网点对点控制正在逐渐朝着通用网络全球化整体控制方向发展。

4）程控智能天文台（Robotic Intelligent Observatory）：即一个由人工智能系

统进行决策的程控天文台。这是程控技术未来的发展方向。

从望远镜程控技术发展历程来看，第一代自动控制望远镜和第二代远程操作望远镜的观测模式中，还需要有人操作或有人参与来进行工作。而第三代程控自主天文台技术与我们一直追求的LAMOST观测控制系统自动化有诸多相似之处，因此可以借鉴。

2.5.2 RTS2软件

第三代程控自主天文台发展过程中，远程望远镜系统第二版（Remote Telescope System Version 2，RTS2）无疑是一个里程碑式的软件系统。

RTS2是由远程望远镜系统（Remote Telescope System，RTS）逐渐发展而来。在2000年左右，最初RTS设计的主要目的是为了开发一套可以发现和跟踪观测瞬时爆发的伽玛暴（Gamma Ray Bursts，GRBs）小型望远镜控制系统。在RTS顺利完成开发任务并投入运行之后，天文学家们对该软件系统提出了更高的要求，即：使之成为一套真正的全自动观测控制系统，并且仍需保留能够实现快速GRBs观测模式切换能力。为实现这一目标，软件开发团队对RTS进行了彻底的重构和修改，采用具有面向对象设计能力的C++语言重新实现了该系统，取名RTS2。

RTS2从设计理念到实现方法都有很多过人之处，比如：其插件化设计风格可以保证整套系统具有运行时动态修改和扩展的能力；其严谨的面向对象设计思路为各种硬件设备驱动加入其中提供了极大的便利条件；其自定义的通信协议简单明了不但便于维护而且兼顾了运行时通信效率；其自定义CentralD核心组件起到了命名服务的作用使组件间动态创建交互通道实现起来变得更加简单；其自定义的系统状态变量仅占用4字节（32位）配合基于最基本的TCP通信技术实现了整个系统关键部件状态的全局共享等等。

正是由于RTS2的诸多优点，使之已经问世就获得了业界极大的肯定，并迅速推广。如今，RTS2已经安装并运行于遍布世界各大洲的20多个天文台的30多台望远镜使用，其控制的望远镜口径也从最初的10厘米级发展到近2米级。

2.5.3 RTS2类层次与继承关系

在参考了各种望远镜程控系统设计的优缺点，深入分析各种类型小口径望远镜硬件特点以及自动天文观测流程等一系列问题后，RTS2开发团队抽象出了望远镜观测的一般步骤以及各种设备与服务之间的逻辑关系。在选定C++作为设计和开发语言之后，整个RTS2系统严格遵从面向对象设计要求来实现各种类

型（由抽象类到具体类逐层继承实现）以及类型之间的关系（组合、聚合）。

从实现功能上划分，图2.14比较完整地展现了整个RTS2系统中各种类的继承和组织关系。从图中可以看出整个RTS2系统类继承关系设计的较为复杂并有多层次划分，其中最为关键的几种类型分别为：Rts2Block类系列（进一步发展成为Rts2Client、Rts2Device、Rts2Server等）是系统中各独立组件的主体，通常用于完成某种独立的功能并运行于独立的进程上；Rts2DevClient类系列主要是设备之间以及设备与服务之间的远程代理类，用于保存远程被代理实体的状态等实时信息；Rts2Conn类系列是RTS2根据通信两端的需求以及物理意义对于原始的socket连接的封装类，对于不同的连接（如客户-设备、客户-服务等）其实现对象可以利用C++语言的封装、继承和多态能力进行进一步细化和开发。

从物理意义以及实际运行角度看，RTS2中的各种运行组件基本可以分为以下四种，分别为：中控组件（CentralD）、设备组件、服务组件以及客户组件。它们均为程序实体，相互独立的运行于不同进程甚至不同主机之上，各司其职又相互交互，共同完成整个RTS2软件体系的运行，有条不紊的处理自动控制过程中所遇到的各种问题。各类型组件及相关功能如表2.1所示。

表 2.1 RTS2应用组件类型表

模块类型	描述
中控	RTS2的中央组件，提供三种主要服务：1）系统中所有可用的设备与服务列表；2）系统状态改变；3）各设备的同步。
设备	与各个硬件相关联；针对不同设备提供不同的类，通过类对不同厂商的硬件产品进行封装。
服务	系统的执行逻辑，提供对最终用户的各类功能，实现与其它预报源的数据接收入，实现XML-RPC的访问。
客户	提供给最终用户各类信息，一般工作在交互模式，通过交互终端，或者简单工具实现观测脚本。

RTS2采用层次设计，是非常符合当前系统的构建方法的。特别是RTS2将中央控制、设备、服务分开，利于今后的发展中，可以针对日渐增多的设备提供支持，使得更多的设备可以快速融入RTS2系统中。

同时，中央控制部分可以针对具体的观测模式（自主、自动、远程等）提供扩展。而服务部分的抽象与封装更利于与其它系统的互连，从总体来看，这

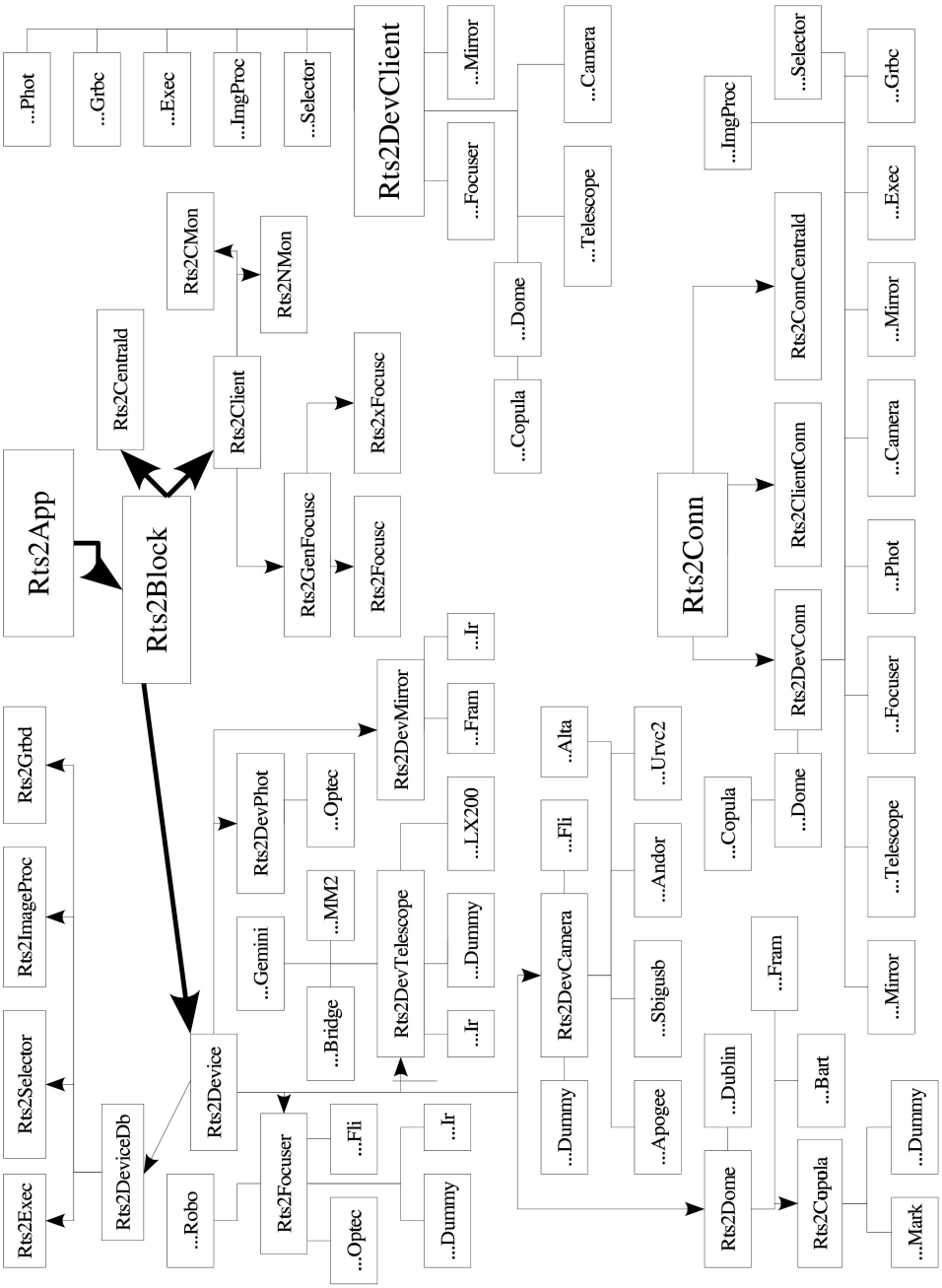

图 2.14 RTS2类继承关系图

样的设计模式思想是先进的，所实现的系统可扩展性较好。

2.5.4 RTS2通信协议与全局状态机

RTS2采用了一系列的自定义协议，这些协议完全基于字符串。该协议设计参考了SMTP协议，客户机发出命令，服务器端响应，这样的模式使得传输快速、简洁并且可靠。其中3个方面较为重要：

1）发送“I am alived”报文作为心跳报文，当没有收到应答时中断连接。

2）可以在文本与二进制间切换，从而实现图像或其它大数据单元的传输。

3）为提高传输率，没有改变的数值不会传输。

分析上述通信过程，可以看出客户端与服务端采用命令应答方式，这样的模式简捷，但无法解决需要长时间运行的异步通信问题。因此，RTS2拓展了状态（State）信息，也就是在协议设计时，增加了一些前缀，如S（状态）、V（值），M（系统信息），这一类的信息发送到客户端后，客户端可以快速分析，获得当前望远镜的实时状态。

为了解决在同一TCP通道上区分处理需要短时快速响应的命令状态信息以及需要长时间传输数据信息，RTS2协议进一步增加了C（二进制数据通道）与D（二进制数据）等信息前缀(Kubánek et al., 2008)。表2.2列举并描述了RTS2常规通信协议前缀及其意义。

表 2.2 RTS2协议前缀说明表

前缀	描述
A	Authorization，验证请求，用于初始化握手和验证。
B	Blocking state，阻塞状态，是一个32位的二进制掩码形式的变量，用于判定命令队列中各命令是否允许执行。
C	Binary data channel，二进制数据通道，用于创建专门传输始二进制数据的传输通道。
D	Binary data，二进制数据，该前缀后面跟随的数据为二进制。
E	Variable description，用于对设备类型中保存的变量进行描述说明。
F	Selection variable elements，用于描述RTS2自定义的枚举型数据结构中的选中项。
M	Message，消息。
P	Priority information，优先级信息。
Q	Priority information request，优先级信息请求。
S	Device status，设备状态。

接下页

接上页

前缀	描述
T	Housekeeping sentence，心跳连接。
V	Update variable value，更新变量值，一旦某个变量的值发生改变是本信息被发送。
X	Set variable value，设置变量值，当连接类试图远程修改对端(设备或服务)存储的变量的值时，使用本信息。
Y	Set variable value and update default value，设置变量值并修改其相应的默认值，当连接类试图远程修改对端(设备或服务)存储的变量的值时，使用本信息。

另一方面，RTS2系统采用状态机的方式解决各个设备的状态变化，并通过设备状态来进行各个设备的动作互斥（互锁），比如：相机曝光中，需要保持望远镜跟踪姿态，不允许随意转动望远镜。

状态和状态阻塞分别由一个4字节（32位）掩码变量保存，状态变量中用各个位的0或1来记录望远镜各部件的当前运行步骤情况，而状态阻塞变量用于命令队列判定其内待执行命令当前是否可以执行。

表 2.3 RTS2部分设备状态机说明表

设备(或服务)	状态	描述
Dome	opening	正在打开中
	opened	已打开
	closing	正在关闭中
	closed	已关闭
Mount	moving	望远镜指向转到
	prking	望远镜指向转向停靠点
	prked	望远镜已转至停靠点
	searching	望远镜进入搜索模式
	correcting	望远镜进入校正模式
Camera	exposing	相机曝光中

接下页

接上页

设备(或服务)	状态	描述
	reading	相机读出中
	idle	相机空闲
Filter	moving	滤光片切换中
	idle	滤光片已到位
ImageProc	running	正在进行图像处理
Executor	moving	驱动望远镜正在指向一个新位置
	acquire	驱动相机正在获取图像
	acquire wait	驱动图像处理
	last read	当前脚本中最后一次曝光正在读出
Centrald	daytime	包含白天、黄昏、夜间清晨等几个时间段，根据本状态可以标识望远镜在当前时段可选任务
	standby	望远镜整体处于就绪状态
	off	望远镜处于关闭状态

表 2.4 RTS2部分阻塞状态说明

阻塞状态	阻塞操作
exposure	望远镜正在曝光不允许指向修改
readout	望远镜正在读出，不允许曝光，但可以进行指向修改或更换滤光片等操作
move	望远镜正在调整指向，不允许曝光

表2.3列举了部分设备（或服务）以及部分其相关常见状态，整个望远镜状态。而表2.4列举了几种常见的阻塞变量及相关说明。

状态仅保存在一个4字节变量中，减少了存储空间的占用，极大的降低了状态变化时广播的信息量。而且由于采用的是掩码形式存储、修改和判断状态，因此代码实现中直接采用二进制与、非、异或操作，提高了系统运行效率。

2.5.5 RTS2设备类扩展

RTS2不仅是一套现成可用的望远镜控制系统，它更是一套易于再次开发和扩展的望远镜自动控制系统软件框架。

RTS2秉承“即插即用”的设计理念，所有设备在接入RTS2时，都可以随时地开启和关闭。模块化设计使得系统的各个部分彼此独立，这为每个设备的单独控制和调试带来了方便。另外RTS2被设计成基于Linux用户空间的模式，不涉及到操作系统的内核部分，有利于二次开发和安装时避免过多代码库的依赖。

从RTS2的代码上看，RTS2具有逻辑清晰的类继承关系，分类也非常合理。RTS2处理提供了种类繁多的各种小型望远镜设备的实际驱动类之外，还在上一层提供了通用型的设备纯虚基类，例如：Rts2Teld、Rts2Camd、Rts2Dome等等。在广泛参考个类型望远镜的软硬件异同的基础上，充分思考望远镜观测运行控制过程的一般规律后，这些设备纯虚基类中已预留出了可以被子类复写的各种有用的步骤或操作的接口虚函数或纯虚函数。从而为基于RTS2框架的二次开发保留了极大的自由度。

使用RTS2作为框架进行二次开发时，在充分理解了RTS2运行机制后，可以根据实际工程需求简单的改写、复写、重写部分或全部预留函数接口，从而实现新设备的接入。

在这方面已有许多研究成果问世，例如：李建 等 (2013)利用RTS2二次开发控制信达NEQ6赤道仪，冉凡辉 等 (2013)利用XML-RPC技术扩展RTS2的远程访问能力，梁波 等 (2015)提出了基于异构操作系统的RTS2相机扩展，Zhang et al. (2015); 张光宇 等 (2015)将EPICS总线技术与RTS2框架相结合设计了南极多色测光太阳望远镜自动控制系统等。这些二次开发经验也为将RTS2框架引入LAMOST观测控制领域提供了宝贵的参考意见。

2.6 大型望远镜设备虚拟化概念与RTS2对OCS的映射

经过深入思考LAMOST观测控制系统需求，认真理解现行OCS设计原理运行机制以及优缺点，并分析RTS2软件框架后，我们可以提出一个大胆的设想：可否将RTS2软件框架引入LAMOST观测控制系统之中，充分发挥RTS自动控制的优势，弥补现行OCS系统的不足？

诚然，RTS2本身无论是从设计初衷还是当前推广使用情况来看都集中与小口径简单望远镜控制领域，并无在如LAMOST这种复杂的大型望远镜上运行的

先例。但是，RTS2本身与LAMOST现行OCS系统相似或相通之处也很多。

1）OCS需要协调的子系统主要有TCS、ICS、GSS、DHS、SSS、WIS等，而RTS2设备（或服务）类主要有Rts2Domed、Rts2Teld、Rts2Camd、Rts2Filter、Rts2Selector、Rts2Senser等，两者之间有着直接对应或间接包含的关系。

2）OCS系统采用子系统代理机制来解耦和简化与子系统之间的信息交互，而RTS2也提供了Rts2DevClient系列类作为socket对端设备在本地的代理机制。

3）RTS2中的中控服务（CentralD）用于完成组件启动接入以及全局状态广播，完全类似与现行OCS系统中的CORBA命名服务和消息总线机制。

4）OCS系统为了实现命令驱动的异步通信模式采用了命令、状态和命令执行反馈三种协议，而RTS2协议也分别包含命令（Command）、消息（Message）和状态（Status）等；OCS系统采用串并行相互嵌套的XML格式命令流来实现子系统驱动以及子系统间调度功能，而RTS2则是将命令压入每个设备（或服务）的待执行命令队列中，根据全局状态变量和阻塞状态变量来判定当前命令是否可以执行，两者思路其实有相似之处，区别在于现行OCS将判定操作全部集中于命令解析器中，而RTS2则将判定操作分散在各种设备中以全局阻塞状态变量来进行约束。

还有诸多相似之处，在此不再一一列举。总之，尝试采用某种映射方式将现行OCS系统组件与RTS2中的各种客户类、设备类、服务类建立直接联系，从而将RTS2框架引入LAMOST观测控制系统领域，利用RTS2自身自动化控制优势提升LAMOST观测运行效率是一个大胆的且极有意义的研究课题。

为了实现RTS2与OCS之间的映射，我们首次提出了大型望远镜设备虚拟化这一重要概念和思路。

大型望远镜设备虚拟化概念主要包括以下几方面：

1）承认并清醒意识到大型望远镜观测控制的复杂度。无论是从软硬件角度讲还是从望远镜观测任务讲，大型望远镜均非以小型望远镜控制为主的望远镜程控技术能够直接应用的。这是客观实际情况。

2）对大型望远镜的操作的抽象。大型望远镜即使再复杂仍然属于天文望远镜控制范畴，均经历了从无到有、从小到大逐渐发展的过程。因此，经过简化和抽象处理，能够获得本质的操作流程。这是实现依据。

3）对大型望远镜繁杂交互信息的分类。大型望远镜所涉及的命令、状态及其繁多，其子系统交互过程也极其复杂，但是这些交互过程和信息中必然存在不同的等级，通过认真分析和思考，可以将之根据子系统内聚性划分成不同等

级，从而进行逻辑分层处理。以现行OCS和子系统为例，仅望远镜主动光学校正时就需要处理夏克哈特曼（Shark Hartmann，SH）波前检测器SHI、SHII、力促动器、位移促动器等多种命令和相应状态，而其实这些状态相机控制子系统并不关心，相机控制子系统只需要知道主动光学已校正好可以开始曝光操作即可。通过分层信息处理、将信息尽量内聚后剩余关键信息才是子系统之间交互所必须的。而这种分析、分类与分层处理正是大型望远镜设备虚拟化需要完成的工作。这是努力的方向。

在提出大型望远镜设备虚拟化概念以及明确了研究课题需要完成的主要工作的基础之上，认真分析当前LAMOST软硬件实际特点、现行OCS系统信息、以及RTS2核心组件的设计后，我们提出了基于RTS2软件框架的LAMOST观测控制系统改造升级的方案，如表2.5所示。

表 2.5 LAMOST子系统(或软件)与RTS2设备(或服务)映射表

RTS2设备(或服务)	LAMOST子系统(或软件)
Dome	圆顶控制、焦面门控制、通风控制、制冷控制
Mount	机架指向跟踪、焦面姿态调整、主动光学、导星校正
Camera	32台CCD相机组成的集群
Filter	光纤单元定位系统
ImgProc	数据在线处理与观测质量反馈
Selector	SSS以及OCS选择观测计划功能
EXEC	OCS整套命令执行和状态处理机制
Sensord-1	环境监控、小圆顶控制
Sensord-2	计算机资源监控
XmlRpcd	OCS尚未提供的基于WEB技术的远程访问机制

在表2.5中，我们基于LAMOST各组成部分的功能逻辑以及RTS2实现机制，对LAMOST各子系统或软件进行分类，然后尝试逐一进行虚拟化映射到RTS2软件框架中去，从而实现RTS2对LAMOST的自动化控制。

经过深入分析并结合工程实际条件，通过这种合理化分类和映射，不但使我们的研究思路更加清晰，也为下一步逐个实现设备虚拟化的过程降低了难度。本文随后的章节将逐一研究Sensord-2、Camera、selector以及Filter的虚拟化，并建立测试系统进行基于RTS2的LAMOST自动化观测控制实验。

第3章　基于RTS2框架的LAMOST计算机资源监控

将RTS2框架引入LAMOST观测控制系统本身是一个意义重大但又十分复杂的课题。如前文所述，为了能使用RTS2的自动控制能力来调度协调LAMOST各子系统我们提出了大型望远镜设备虚拟化的概念。为了能够使本课题研究进行的更加顺利，我们需要首先找到一个切入点。这个切入点既有有助于我们进一步深入理解RTS2内部的运行机理以及各种细节，另一方面这个切入点应该相对功能简单且对整个LAMOST运行影响较小。经过认真分析工程需求，我们选定将LAMOST计算机资源监控映射为RTS2传感器设备（Rts2Sensord）作为整个课题的第一个切入点。Rts2Sensord本身属于RTS2框架中的设备类，完全符合RTS2设备类通用运行规则，通过对其进行扩展可以进一步理清整个RTS2运行以及组件间交互的细节。另外，由于资源监控系统只是对LAMOST各种计算机的硬件资源（CPU、内存、硬盘、网卡等）状态进行监视，并不涉及过多的复杂命令控制，因此不但实现起来也相对容易一些，而且对整个LAMOST正常运行影响也较小。

3.1　LAMOST计算机资源监控系统

如前文所述，LAMOST设计新颖，结构复杂，由观测控制系统、主动光学系统、巡天战略系统、数据处理系统等八个子系统构成。每个子系统又包含大量复杂的软硬件设备。作为现代大型天文设备，每个子系统、每种功能的实现都离不开计算机的控制。LAMOST观测期间，为了保证稳定高效运行，多达上百台的计算机组成了一个庞大的计算机集群。集群中的各计算机节点（尤其是关键节点）的性能、效率和稳定性等因素会直接影响整个望远镜的运行效率。对集群中各节点计算机资源的监视、预警和管理是一项繁重、困难但又必不可少的工作。

LAMOST所使用的计算机不但数量众多，而且计算机之间的软硬件差异也十分明显。在设备类型方面，既有大型刀片服务器、性能优越的工作站，又有适应各种恶劣环境的工控机以及适合工作人员操作的台式机。在分布位置方面，数据服务式计算机（如：数据存储服务器、数据传输设备、在线数据处理工作站等）安置于专业机房中，人机交互式台式机安置于观测控制室，硬件驱动式计算机（如：相机控制集群、机架焦面控制机等）安置于望远镜主体建筑内，

辅助服务式计算机（如：气象站部分室外机、DIMM等）则放置在室外。从操作系统分类来讲，目集群中计算机所采用的操作系统包括了Windows系列、Linux（RedHat、Ubuntu）系列、以及Apple MacOS系列。

面对类型众多、位置复杂、操作系统又存在巨大差异的大型望远镜控制计算机集群，节点计算机的硬件资源信息监视、预警和管理工作完全由人工逐节点来实现的话，不但工作效率低下，而且极其容易出现错判漏判等情况，从而影响大型望远镜的使用率。综上所述，对于现代大型望远镜，需要有一套硬件资源监控软件系统，该系统负责完成对其集群中各种节点（尤其是关键节点）计算机的资源、性能进行采集、存储、跟踪、监控、预警和后期分析处理。

3.1.1 领域调研与分析

以上论述表明了实现大型望远镜计算机集群资源监控系统的意义和必要性。接下来，我们需要充分调研集群资源监控软件领域是否有适合的开源软件以供我们直接使用。

在工程领域，开源计算机集群资源监控软件常见的主要包括Cacti、Nagios、Zabbix等，这些开源软件均在应用领域起到了重要作用(赵文瑞 等, 2014; 郭晓慧 等, 2013; 赵哲 等, 2018)。但是，Cacti的应用更倾向于单一的网络资源监视，不能全面监视节点的各种资源状态。虽然，Nagios和Zabbix均可以全面监视节点资源状态并且提供了报警机制和插件模板等技术。但是因为是通用性软件，两者配置比较繁琐，二次开发难度较大。并且，根据研究课题，我们需要将资源监控软件映射成为RTS2框架的传感器设备组件，从而实现将资源监控软件整合到基于RTS2框架的LAMOST观测控制系统中去，现有资源监控系统软件均无法达到这一要求。因此，基于实际工程需求，我们需要开发一套适合于RTS2框架的LAMOST计算机集群资源监视系统。

由于需要开发一套适用于LAMOST望远镜的计算机集群资源监控系统，在实际设计和开发过程中，还应充分考虑以下几方面的问题，以保证系统的稳定性和可用性：

1）采集信息的内容：由于节点设备不同、作用不同，因此，各节点硬件资源关注的方面也并不完全相同。在软件设计开发中，应采用默认采集内容与配置文件设置相结合的方法，灵活配置各节点采集频度和需要采集信息的内容。例如：CPU使用率、内存使用率、硬盘使用率、网卡上行下行速度、系统进程数、设备温度、风扇转速等。

2）信息采集和传输技术的通用性：由于各节点使用的操作系统差异很大，操作系统提供的系统调用接口差异也很大，因此，资源信息采集和信息传输的方法也会存在很大的差异。在条件允许的情况下，应尽可能保证信息采集和传输方法的通用性和一致性。

3）系统架构：本系统涉及到单机系统调用、网络通信、数据库操作以及人机交互，因此系统架构需要认真选择。目前，主流架构主要分为集中式设计和分布式设计两大类。具体到本项目，虽然设计到的节点众多，但通信和数据处理量很小，因此选择集中式设计，即“客户端-服务器”模式，从而避免了分布式中的数据同步等问题，降低开发和维护难度。

4）系统自身的资源消耗：整个系统采用"服务器-客户端"模式构建，客户端必然部署和运行在每个节点上，因此，必须控制客户端自身的资源消耗量，将整套系统对望远镜计算机集群的负担降至极低。

5）系统开发难度和可扩展性：系统源码应尽量简洁以提高开发速度和运行效率，并且在服务器端应充分考虑可扩展性，为以后软件系统增加功能升级版本预留出稳定、友好的接口。

3.1.2 设计原则与工具选择

为了保证良好的可用性和稳定性，硬件资源监控系统应采用分层方式设计和开发。综合考虑在实际工程环境中的诸多因素，该系统应分为信息采集传输层、信息接收处理层、信息展示应用层。信息采集传输层要充分考虑节点之间的巨大差异，以及传输实现方式的统一性。信息接收处理层具备较好的接收能力和稳定的处理能力，并为上层应用提供解耦的接口。信息展示应用层的功能应该模块化，从而达到人机交互的灵活性和友好性。整套系统还应考虑运行效率，不应给各节点以及整个网络带来较大的负载。

基于以上工程需求分析，结合LAMOST实际工作环境，我们采用了“客户端-服务器-应用”的架构来实现资源监控系统。在编程语言方面，选择了通用性较强跨平台能力出众的Python作为主题语言，并利用了Python的标准库(模块)以及实现各种功能的第三方库。经过充分调研和认真筛选，在本课题主要涉及到以下Python库(模块)的使用：

psutil模块：能够轻松实现获取系统运行时的进程信息和系统利用率（包括CPU、内存、磁盘、网络等）信息，主要应用于系统监控、分析和管理领域。

Python标准库的asyncio模块：支持异步协程。传统服务器端的设计思路大

多采用单进程多路IO复用、多进程（池）、多线程（池）等技术，这些技术对于提升服务器端的响应效率起到了积极的作用，但以上技术也有各自的限制和缺陷。例如：多路IO复用技术在编写大型服务器处理程序时过于复杂不易维护。多进程技术需要占用大量系统资源（占用内存、进程间上下文切换消耗等），并且进程间通信实现过程较复杂。多线程技术相对于多进程技术消耗的系统资源较少，但是线程间同步、竞争、死锁等问题给服务器设计带来很大麻烦。协程是一种开发人员可控的用户态上下文切换技术，相比于进程、线程，协程更加轻量级且调度更加灵活，因此非常适合小数据量、逻辑简单、高并发的网络通信环境。在Python中，协程属于较新引入的概念。Python从3.4开始支持协程，3.5引入协程关键字（async/await）和语法，同时提供了asyncio标准库。

curio库：对asyncio标准库进行了封装，它进一步隐藏了事件循环和复杂回调机制，还提供了Queue（用于协程间通信）、SingalQueue（用于系统信号处理）、run_in_process（用于子进程调度）、run_in_thread（用于子线程调度）等开发工具。从而使编写异步协程程序更加简洁安全。

PyZMQ库：Python版的ZMQ通信库。ZMQ在标准socket库的基础上进一步开发了上层协议，提供了断续重传、负载均衡、发布/订阅等高级接口。适合大型软件组件间通信的解耦，使之更加稳定。

aiomysql模块：Python的异步MySQL数据库接口模块。它允许Python应用程序与MySQL数据库之间进行异步数据交互，同时还提供了可配置的线程池，以提高了数据库操作性能。

3.2 资源监控系统的设计

在对工程需进行深入分析并对所采用的语言、工具进行认真筛选的基础上，结合LAMOST实际工作情况，对基于LAMOST计算机集群资源监控系统（又称之为LAMOST控制节点分布状态采集与监视系统）设计如下。

3.2.1 总体设计

目前，网络通信主流架构主要分为集中式设计和分布式设计两大类。集中式设计需要有强大健壮的中央服务器作为整个通信系统的枢纽。优点是技术成熟，且便于管理和维护。缺点是依赖于中央服务器的稳定性和处理能力。而分布式则将信息或功能分散在不同的计算机之上，节点间相互协作共同完成任务，因此对每台计算机性能要求不高。优点是可以极大的降低服务器成本。缺点是

程序设计更复杂，协作通信量会增加。

具体到本项目，虽然会涉及到LAMOST集群中的计算机数量众多，但通信量和数据处理量均很小，因此选择集中式设计，即“客户端-服务器-应用”模式作为整体架构更为合理，从而避免了分布式中的数据同步等问题，降低开发和维护难度。

LAMOST资源监控软件系统总体分为三层：信息采集器（客户端）、中央信息处理服务器以及上层应用模块，如图3.1所示。

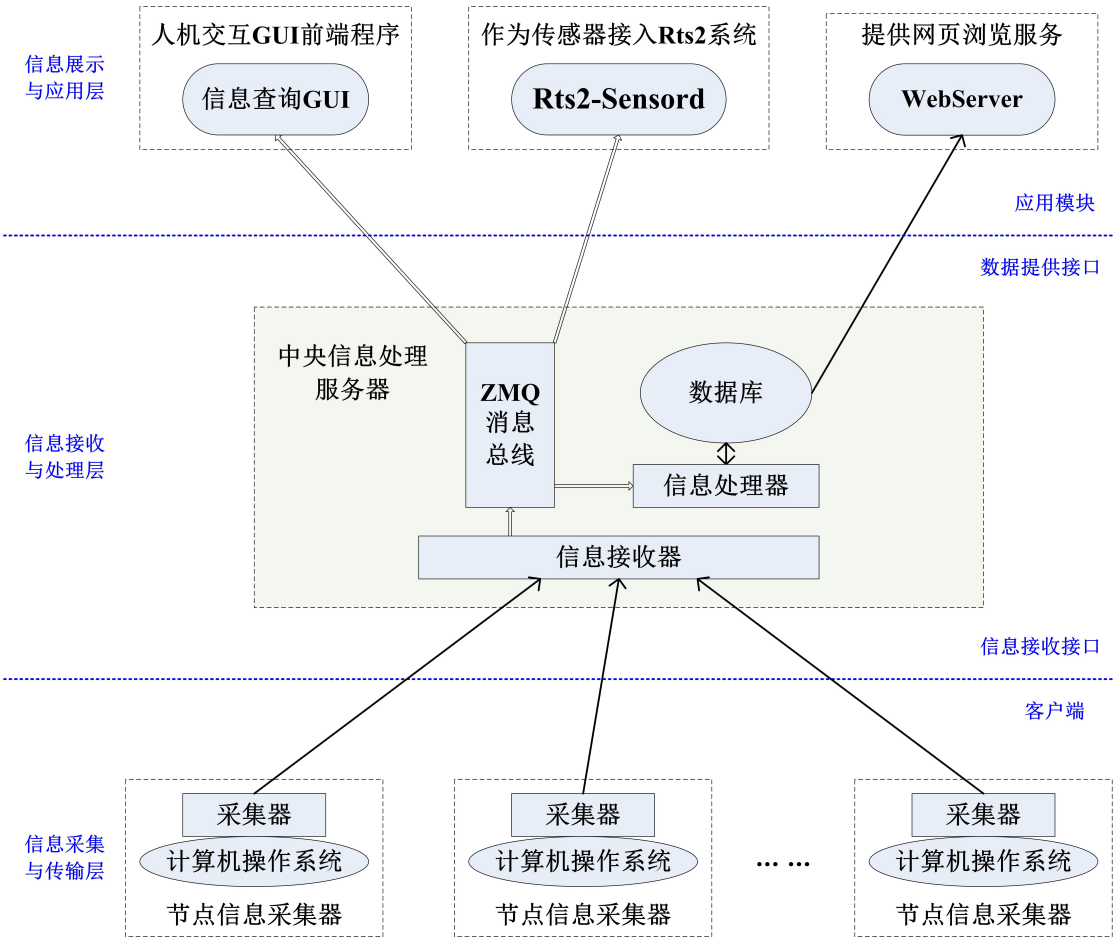

图 3.1 LAMOST资源监控系统总体框架图

部署在LAMOST计算机集群各节点计算机上的信息采集器根据默认设置和配置文件来确定信息采集间隔时间以及本节点需要采集的具体信息的内容后，周期性的采集本节点硬件资源信息并发送给中央信息处理服务器。

中央信息处理服务器接收众多客户端发来的信息，进行信息处理（如：数据验证、数据分析、数据存储等），并通过自身的服务接口向上层应用模块提供数据服务。

上层应用模块可以有多个，分别实现不同的功能（如：图形界面人机交互、

网页显示、接入观测控制系统、数据详细分析统计等）。它们均通过中央信息服务器的服务接口获取数据，然后根据所获得的数据完成自己的工作。

应用模块和中央信息处理服务器相互独立，不强制规定启动先后顺序，也不需要指定在同一台电脑上运行，最大程度实现系统解耦，从而提高系统稳定性和可用性。

3.2.2 信息采集器设计

信息采集器主要责任是周期性采集节点系统资源信息、将信息发送给服务器。考虑代码的简洁性和易维护性，选取Python的psutil包来实现信息采集。绝大多数节点的操作系统中都已自带了Python解释器以及相应包。对于未安装psutil包的节点采用pip安装或源码安装也十分简易。个别节点无法安装Python，则采用C语言调用原生系统库接口获取信息。图3.2描述了采用psutil包获取系统资源的部分代码。

```
import psutil as ps
… …
cpu = ps.cpu_percent(interval=60)            # 获取CPU使用率，周期60秒
mem = ps.virtual_memory()                    # 获取内存使用率
disk = ps.disk_usage(‘/’)                    # 获取挂载点‘/’所对应磁盘分区使用情况
nic_in = ps.net_io_counters().bytes_recv     # 获取网卡接收字节数
nic_out = ps.net_io_counters().bytes_sent    # 获取网卡发送字节数
procs = ps.pids()                            # 获取进程列表
… …
```

图 3.2 信息采集器psutil用例代码片段

在采集到信息之后，信息采集器需要通过网络将信息发送给服务器。网络信息传输的实现方式有多种选择，例如：UDP传输、TCP传输、ACE通信库、ZMQ通信库等。采用UDP通信可靠性相对较差，因此暂不考虑。而ACE通信库是一套完善C++通信库，它完全采用面向对象设计，支持和采用的多种设计模式。但是由于过于庞大复杂且开发维护难度很高，因此并不适合本文提到的实际工程需求。ZMQ通信方式（见3.2.3节）虽然使用方便灵活，但是需要在各个客户端上安装ZMQ库。如前文所述，各个节点硬件设备差异较大、操作系统种类繁多，如果根据各节点软硬件配置选择相应版本的ZMQ库并进行安装，不但工作量巨大、可操作性很差，而且不利于今后设备的更新和扩展。TCP传输作为最常见的底层通信协议各种操作系统均自带了其实现库，无需再自行安装配置。因此，信息采集器与服务器之间的通信采用了TCP通信协议。

基于以上分析，我们选择TCP通信协议作为信息采集器信息发送的通信协

议，且具体通信功能的实现采用Python语言。为保证通信稳定性设计了较为简单的保障逻辑：设置错误容忍阈值，将采集信息和发送信息功能包裹在循环体中。出现通信故障时捕获异常错误计数器自增1，若错误计数器未到阈值则休眠指定时间后重新循环，若已到达阈值则结束程序释放系统资源。

3.2.3 服务器设计

中央信息处理服务器是整套系统的枢纽，主要任务有两个：接收大量信息采集器（客户端）周期性发来的资源信息，对信息进行处理后存入MySQL数据库。因此，对其运行效率和稳定性要求很高。为了使设计思路清晰、实现代码简洁，在实际方案中采用了两个子进程，分别为：异步消息接收器子进程和MySQL异步处理器子进程。中央信息处理服务器内部结构如图3.3所示。

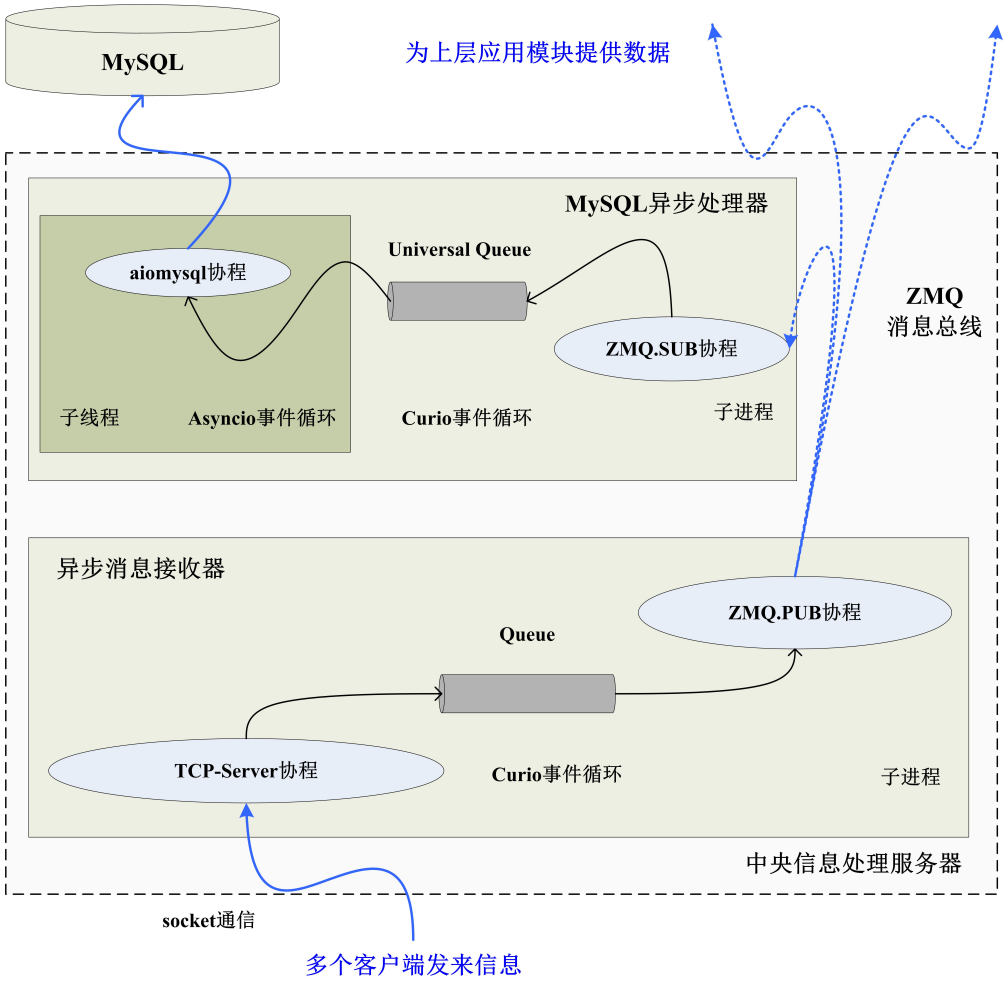

图 3.3 中央信息处理服务器内部结构图

异步消息接收器子进程内部分别包含了TCP-Server协程和ZMQ.PUB协程。TCP-Server协程负责处理节点采集器的TCP连接申请以及并发的接收多个信息采集器发来的消息。ZMQ.PUB协程负责将接收到的消息发布到消息总线上，以

供上层应用模块使用。两者之间的采用curio库的Queue来实现单线程内不同协程之间的异步通信，整个子进程由curio库事件循环驱动。

MySQL异步处理器子进程内部分别包含了ZMQ.SUB协程和aiomysql协程。ZMQ.SUB协程负责向消息总线订阅感兴趣的消息类型并获取消息。aiomysql协程负责将获取到的消息异步的存储到MySQL数据库中。由于aiomysql模块必须采用标准的Python.asyncio事件循环来驱动（curio库目前尚未提供aiomysql模块接口），而Python.asyncio事件循环本身必须独占一个线程，因此程序在实现过程中需要为aiomysql协程创立独立的子线程。主线程和子线程之间采用curio库的Universal Queue实现异步通信。与异步消息接收器中的Queue不同，curio库提供的Universal Queue可以解决分属于多个线程的不同协程之间的异步通信问题。整个子进程也是由curio库事件循环驱动的。

此外，中央信息处理服务器采用了ZMQ的发布/订阅机制组建消息总线，实现对上层应用模块提供数据的功能。ZMQ.PUB端口用于发布消息，可以有任意数量的ZMQ.SUB端口用于订阅和接收消息。ZMQ.PUB和ZMQ.SUB无需在同一计算机上部署，且运行互不干扰。由于ZMQ以上优点，极大的简化了服务接口层的开发难度，实现了中央信息处理服务器与各应用模块之间的完全解耦。

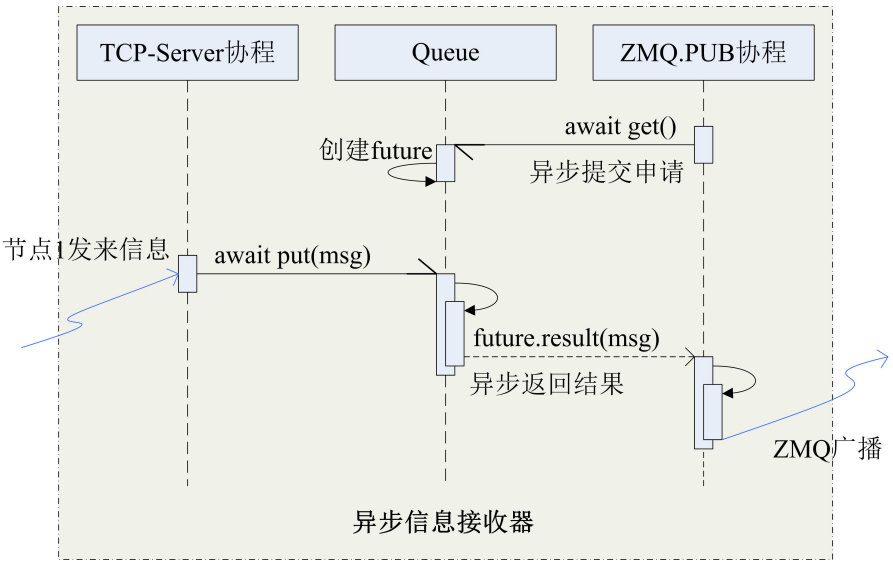

图 3.4 异步消息接收器运行时序图

中央信息处理服务器异步信息接收器运行时序如图3.4所示，MySQL异步处理器运行时序如图3.5所示。由于组件均为协程，组件间均采用异步方式通信，协程逻辑执行与上下文切换由用户精确控制，因此在不占用更多操作系统资源的前提下，极大的提高了服务器的并行处理能力和数据吞吐量。实际运行中，

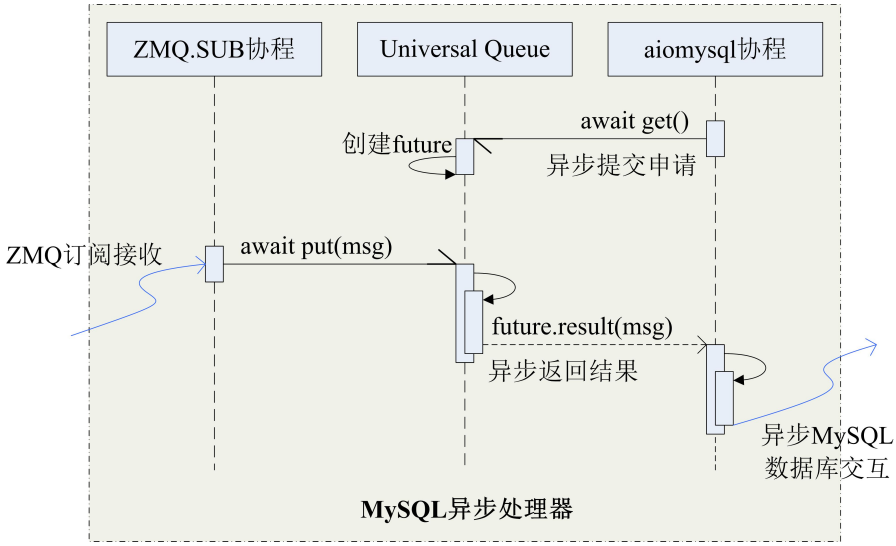

图 3.5 MySQL异步处理器运行时序图

中央信息处理服务器采用Linux后台守护进程方式运行，并设置为开机自启动，从而保证了程序运行更加稳定和高效。

3.3 资源监控系统运行

经过精心设计，采用Python语言并将大量开源软件包引入到开发过程后，LAMOST计算机资源监控系统的客户端以及后台服务器便可以快速开发、测试和部署到实际工程环境中了。

如前文所述，LAMOST计算机资源监控系统采用的是“客户端-服务器-应用”模式设计。而应用模块主要负责数据的应用于展示。资源监控系统允许运行多个相互独立的应用模块来完成不同的功能。由于ZMQ消息总线的应用，使中央信息处理服务器与应用模块之间彻底解耦，因此，极大的降低了各种应用模块的开发难度。

根据工程需求，LAMOST资源监控系统目前提供了三种应用模块，分别为：基于PyQT5的图形界面（GUI）模块、基于Django的网站服务模块以及基于RTS2框架的传感器设备（Rts2sensord）模块。其中，由于基于RTS2系统的传感器设备涉及到RTS2框架通信机制以及设备扩展方法，留在3.4节描述。

3.3.1 基于PyQT5的本机图形界面运行

由于资源监控系统服务器层采用了ZMQ消息总线，使服务器层和上层应用完全解耦。上层应用模块可以通过ZMQ.SUB接口订阅和获取总线上的消息。另一方面，由于选择了PyQT5作为本地图形界面开发工具，可以快速的完成GUI相

关代码的编写，从而极大的提高开发速度。

但是，这里有两个问题需要解决：

1）PyQT5包本质上是Python语言对QT5库的一种封装。而QT5核心库在实现GUI时必须占用主线程且始终维持着一个内部的事件循环(陆文周, 2015)，这样就无法采用上一节中央信息处理服务器的方式利用Curio异步库来实现GUI应用模块了。另一方面，ZMQ.SUB接口作为消息接收者，由于与消息发送者彻底解耦，导致其接收信息函数必须也是一个包含在无限循环中的阻塞调用。两者都需要以无限循环的形式占用程序逻辑流程线，且两者之间无法直接相互嵌套，因此GUI模块需要采用双进程或双线程方式来实现。实际开发中，我们采用了双线程方式。

2）线程间通信问题。采用双线程方式实现时，需要考虑两者之间信息交互的方法。最常规的方式是双线程采用全局变量和线程同步技术(线程锁、条件变量等)来实现线程间交互。但是这种方式实现复杂(为了防止死锁需要精心设计)、效率较低。QT本身提供了更为优雅的解决线程间通信的方式，即QT信号槽。因此，实际开发中我们采用了多线程QT信号槽方式来完成数据交互。

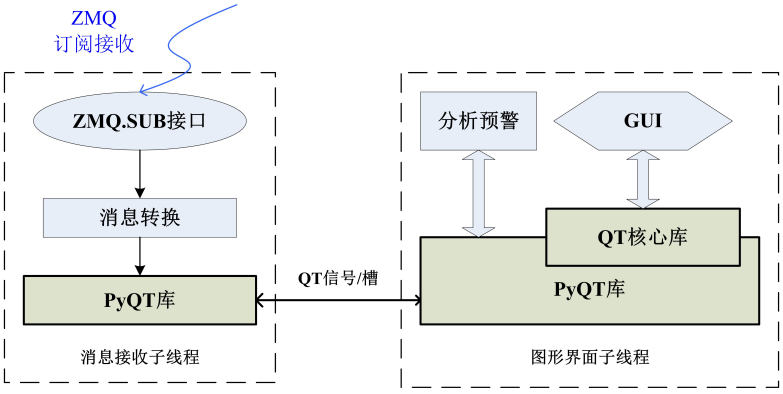

图 3.6 GUI模块实现图

图3.6描述了基于PyQT5的图形界面模块实现原理图。(Qt-GUI相关业务放置于主线程，ZMQ消息接收转换业务放置与从线程。两者之间采用QT信号槽进行通信。在消息接收线程中，ZMQ.SUB接口接收从ZMQ总线发来的各节点硬件资源信息，经过格式转换后以QT信号形式发射出去。在图形界面线程中，实现QT槽函数，用槽函数接收发来的信号并进行一定的分析，最终显示在由QT编写的本地图形界面上。

在GUI图形界面实现过程中，为了达到更好的显示效果，采用了QT样式表(QT Style Sheets，QSS)，QSS类似于HTML的层叠式样式表(Cascading Style

Sheets，CSS)，通过设置独立的QSS文件将显示效果与业务逻辑代码分离，从而降低图形界面开发难度。

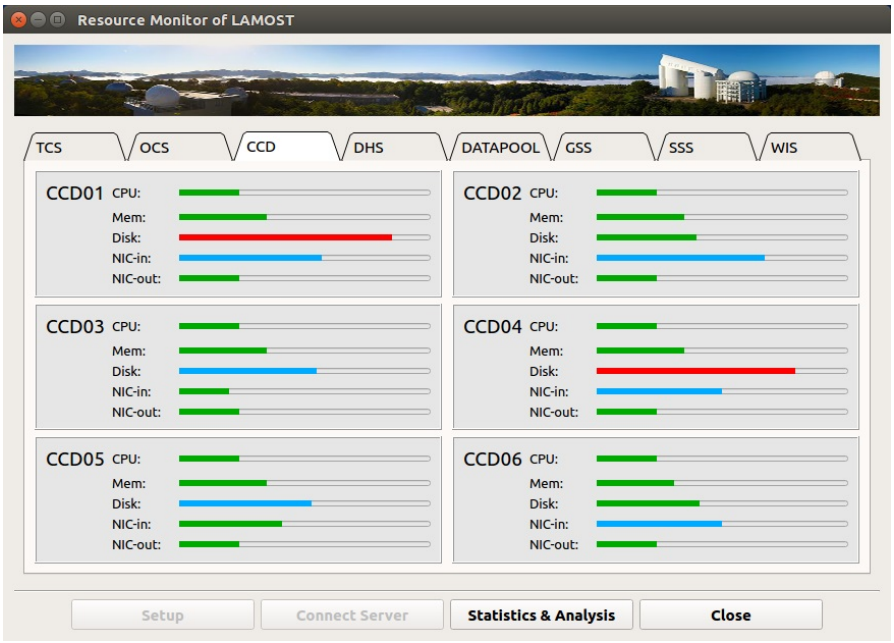

图 3.7 资源监控系统GUI运行时截图

图3.7为本机运行显示界面(GUI)模块运行截图。GUI实时显示各节点计算机资源信息并以颜色进度条的方式为运维工程师提供资源占用信息。图3.7中，01号和02号CCD计算机的硬盘占用率超过80%，因此，GUI相应节点子界面的硬盘进度条会变成红色。以提醒观测者注意该情况，从而达到预警效果。

3.3.2 基于WEB技术的网页浏览

如前文所述，由于资源监控系统中央信息处理服务器的MySQL异步处理器已经将接收到的各种资源信息推入数据库中并保存起来，因此，本工程中实现WEB网站就变得十分容易。

随着互联网的飞速发展，WEB技术也取得了长足的进步。WEB技术不但可以提供跨平台、跨网域的信息访问，而且其自身性能瓶颈也得到了极大的解决。由于技术和市场的双重驱动，基于WEB网站访问的B-S架构也不断推陈出新。我们选择了基于Python语言实现的Django作为服务器框架。该框架不但简单易用、资料众多，而且开发维护效率也不高。由于目前资源监控系统中的WEB应用模块只是安装Django基本配置来实现的，因此，此处不再累述。

目前，网站应用模块架设于LAMOST内网中，网站应用模块提供网页浏览查询服务。允许观测者和运维工程师在LAMOST内网中的计算机上通过浏览器

查询各节点资源信息数据。

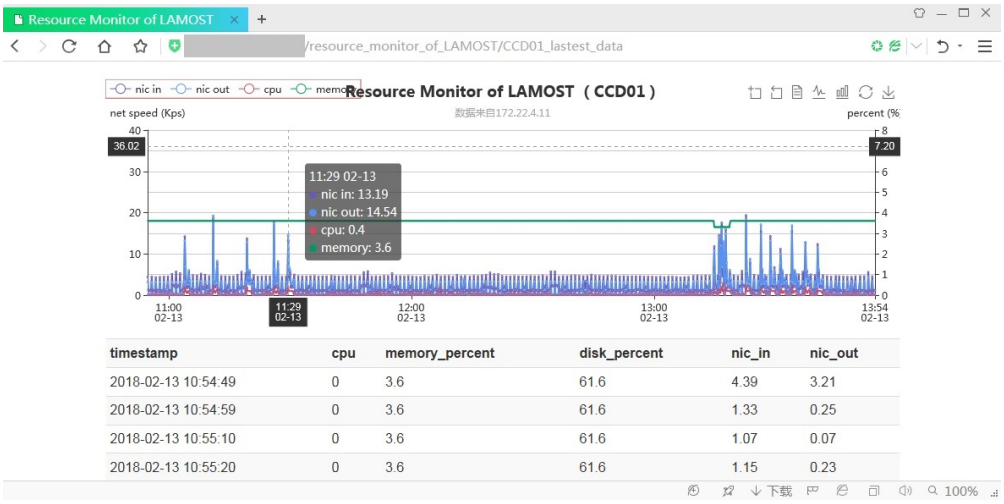

图 3.8 资源监控系统网页截图

图3.8所示为01号CCD计算机的资源信息页面。页面上半部分以曲线图方式显示了该计算机各种资源信息的历史记录和趋势，并采用了可设置虚线的形式提供预警功能。网页下半部分为该计算机的资源信息列表，表中分别列出了信息采集时间(timestamp)、CPU使用率、内存占用率(memory_percent)、硬盘使用率(disk_percent)、网络下行速度(nic_in)、网络上行速度(nic_out)等信息。

3.3.3 系统运行时资源占用情况

在设计资源信息监控系统之初，其中一个工程需求就是系统本身不能占用过多硬件资源，否则的话，整套系统部署和运行就变得得不偿失。因此，在软件设计和开发过程中，在保证可用性的前提下尽量采用简化和优化的代码。

将资源监控系统部署在实际工程环境中，实际运行测试结果如下：

```
lamost@localhost:~
[lamost@localhost ~]$ ps -e -o'pid,cmd,pcpu,rsz' | grep rsmolc.py | grep -v grep
24622 python2 ./rsmolc.py start    0.0  3768
```

图 3.9 客户端资源消耗

从图3.9中可以看出，节点消息采集器（客户端）资源消耗：CPU占用小于0.2%，内存占用小于4KB。

```
lamost@localhost:~
[lamost@localhost ~]$ ps -e -o'pid,cmd,pcpu,rsz' | grep resource_mon | grep -v grep
12354 python3 ./resource_monitor_  0.0  9648
12355 python3 ./resource_monitor_  0.0 10624
12356 python3 ./resource_monitor_  0.0 11308
```

图 3.10 服务器端资源消耗

从图3.10中可以看出，中央信息处理服务器资源消耗：CPU不超过0.5%，内存约占用40KB。

其他应用层模块，由于并不需要始终运行，且基本都是运行在服务器端或其他终端设备上，因此并不违背设计需求，此处不再赘述。总之，从实测结果可以看出，本系统不会对节点计算机和中央信息服务器产生过大的资源消耗。

而网络传输方面，每个节点产生的信息均以字符串形式传输，单条信息不超过128个字节，信息采集周期默认值为10秒。因此，100个节点采集器每秒产生的信息大小不超过1280字节，即整个资源监控系统对LAMOST内网带宽占用不超过1.25KBps。未来，如果由于节点信息量增多或系统总节点数增加导致网络带宽占用过多，可以考虑放弃字符串采用二进制数据传输。

从以上分析可知，LAMOST资源监控系统无论是在自身资源消耗方面还是网络负载方面，都符合当前工程需求。

3.3.4 系统预警和分析处理功能的扩展

LAMOST资源监控系统已初步完成，部署于实际工程环境中能够准确有效的采集和存储集群中各节点计算机资源信息数据。但仍有以下两方面不足：

在人机交互预警方面，如3.3.1节和3.3.2节所述，目前通过本机GUI资源条颜色（绿、蓝、红）变化，以及网页预警线的方式为工作人员提供最基本提醒功能。未来可以借鉴Zabbix系统实现思路，根据预警信息类型或错误等级为工作人员提供邮件推送、微信消息等功能，从而进一步提高系统的友好性。

另一方面，已存储信息的后期分析、统计和处理功能目尚未完成。主要原因在于该功能需要不断积累大型望远镜运维经验。

不过，由于服务器内部采用了基于发布/订阅机制的ZMQ消息总线技术，使预警组件和数据分析处理组件与整个软件框架完全解耦，极大降低了进一步开发的技术难度，因此，LAMOST资源监控系统会不断完善，最终为望远镜高效运行提供有力保障。

3.4 LAMOST资源监控系统接入RTS2框架

在完成了资源监控系统客户端、中央信息处理服务器、本地GUI应用模块和WEB应用模块之后，如何实现基于RTS2Sensord传感器模块成为重中之重。传感器模块的实现是将RTS2框架引入LAMOST观测控制的第一步，它为以后其他更加复杂和重要的子系统引入开辟了道路。

为了实现基于RTS2传感器模块，我们还需要深入分析RTS2设备类内部运行机制以及不同实体之间的通信原理。

3.4.1 Rts2Daemon类与RTS2整体通信机制

RTS2框架是一套严格且完整的面向对象程序，以我们所要实现的传感器为例，RTS2本身已经提供了纯虚的Sensord类。Sensord类中给出了作为传感器设备向整个RTS2系统发送信息的接口函数（详情见3.4.2节）。但是作为一个运行在进程中的实体，其运行逻辑并不是在Sensord类中定义的。而是来自于其父类或祖先类。由于RTS2各个实体的运行逻辑相同，因此将之抽象到祖先类中，而无需在各个实体中逐一实现，这也正是面向对象的威力所在。

在RTS2框架中，能够长时间运行的模块类均继承自rts2core::Daemon，该层抽象出了几乎所有与通信、存储和运行逻辑相关的数据成员。Daemon类中与通信和运行逻辑的关键数据成员如图3.11所示。

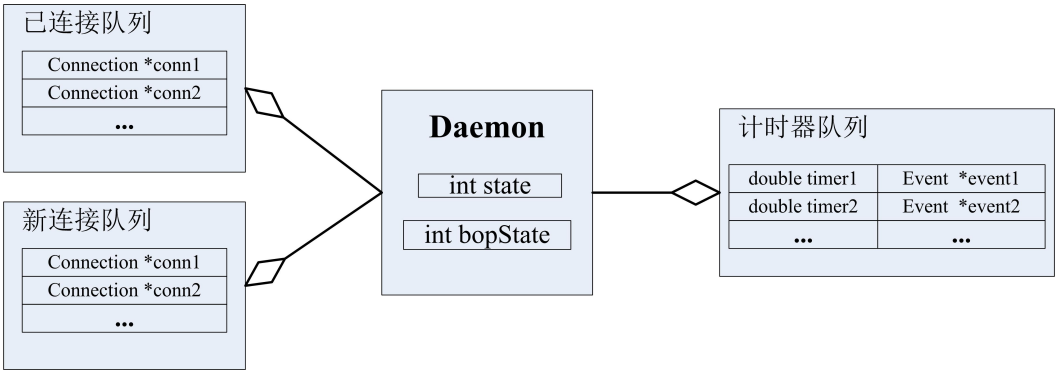

图 3.11 Daemon类的核心数据成员

1）已连接队列，与其他组件的通信连接，负责管理和维护底层socket。

2）新建连接队列，刚接入本组件的socket连接类的容器，这些刚接入的连接类的实例对象可能需要作进一步的状态同步或注册等处理。

3）计时器队列，提供定时服务，当计时器到时时，采用统一的postEvent()函数来处理相关事件。

图3.12描述了Daemon运行的基本流程，其中最关键的预留接口函数为：

1）initDaemon()，在开始运行前，对其自身(继承自它的子类亦然)进行初始化工作，包括读取命令行参数、修改各种数据成员初值等。

2）oneRunLoop()，程序运行过程中无限循环调用本函数，来一次次完成常规业务逻辑操作。

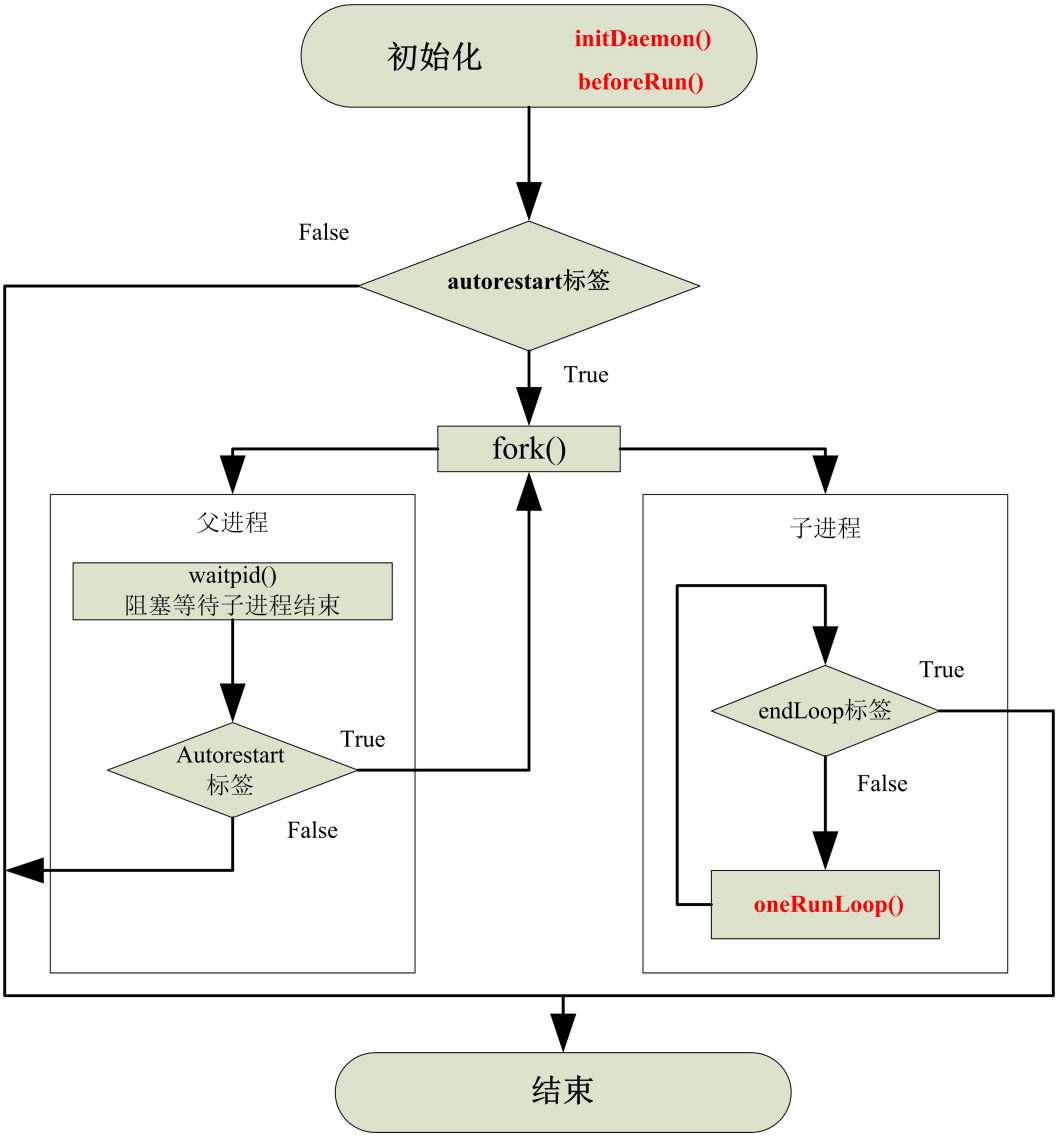

图 3.12 Daemon类的运行逻辑流程图

oneRunLoop()是每个组件实际功能的具体执行者，图3.13描述了该函数内部执行的操作。

1）计算本次多路IO监听的超时时长。

2）如有需要添加新的socket到需要监听的文件描述符集合中。

3）调用select()函数，进程阻塞等待。直到文件描述符集有变化或超时。

4）如果某个socket上有新信息到来，则调用selectSuccess()进行相应处理。

5）调用idle()，进行更新处理，为下一次再调用oneRunLoop()做好准备。

由面向对象继承关系可知，所有继承自Daemon类的实际运行类(设备类、服务类等)都会按照这一业务逻辑一轮一轮的反复执行。各中子类之间的区别主要在于如何重新实现用于对创建不同类型连接做自定义处理的addSelectSock()函

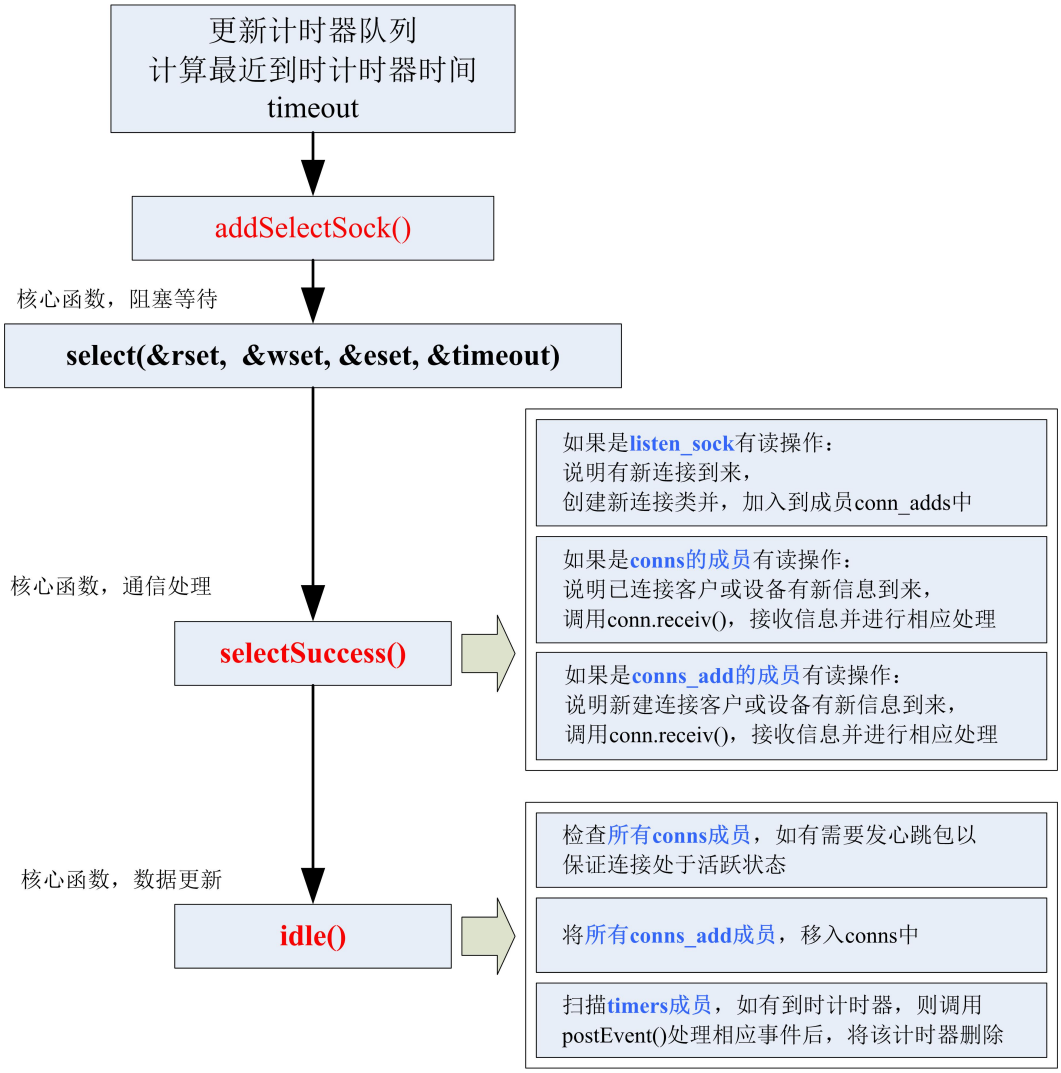

图 3.13 Daemon类核心函数的业务逻辑图

数、当有新连接到来或已有连接上有新信息到来时做处理的selectSuccess()函数、以及更新操作idle()函数。

在深入理解了RTS2的Daemon类及子类运行逻辑的基础上，结合源代码可以分析出各中设备类如何创建socket连接并融入整个RTS2的通信框架之中。

图3.14描述了设备类在初始化过程中(initDaemon函数)是如何连接到RTS2的中控服务组件(Centrald)上的。

Centrald是RTS2的核心组件，其功能是存储和广播当前系统中各组件的状态(Kubánek, 2010)。彻底理清Centrald与设备(命令执行者)之间的交互，对于使用RTS2，以及改造RTS2为LAMOST所用至关重要。

Centrald运行中提供了监听套接字listen_sock。设备启动时，通过connect()连接listen_sock。Centrald在selectSuccess()中判断listen_sock上是否有可读信息。如

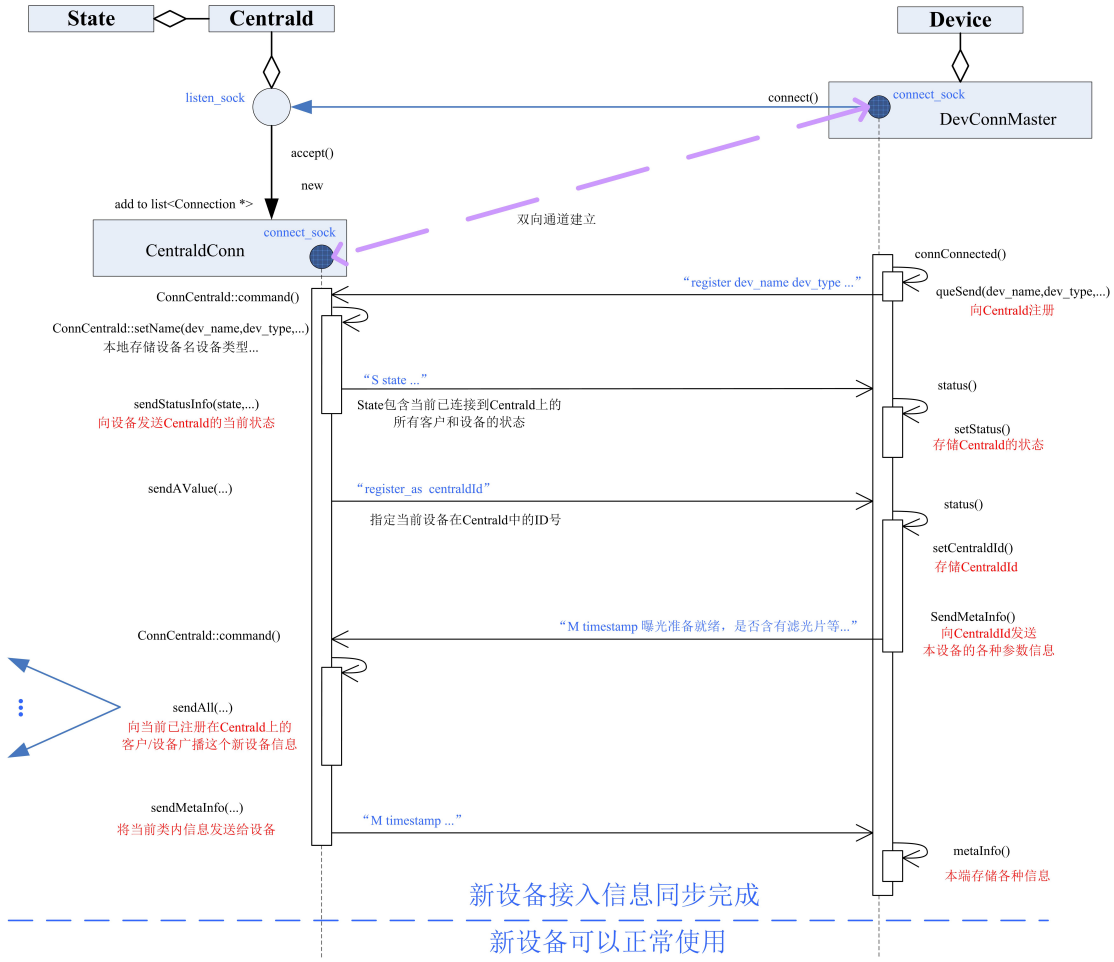

图 3.14 设备类接入RTS2框架的时序图

果有，说明是新设备接入，则在堆上创建封装已连接socket的对象ConnCentrald。其后所有与该设备相关的网络交互都由ConnCentrald来完成。

新用户接入后，通过几次快速交互，即可达到状态信息同步。其后，设备即可正常运行。

3.4.2 资源监控系统接入RTS2框架

在掌握了RTS2设备类继承关系以及运行机制之后，我们可以利用GUI应用模块相似的设计思路来实现LAMOST资源监控系统的RTS2传感器应用模块，称之为LAMOST-Sensord模块。和本地GUI应用模块的区别主要在于：

1）RTS2采用C++语言，因此LAMOST-Sensord类无法用Python语言编写。

2）RTS2框架没有集成QT相关库，因此线程间通信不能采用QT信号槽机制而只能选择POSIX线程同步技术。

图3.15描述了LAMOST-Sensord模块内部结构图。其中消息接收子线程中除了类似于GUI应用模块中的ZMQ.SUB接口(此处是C++版的ZMQ)、格式转换单

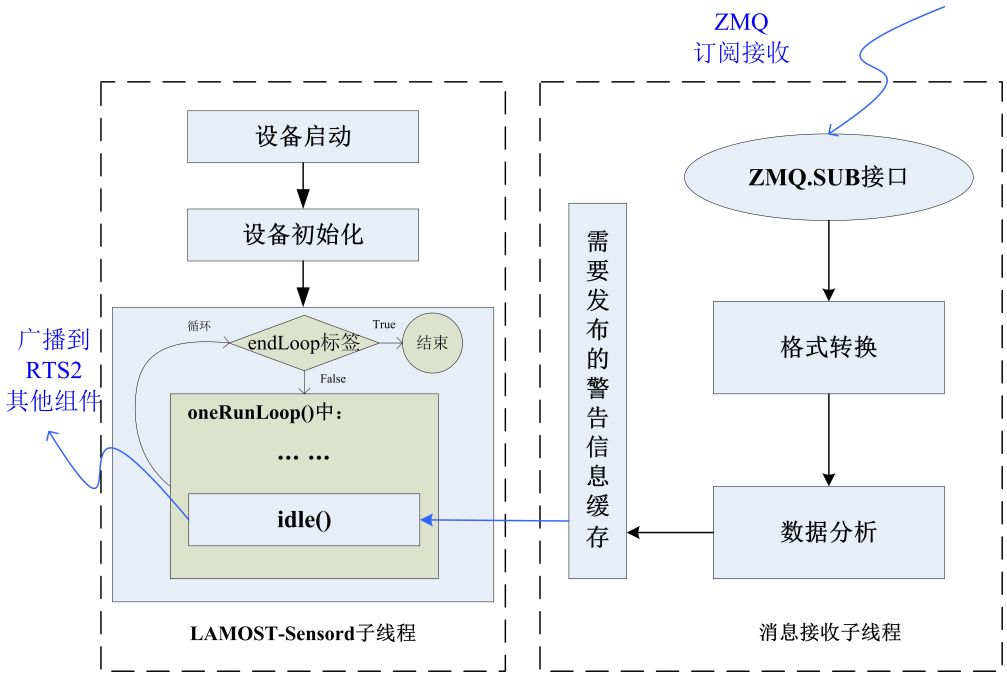

图 3.15 基于RTS2传感器设备的资源监控系统接口

元之外，还增加了数据分析单元和“需要发布的警告信息缓存”单元。资源监控系统所采集到的各节点信息数量众多、频率也较高，这些信息并不是全部需要向整个RTS2系统各个组件发布。对这些信息进行分析，以判定否已达到预警值、是否有可能提示某些节点存在隐患甚至威胁到整个望远镜的运行，这正是数据分析单元所要完成的工作。如果确实有必要对整个RTS2系统发布警告信息，则将警告信息缓存在“需要发布的警告信息缓存”单元中，以供LAMOST-Sensord进行查询。

LAMOST-Sensord子线程上运行的是继承自RTS2Sensord类的自定义类，命名为LAMOST-Sensord类。如前文所述，在RTS2设计理念中，idle()函数是实际设备常规更新操作，该函数被包含在业务逻辑函数oneRunLoop()中，一轮一轮被调用，这正是我们需要的机制。

在LAMOST-Sensord类中我们重新实现了idle()函数，在保持原有功能的基础上周期性的查询“需要发布的警告信息缓存”单元，当发现缓存中有需要发布的信息时，利用RTS2已存在的信息发布函数logStream()将该警告信息向整个RTS2系统广播。

开发完成LAMOST资源监控系统的RTS2传感器应用模块之后，将之接入整个资源监控系统中，并启动RTS2软件后，当LAMOST某节点硬件资源占用过多触发预警条件时，RTS2系统便可以收到资源预警信息。图3.16是RTS2系统监视

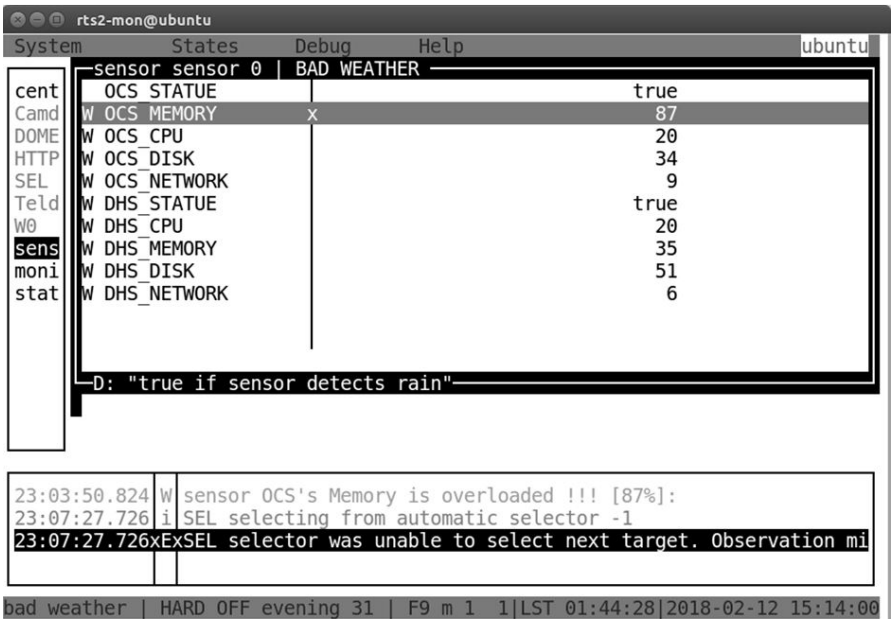

图 3.16 资源监控系统接入RTS2运行截图

器（Monitor）收到预警信息的截图。

通过对资源监控系统的开发，不但提高了LAMOST运维效率，而且进一步加深了我们对于RTS2运行机制以及二次开发的认识。将LAMOST资源监控软件映射成RTS2传感器设备并接入RTS2系统这一课题的顺利完成不但验证了我们之前提出的大型望远镜设备虚拟化概念的正确性，更为进一步将LAMOST现有的更为复杂的软件以及子系统虚拟化指明了思路。

第4章　基于RTS2框架LAMOST的CCD相机集群控制

在完成了LAMOST计算机资源监控系统向RTS2传感器虚拟化，并将之成功接入RTS2软件系统之后，我们将下一个研究目标选定为对LAMOST多CCD相机集群控制软件。之所以选择多CCD相机集群是基于以下两方面原因的考虑：

1）在RTS2框架中，相机控制类Rts2Camd的状态较少、实现复杂度较低，便于理解和扩展。而且，RTS2框架中已经提供了多种小型望远镜常用相机设备的实现类，这些类的实现方式为我们将LAMOST的CCD集群虚拟化并映射到基于RTS2的相机扩展类提供了宝贵的参考。

2）LAMOST现有CCD相机集群控制软件设计结构清晰、实现代码合理并已经平稳运行超过十年，经历了多种严格测试并完成了大量复杂的观测任务。而且本人也参与了该套软件的设计开发和测试工作，因此，对其虚拟化和映射成为RTS2相机类的准备工作更加充分。

遵循大型望远镜硬件设备虚拟化原则，结合上一章所论述的实际设备虚拟化思路，本章将着重讨论如何将LAMOST的CCD相机集群映射到RTS2相机模块这一课题。

4.1　LAMOST的CCD相机集群

作为大口径光谱巡天望远镜，LAMOST的光谱接收装置的重要性不言而喻，光谱接收装置软硬件的稳定性和运行效率将会直接影响着整个LAMOST巡天计划的实施。

LAMOST的主要光谱测量仪器为16台低色散光谱仪（当前正在测试的中色散光谱仪也将于不久后加入巡天计划中），分别安置于望远镜焦面楼的六层和七层光谱仪房中。每台光谱仪内包含准直镜、分光镜、光栅等精密光学仪器设备(Hou et al., 2010)，还包含两台电荷耦合器件（Charge-coupled Device，CCD）相机作为光谱接收终端。32台CCD相机采用的是英国电子阀公司（English Electric Valve Company，E2V）生产的203-82型科学级CCD，分辨率为4K × 4K像素(Zou et al., 2006)。LAMOST的光谱仪和CCD相机集群示意图如图4.1所示。

从运行原理上讲，望远镜接收到暗弱星光经过汇聚后由焦面板上的4000根光纤分别引入到16台光谱仪中。在每台光谱仪中，入射光束经过一系列光学仪器调整后最终分成红蓝两束光线照射到两台CCD相机上，由相机采集并记录数

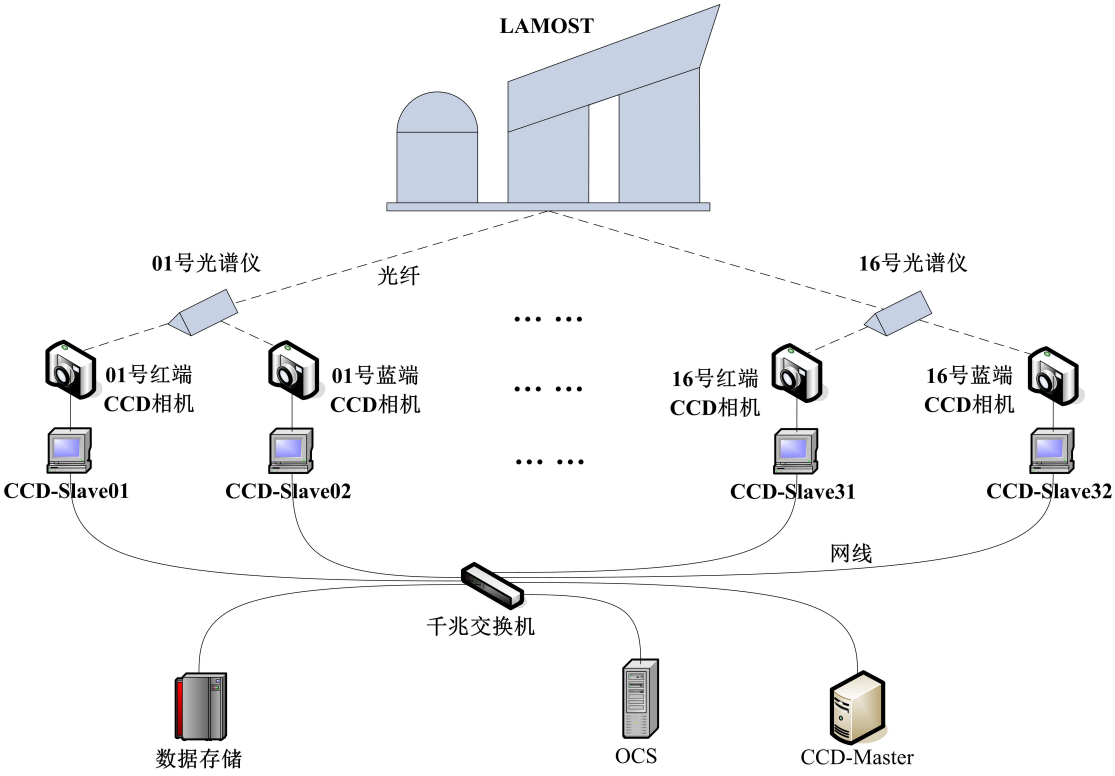

图 4.1 LAMOST的CCD集群示意图

据。光谱仪结构及光路如图4.2所示。

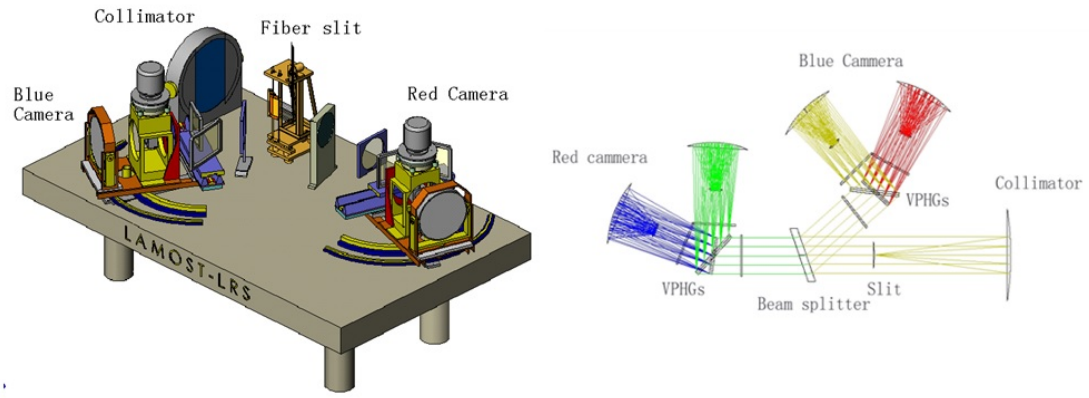

图 4.2 LAMOST光谱仪结构图（左）和光路图（右）

由32台CCD相机组成的集群由统一的软件控制，该套软件提供了命令分发、状态采集以及数据存储等功能。该套软件采用服务器-客户端模型，分层式设计，多进程和多线程相结合，并采用UDP广播结合自定义可靠通信协议来完成服务器与客户端之间的交互(邓小超 等, 2011)。整套软件于2008年开发完成，目前已平稳运行十年，基本满足了最初的工程需求。

但是，这套软件也存在着一系列不足之处，例如：UDP广播结合自定义可靠通信协议比较晦涩、调试和维护难度较大；主界面采用QT3编写与当今流行的QT5相比无论显示效果还是性能都有较大差距。更重要的是为了获得更好的自动控制能力，本课题所引入的RTS2框架并不直接支持LAMOST自主研发的相机集群控制软件。

综上所述，我们需要在保留现有软件系统优点的同时重构该软件系统，将至映射为RTS2Camd并融入整个RTS2软件体系之中。

4.2 基于RTS2软件框架的多CCD相机集群控制系统的设计与实现

本节将对基于RTS2框架LAMOST的多相机集群控制软件的设计与实现中的细节问题加以讨论，下一节将展示该软件的运行效果。首先让我们先从多相机集群控制软件在整个LAMOST软件体系结构中的位置说起。

4.2.1 CCD相机集群控制软件与OCS

LAMOST多CCD相机集群控制软件在整个观测控制系统软件体系结构中的位置如图4.3所示。

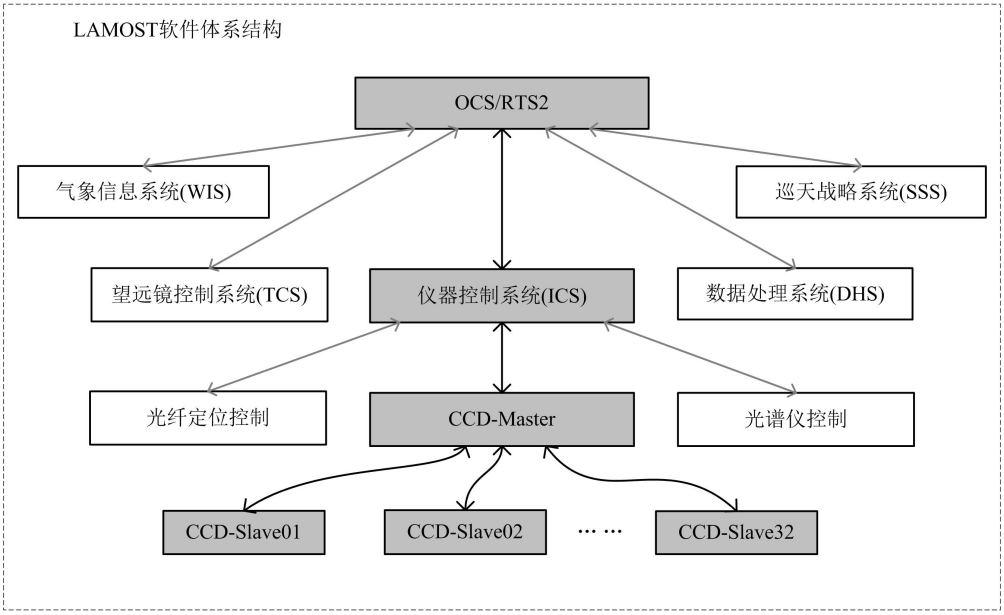

图 4.3 CCD集群控制软件在LAMOST软件体系中的位置

相机集群控制软件隶属于OCS的焦面仪器系统（ICS），由运行于服务器上的总控软件CCD-Master和运行于32台相机控制电脑上的CCD-Slave组成(Deng et al., 2010)。

在通信方面，该套软件严格按照LAMOST通信协议规定，无论是总控软件CCD-Master与OCS之间的通信还是CCD-Master与32个CCD-Slave之间的通信，都分别采用命令、状态以及命令执行反馈三种格式，实现了对需要长时间执行的操作的异步支持。这些特点也正是该套软件的优点所在。

因此，在重构该套软件的过程中，应该在尽量保持该套软件总体结构不做大的改动的前提下，将之映射到RTS2相机类上。

4.2.2 软件总体设计方案

如第2章所述，RTS2采用了面向对象的设计方法，提供可支持多种小型望远镜常用相机设备驱动类。但是，显然这些相机驱动并不包含LAMOST所使用的E2V科学级CCD相机。另外，RTS2还提供了相对简洁的用户自定义扩展机制，但是，由于LAMOST当前使用的是32台CCD相机，迄今为止也从未出现过采用RTS2直接控制如此之多的CCD相机集群的先例。因此，我们需要充分理解RTS2所提供的扩展机制，对多相机控制软件融入RTS2软件体系做出精心设计并进行仔细的测试。

为了创建用于LAMOST相机集群控制的RTS2定制相机设备模块，有以下两种设计方案可以考虑：

1）第一个方案：利用RTS2提供的自定义扩展相机机制，创建可以驱动单台E2V科学级CCD的相机类，这个类将继承子Rts2Camd类并覆盖某些接口（虚函数）以实现实际控制目的(卫守林 等, 2014b)。实际运行时，创建32个该类的对象实例分别连接各台CCD设备，并将这32个对象实例注册到Rts2Centrald上从而融入整个RTS2软件体系中。这是RTS2常规的设备扩展思路，看起来过程比较简单，但是具体到LAMOST多CCD相机集群这一应用情景中，这种方案隐藏着致命的缺陷和很高风险(Kubánek et al., 2012)。理由如下：

RTS2软件系统最初是为小型望远镜而设计的，其模块之间的通信机制采用的是基于TCP的网状拓扑结构（而不是星形拓扑结构）。随着系统中模块数量的增加，模块之间的TCP连接将成几何级数增长。过多的TCP连接不仅会占用大量额外的网络带宽（许多冗余信息会在各条TCP连接上传输），而且降低了整个RTS2系统的易用性和稳定性。

2）第二个方案：添加一个虚拟层用来桥接RTS2系统与实际的32台CCD相机。在此方案中，我们只创建一个用户自定义的RTS2（虚拟的）相机模块并将之注册到RTS2系统中，就像一台普通的简单相机设备。然后，添加一个虚拟层

（或称为代理），由该虚拟层来完成将RTS2命令转译分发以及32台实际相机的状态采集和汇总工作。通过添加虚拟层，不但降低了RTS2系统和32台CCD相机之间的耦合性，提高了整套软件的稳定性和扩展性，而且，我们可以充分利用现行多相机控制软件的经验，避免重复开发，简化开发难度，这也符合代码重构原理(MartinFowler, 2015)。

显然，第二个设计方案更符合大型望远镜设备虚拟化的概念，为我们的开发带来更多好处，因此，选择了第二种方案来实现基于RTS2框架的LAMOST多相机控制软件。

多相机控制软件的总体设计框架如图4.4所示。整套系统分为三层：中心控制层、虚拟设备层和实际控制层。

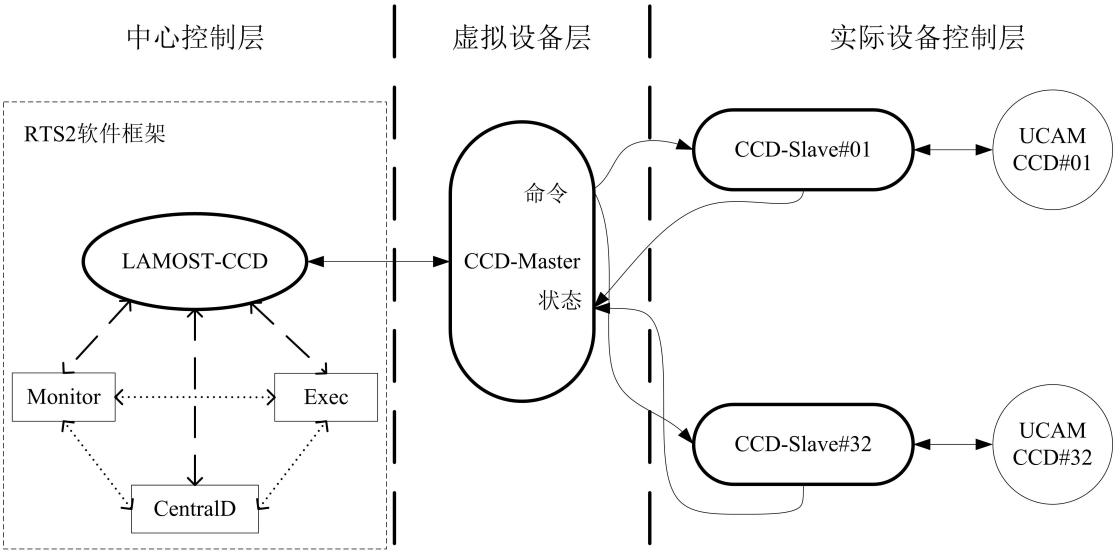

图 4.4 CCD集群控制软件总体框图

在中心控制层中，我们创建了继承自Rts2Camd类的自定义相机类，名为“LAMOST-CCD”，用于与RTS2系统其他模块进行交互并对RTS2命令进行相应处理。用主从（Master-Slave）范式设计虚拟设备层和实际控制层。虚拟设备层包含一个主设备，即CCD-Master，它将RTS2命令转译并分发给实际控制层，还负责收集实际设备层发来的相机各相机状态并汇总成虚拟相机状态。在实际的控制层中，CCD-Slave作为从属设备来实现。

图4.4中UCAM软件是E2V科学级CCD相机的驱动软件，它可以提供了操作者与单台相机交互的功能。

这种设计方案不但保留了RTS2的优点，例如目标调度、控制自动化和容错。而且让我们有机会重构现有软件，充分保留现有软件前台显示/后台逻辑分

离、命令/状态分等优点，简化设计难度，加快开发速度。

4.2.3 基于RTS2的相机类的扩展

从RTS2软件系统角度分析，LAMOST-CCD类是一个RTS2相机模块的实现类，它继承自rts2camd::Camera类，其类继承关系如图4.5所示。

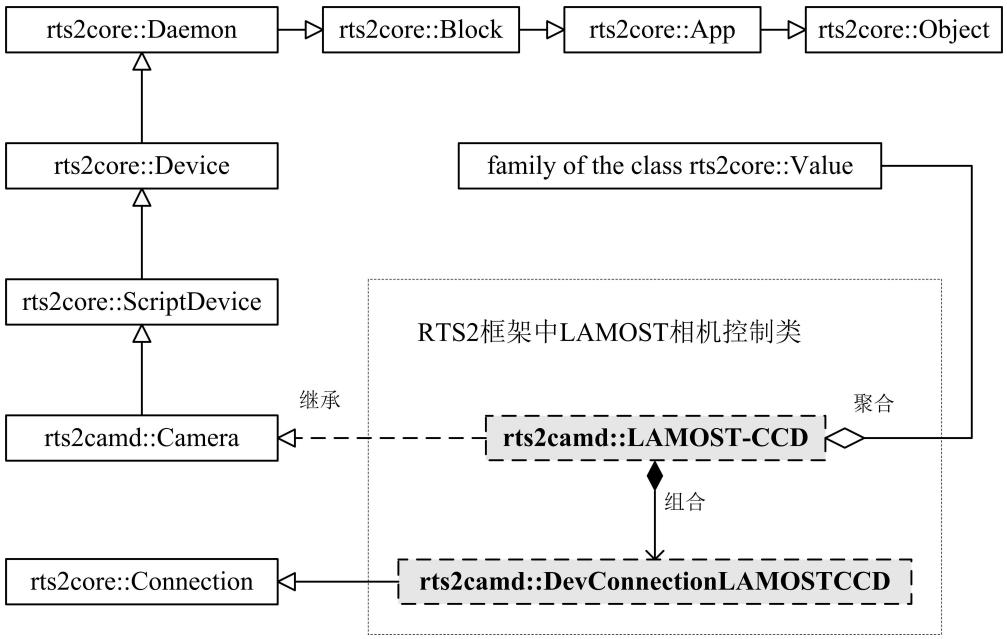

图 4.5 LAMOST-CCD类在RTS2框架中的继承关系图

其中，rts2camd::Camera类是由RTS2提供的通用相机模块类，作为各种相机的抽象类。它总结和简化了天文观测过程中相机的一般控制步骤和状态，并提供了大量的接口（虚函数）以便二次开发扩展功能。在本课题中，LAMOST-CCD类通过覆盖少量的虚函数（例如：传输自定义命令和状态），就能够使之被RTS2系统控制，融入整个RTS2自动化控制中去。有关自定义命令和状态详细信息，请参见第4.2.6节。

如前文所述，RTS2软件体系使用TCP连接作为内部模块之间的交互通道，并采用了自己的通信协议，其核心通信类是rts2core::Connection(Kubánek et al., 2008)。为了满足RTS2的通信要求，我们创建了继承自rts2core::Connection类的自定义通信连接类，名为“DevConnectionLAMOSTCCD”。并覆盖processLine()、setState()等接口函数，来达到管理TCP连接和实现定制功能的目的。扩展该类的设计如图4.6统一建模语言（Unified Modeling Language，UML）图所示。

遵循RTS2运行机制，作为独立运行模块被启动时，LAMOST-CCD会尝试通过将自己注册到RTS2中控服务CentralD上而接入RTS2系统。同时，它还会创建

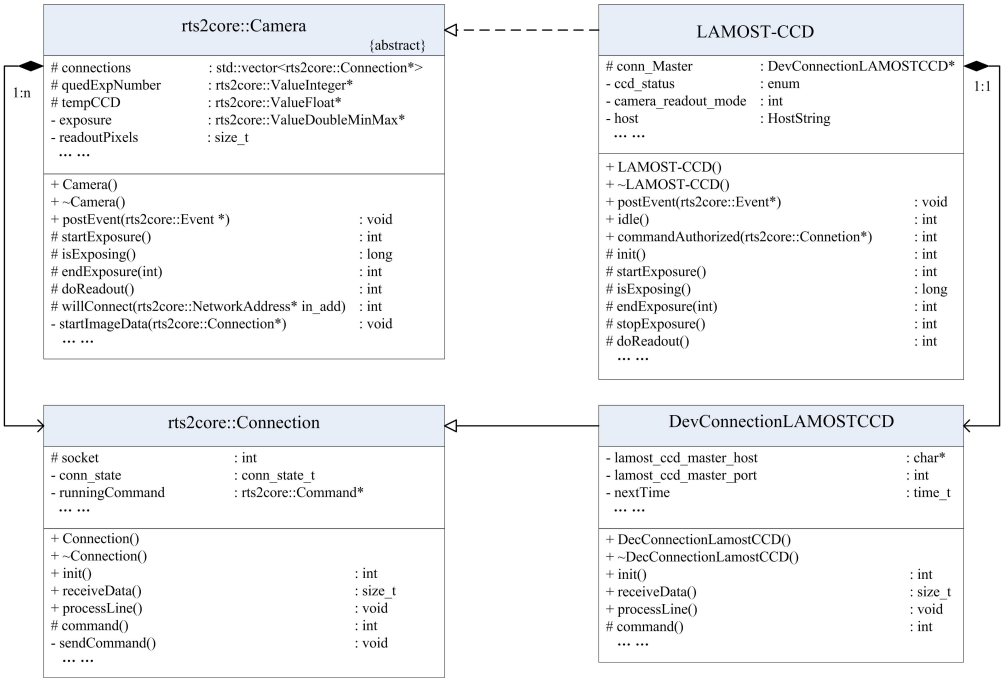

图 4.6 LAMOST-CCD类设计的UML图

一个DevConnectionLAMOSTCCD类的对象实例，并尝试连接到CCD-Master上。当此连接建立后，LAMOST-CCD将该对象实例添加到自身的连接队列中。随后，LAMOST-CCD的运行就和其他RTS2设备模块完全相同了。

当RTS2的执行器EXEC或者客户端向LAMOST-CCD发送命令时，后者通过由DevConnectionLAMOSTCCD对象实例管理的TCP连接向CCD-Master转发命令。当CCD-Master采集多台相机实际状态并据此产生一个虚拟状态时，它会将该虚拟状态转换成RTS2框架中自定义事件（EVENT_LAMOST_EXPOSURE_START、EVENT_LAMOST_READOUT_END等），通过DevConnectionLAMOSTCCD对象实例传递回LAMOST-CCD。最后，作为RTS2设备模块的LAMOST-CCD改变自身状态并执行了适当的后期处理。

由于深入理解了如前文所述的RTS2提供的Daemon运行机制、连接管理机制，事件处理机制，我们可以非常简单快速地实现LAMOST-CCD类。

4.2.4 总控软件CCD-Master的实现

如总体设计方案中所述，为了实现LAMOST多CCD相机控制软件到RTS2相机模块的映射，我们增加了一个虚拟层CCD-Master。为了保持命令/状态分离的异步操作能力，CCDMaster需要保持两个方向的通道：

1）命令通道，CCD-Master（作为socket服务器）需要接收来自于RTS2相机LAMOST-CCD的连接并能够接收其发来的命令。它还需要能够对接收到的命

令做出相应处理（转译、设置特殊参数等）。最后，它（作为socket客户端）也需要能够连接到各个CCD-Slave上去并分发处理后的命令。

2）状态通道，CCD-Master（作为socket服务器）接收来自于CCD-Slave的连接并能够接收来其发来的实际设备状态。然后，它判断所有相机是否工作正常。最后，它将集群完整的状态报告作为一个虚拟相机的状态提供给LAMOST-CCD。 CCD-Master的内部结构如图4.7所示。为了保证命令通道和状态通道不相互阻塞，实现中CCD-Master采用了双进程运行：命令子进程、状态子进程。

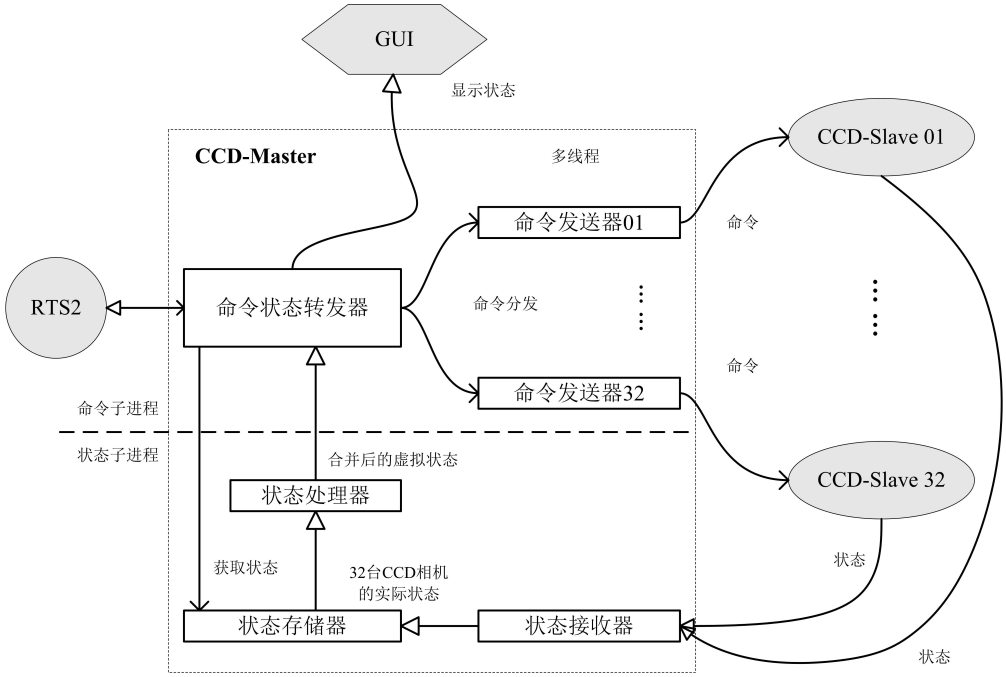

图 4.7 CCD-Master内部结构图

在命令通道的设计中，CCD-Master应该精确地同时向32台CCD相机分发命令，特别是当命令是“开始/停止曝光”时。但是，由程序操作原理可知，这一设想很难非常精确地实现。也是基于这种设想，现行多相机控制软件采用了UDP广播的方式，以期达到精确同步的效果，但实测结果并不理想。

幸运的是，LAMOST进行夜间观测时，通常为长时间曝光（600秒-1800秒），而单机驱动软件UCAM的时间敏感度为0.01秒。因此，我们放弃了现行多相机控制软件中所采用UDP广播结合自定义可靠通信协议的思路，选择基于TCP的多线程命令分发设计方案。

该方案已经可以达到工程需求，多线程之间的时间差在工程中可以忽略不计。且该方案性能稳定高，开发成本低。已有益于课题的进展。相关测试结果见4.3.1节。

在状态通道的设计中，由于对状态收集的时间精确度要求并不高，为了简化开发过程，我们只使用一个线程。状态接收器用于处理socket相关问题接收32台CCD发来的实际状态。添加状态存储器来缓存实际状态。如图4.7所示，当命令状态转发（命令子进程）器发来“获取状态”命令时，状态处理器分析状态存储器中的实际相机状态，合并后生成虚拟相机状态反馈给命令状态转发器。状态接收性能测试见4.3.2节。

为了保持前台显示与后台逻辑分离的设计理念，CCD-Master还包含了本地图形界面(GUI)组件。GUI组件可以为操作者提供更详细真实相机状态的功能。见4.3.4节。

4.2.5 从属软件CCD-Slave的实现

CCD-Slave是CCD-Master和UCAM之间的桥梁，它运行于每台CCD相机控制电脑上。CCD-Slave负责接收CCD-Master发来的命令并发送实际相机状态，且作为桥梁沟通E2V科学级CCD相机驱动软件UCAM。图4.8描述了CCD-Slave的内部结构。

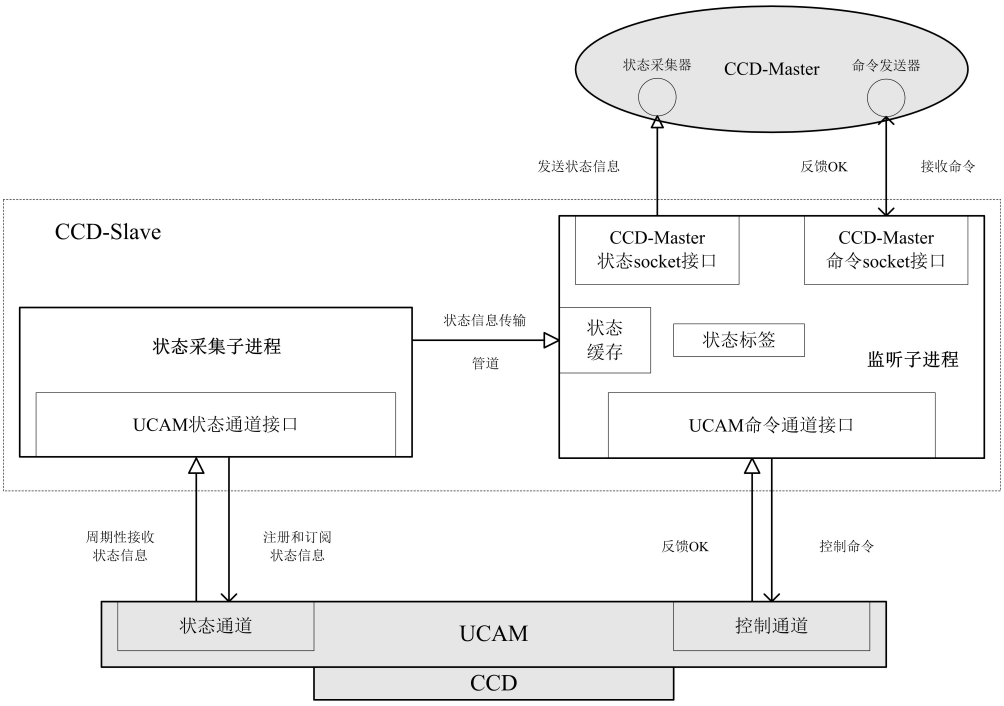

图 4.8 LAMOST-CCD类在RTS2框架中的继承关系图

这种设计可以尽可能的保留现有软件的原始接口，为长期稳定运行的大型望远镜现有软件的升级带来极大的益处。

与CCD-Master一样，CCD-Slave也有两个独立的进程，一个进程用于管理命令，另一个进程用于处理状态消息。命令监听进程创建socket监听端口，接收CCD-Master发来的命令，并将命令转换为UCAM可以理解的特殊字符串（或二进制代码）。为了减少大量底层信息的传输，默认情况下CCD-Slave的命令监听进程的状态标签为False。只有当CCD-Master发来“获取状态”命令时，该状态标志才会变为True。然后，将缓存的实际相机状态信息从状态缓存中取出发送给CCD-Master。

状态收集器进程将自己注册到UCAM信息通道并订阅某些状态主题。然后，它会不断收到UCAM发来的消息。当收到新的状态消息时，状态收集器通过管道将其推入状态缓冲区中，管道是连接两个进程的桥梁，它在CCD-Slave启动时即创建。

4.2.6 内部通信协议

如前文所述，RTS2采用了自定义的通信协议，该协议通过TCP/IP套接字发送美国信息交换标准代码(ASCII)字符串(Kubánek et al., 2008)。该协议简单、快速且功能强大。作为设备模块，LAMOST-CCD直接使用该协议与其他RTS2模块进行通信。而LAMOST-CCD与CCD-Master之间的通信我们采用了基于RTS2协议的自定义命令和状态信息。表4.1列出了主要的自定义命令和状态。

表 4.1 LAMOST相机集群控制软件内部通信协议

格式	描述	发送者	接收者
GV <key>	获取参数	LAMOST-CCD	CCD-Master
XV <key> <value>	设置参数	LAMOST-CCD	CCD-Master
EX	开始曝光	LAMOST-CCD	CCD-Master
SX	停止曝光	LAMOST-CCD	CCD-Master
RD	开始读出	LAMOST-CCD	CCD-Master
A <message>	警告信息	CCD-Master	LAMOST-CCD
+ <message>	成功信息	CCD-Master	LAMOST-CCD
- <message>	失败信息	CCD-Master	LAMOST-CCD
...	...	...	...

为了降低开发难度，我们使用键值模式来传输和存储各种参数。这种方法的实现非常简单也非常灵活。使用这种技术，我们可以在整个代码开发过程中

轻松添加和修改参数。例如：当LAMOST-CCD接收到来自RTS2内部模块的“设置参数”命令时，LAMOST-CCD会先根据表4.1将该命令转译并添加“键值”对（例如，<selected = all>）后，在将命令发送给CCD-Master。 CCD-Master接收到该命令，解析该命令并获取“selected”值，然后将命令转发给所有CCD从器件。每个CCD-Slave接收并解析命令，然后或驱动UCAM反馈实时状态或将自身缓存的已有状态转发并反向逐层返回给LAMOST-CCD。

4.3 多CCD相机集群控制软件的部署、测试与运行

在基于RTS2框架的LAMOST多相机控制软件开发完成后，我们将该软件部署于如图4.1所示的LAMOST的真实工程环境当中。 32台EEV科学级CCD相机从16台低色散光谱仪光路中获取光信号。每台CCD控制电脑均相同并放置于6个机柜当中，其配置如下： Intel(R) Core(TM) i3-2100 3.10-GHz中央处理器（CPU）和4GB大小的DDR3内存。CCD-Master服务器具有2颗Intel(R) Xeon 2.53-GHz处理器（共包含16个逻辑CPU内核）和12GB大小DDR3内存条。LAMOST-CCD电脑的硬件配置与CCD控制电脑相同。所有计算机都连接到千兆位交换机，并在每台计算机上安装CentOS6.8操作系统。

我们设计了一系列实验来测试新控制软件的性能，以验证该软件是否能满足LAMOST的实际观测要求。测试结果在下小节中介绍。

4.3.1 命令分发测试

如前文所述，在现有的多相机控制软件中，使用了UDP广播机制，期望命令可以同时到达每台CCD相机。但UDP本身属于无连接的不可靠的通信协议，为了提高UDP的可靠性，还增加了自定义可靠协议等辅助代码(W.RichardStevens等, 2006)。然而，在测试过程中，数据包丢失和乱序问题仍会时有。在实际工程环境中，特别是当整个网络的信息吞吐量较大或网络负载较重时，这种问题尤其严重。另外，这些辅助代码使得整个软件难于理解并并增加了维护和再次扩展的难度。

另一方面，与UDP技术相对应的TCP提供面向连接的、可靠的、有序的和自带错误检查的流通信技术。如果采用TCP作为连接技术，可以克服LAMOST高速局域网中UDP不可靠的缺点。

从技术角度来看，TCP比UDP效率低。然而，实验表明经过仔细的设计，TCP可以提供接近于UDP的效率，这完全符合LAMOST工程要求（见4.3.1节）。

而且，使用TCP作为传输协议可以使软件既简单又稳定。此外，TCP的使用简化了RTS2框架与新型LAMOST多相机控制软件集成。

这就是我们在基于RTS2的新型LAMOST多相机控制系统中，采用TCP传输协议代替原来采用UDP广播结合自定义可靠通信协议的原因。

我们设计第一个实验来测试基于TCP的命令分发通道的性能，并确定新的控制软件是否满足工程要求。本小节中还讨论了现有的多相机控制软件的基于UDP的性能，以便与升级后的软件进行比较。

生成随机字符的命令数据包（8B，2B，...，1KB），分别经由现有多相机控制软件和新系统分发并进行测量。为了保证实验的准确性，每种大小的模拟都进行了5次测试，每次测试分发100万次，最后去均值。

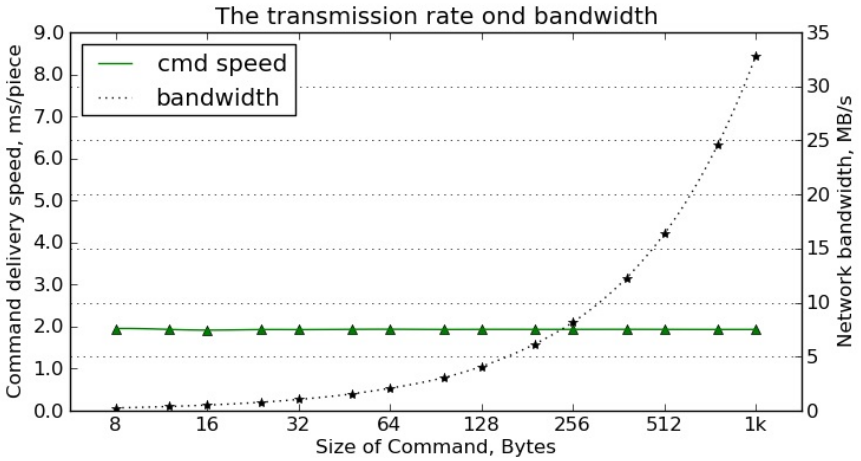

图 4.9 基于UDP广播的原相机控制软件的命令分发速度与带宽占用测试结果图

图4.9给出了基于UDP广播结合自定义可靠通信协议的原有软件的命令分发速度和网络带宽占用率。

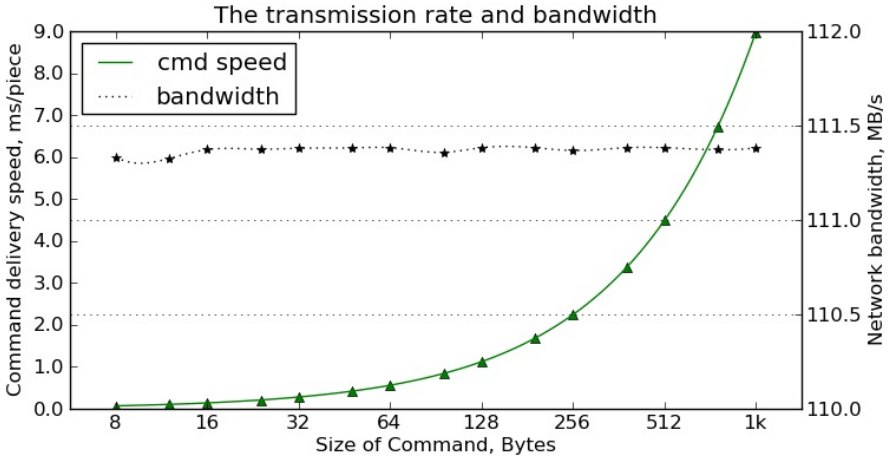

图 4.10 基于TCP多线程的新相机控制软件的命令分发速度和带宽占用测试结果图

图4.10则给出了基于TCP结合多线程技术的新相机控制软件命令分发速度和网络带宽占用率。

将两者测试结果进行对比可以发现：基于UDP的原有软件的命令分发速度恒定在1.94 毫秒/条，网络占用率随着命令包大小的增加而增加。基于TCP的新软件，由于采用了多线程技术，网络占用率总是接近网络带宽极限（千兆交换机的极限带宽为最终速度是10^9bit/s÷8≈119.21MB/s），而命令分发速度随着命令包大小增大而增长。

从这两个图的比较可以看出，当命令包小于200B时，新系统表现出的性能优于原有系统。相反，当命令包超过200B时，原有系统性能更高。

即使命令包大小增加到1KB时，新系统的传输速度也高达8.94毫秒/每条，这种速度远低于UCAM时间敏感度（100毫秒）。因此，两个软件系统的命令分发性能上能满足LAMOST的实际工程需求。在实际观测过程中，CCD相机控制命令通常都小于200B，因此，使用TCP优于UDP。

考虑到使用TCP不会有数据包丢失或乱序数据包的风险。并且TCP技术相对应用更加广泛和成熟，业界也有大量流行的基于TCP的通信库或封装包，利用这些成熟的库可以进一步减少软考开发和维护的工作量。

因此，综合以上多方面因素，我们可以得出结论：新系统在多相机命令发送方面可以更有效地满足LAMOST的实际工程要求。

4.3.2 状态采集测试

在设计和开发过程中，为了简化开发难度，在CCD-Master中采用了单线程单端点来接收和处理所有状态数据包的方案。因此，在第二个实验中，我们将测试新系统的状态采集性能，以确定它是否满足收集和处理32台CCD相机的状态信息的需求。

我们为每台相机产生随机内容字符串的状态数据包（8B，12B，...，1KB）。然后，所有这些状态数据包被同时发送给新系统接收。与第一个实验类似，为了保证实验的准确性，对每种大小的模拟状态数据包进行5次测试，每次测试每台相机连续发送100万个状态包，测试和记录状态传输速度和占用网络带宽率。

图4.11为新系统状态接收处理速度和网络带宽占用率测试结果。从图中可以看出，和命令分发实验相似，状态处理速度随着状态包的大小而增加。然而，网络占用率始终不是很高，在状态数据包大小为48B时网络占用率达到峰值92.3MB/s。这种现象的原因是：由于采用的是单线程，前期由于通信包小，网

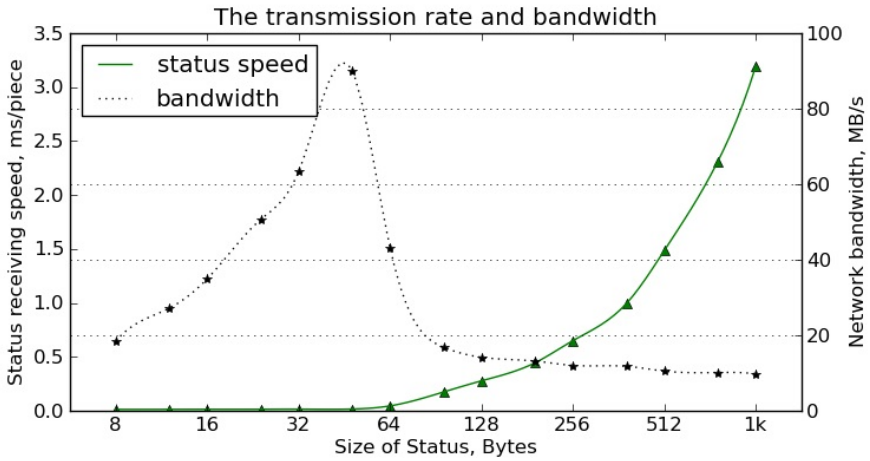

图 4.11 基于TCP和单线程的新相机控制软件的状态接收速度和带宽占用测试结果图

络占用率主要程序自定义的判定状态发送者逻辑限制，而后期则主要受到IO阻塞以及系统调用导致的多次用户态内核态切换的限制。但是，即使状态包大小为1KB，状态接收和处理速度也仅为3.2毫秒/状态。

实际观测对于状态处理速度要求并不高，且单台相机状态产生速度约为秒级。因此，我们可以得出结论：新系统在多相机状态接收处理方面能够满足LAMOST的实际工程要求。

有关原有系统基于UDP的状态接收性能和丢包率在董健 等 (2009)已做出过详细论述，在此不再赘述。

4.3.3 容错和出错处理机制

作为一个工程项目，容错和异常处理机制非常重要，它确保了整个软件系统的稳定运行，并在必要时提供了人机交互机制。新的基于RTS2的LAMOST多相机控制软件的容错和异常处理机制控制系统主要包含以下几方面：

1）新相机控制软件的LAMOST-CCD模块作为RTS2框架中的一个相机设备模块，直接继承了RTS2的容错能力。遵循“即插即用”理念，所有RTS2独立运行组件都具有容错能力，即一台设备发送故障不会影响整个RTS2框架下其他设备的运行。在实际观测过程中，我们可以任意添加或移除我们的LAMOST-CCD模块，而不影响整个系统的运行。

2）在CCD-Master中，为了保证数据有效性，我们在状态处理器内为每一个CCD-Slave的连接都分配了一个定时器，以定期跟踪关键操作步骤。例如，在读出过程中，每个CCD-Slave每秒都会传输读取进度（由UCAM生成）的消息。如果某个CCD-Slave在指定的时间内未更新相应信息，定时器会监控并创建警告

信息。警告信息将通过RTS2的消息机制logStream()函数广播给其他RTS2模块。它也可以驱动GUI弹出消息框提醒观察者。

3）遵循RTS2连接范例，CCD-Slave和CCD-Master之间的连接允许重新连接。当某台CCD相机遇到严重的硬件问题时，工程师可以关闭该CCD-Slave并自动连接。问题解决后，该CCD-Slave可以重新启动并自动重新建立连接。这个过程不影响其他正常CCD的操作。

虽然软件开发尚处于初级阶段，但由于增加了容错和异常处理措施的考虑，能够为新相机控制软件系统的稳定运行提供了必要的保证。在今后的开发和维护中，根据实际需求还可以进一步增强整套系统的容错和错误处理功能。

4.3.4 实际运行

我们尝试利用RTS2软件来驱动新的多相机软件，实现对32台相机曝光读出等操作的控制。实际运行中，由于其他子系统尚未实现虚拟化，因此，望远镜指向模块采用的是RTS2自带的伪指向模块（Rts2Teld-dummy）来实现。光源则采用LAMOST定标灯代替实际的星光。

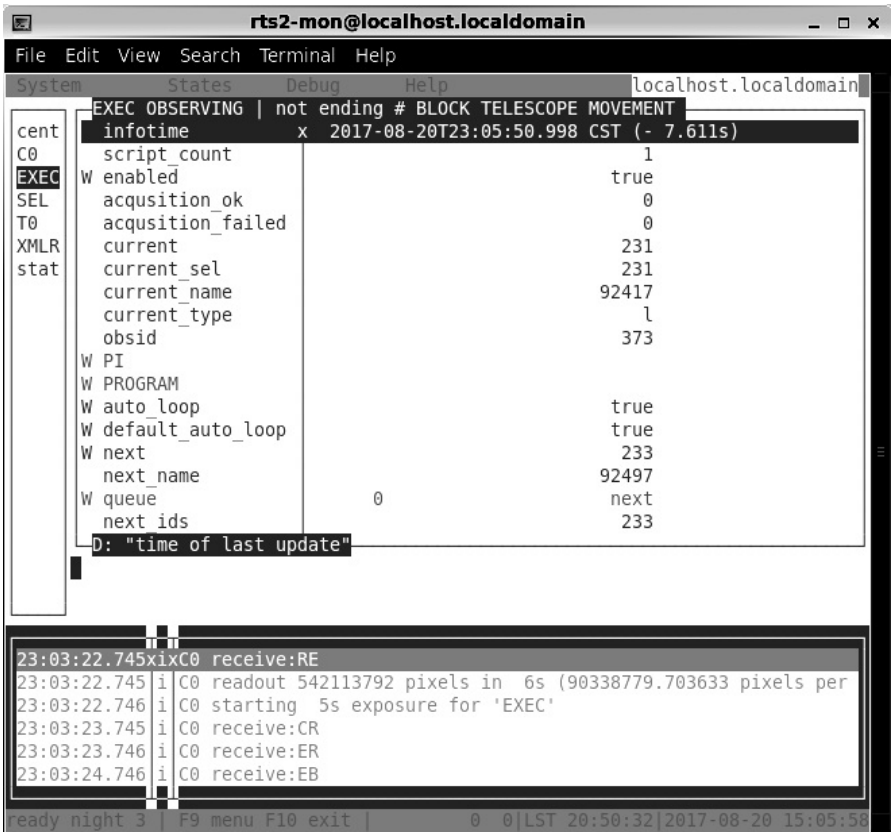

图 4.12 RTS2驱动相机集群曝光过程中的Monitor截图

图4.12为RTS2控制新相机控制软件驱动32台CCD进行曝光时，RTS2提供的

基于终端的监视器Monitor的截图。

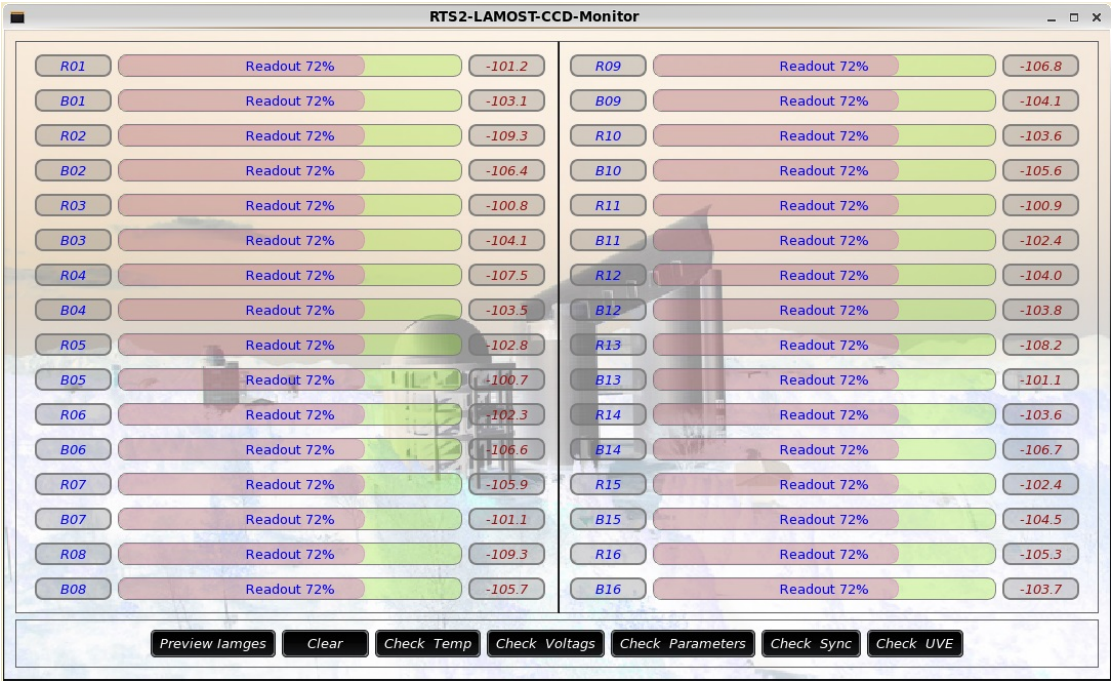

图 4.13 相机集群控制软件本地GUI运行时截图

图4.13为RTS2控制新相机控制软件驱动32台CCD进行读出时，新软件的本地图形界面运行截图。参考原有系统中命令状态分离的设计思路，新控制软件也拥有独立运行的GUI程序。新的图形界面采用QT5进行编写，并利用QSS进行界面效果美化。

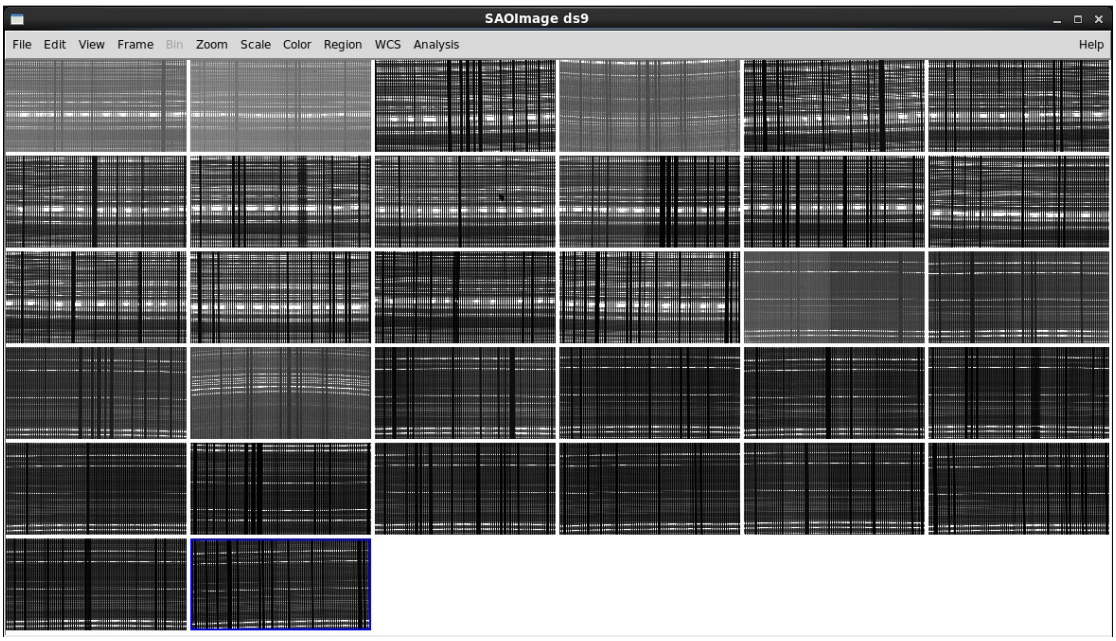

图 4.14 相机集群获得的灯谱图像数据

图4.14为RTS2控制新相机控制软件驱动32台CCD拍摄的灯谱数据。

综合本章所述，按照大型望远镜设备虚拟化思路，结合LMAOST多相机控制自身特点，在深入分析基于RTS2框架的相机二次开发机制后，我们完成了对现有相机集群控制软件的重构，使之融入到RTS2软件体系中，从而获得RTS2软件自动化控制的能力。

第5章　观测计划入库、光纤定位接口映射与模拟测试

在大型天文望远镜设备虚拟化思路的指导下，在完成了资源监控系统映射为RTS2传感器模块，以及多相机控制系统映射为RTS2相机模块的尝试后，为了能够尽快使RTS2框架融入整个LAMOST观测控制系统领域，从而验证RTS2系统自动化控制能力能否为LAMOST所用，我们将目标转向观测计划入库和光纤定位控制系统上。

为了加快开发和实验速度，我们对两者只进行了原型开发。虽然这种原型系统并不能直接应用于实际工程环境，但是其运行机制完全符合RTS2框架的原理。而且，其设计思想和实现技巧会对今后实际开发过程产生积极的指导意义。

5.1　LAMOST的巡天战略系统以及光纤定位软件的介绍

在LAMOST巡天观测过程中，望远镜每次观测的目标天区和4000个光纤单元的目标位置都是由观测计划文件包指定的。观测计划文件包是由一系列不同格式的目标文件组成的，不同文件面向不同子系统的，其中最重要是面向光纤定位软件的XML格式文件、面向机架焦面指向跟踪的TXT格式文件和面向导星的TXT格式文件。

由于本文并未涉及LAMOST机架焦面系统和导星系统的虚拟化，因此，以下重点讨论面向光纤定位系统的XML格式观测计划文件入库与使用方法。

5.1.1　巡天战略系统

LAMOST观测计划文件是由巡天战略系统（Survey Strategy System，SSS）自动产生的。SSS系统作为LAMOST的一个子系统，它的主要作用就是根据望远镜的科学目标和观测条件，以最优化的方式在尽可能短的时间内，制定出可行的观测计划的系统(褚耀泉, 2007)。它是联系天文学家与观测控制系统的中间桥梁，并能根据观测条件变化、观测需求的变化和观测结果调整计划生成。

考虑到LAMOST巡天任务的特殊性，SSS系统必须解决以下问题(袁海龙, 2011)：

1）整体巡天覆盖面积大，达到上万平方度，且要求达到一定程度的天区覆盖完备率。

2）观测目标多，预计超过千万，且观测目标来自多个课题组，具体较大的变动性。

3）观测数额大，每次观测量接近4000，而每个观测夜能进行7-15次计划，一天的观测量在30000以上。

4）观测约束条件复杂，包括一系列的静态和动态的主客观约束条件，天区目标的选择与分配需要综合考虑各方因素。

因此，SSS系统需要负责管理观测目标，处理选星约束，生成观测计划和部署巡天过程。其具体功能涵盖了巡天星表的管理和维护，观测约束条件的分析与优先策略的指定，天区覆盖和光纤分配，观测计划的生成与反馈等范围(任间等, 2007)。

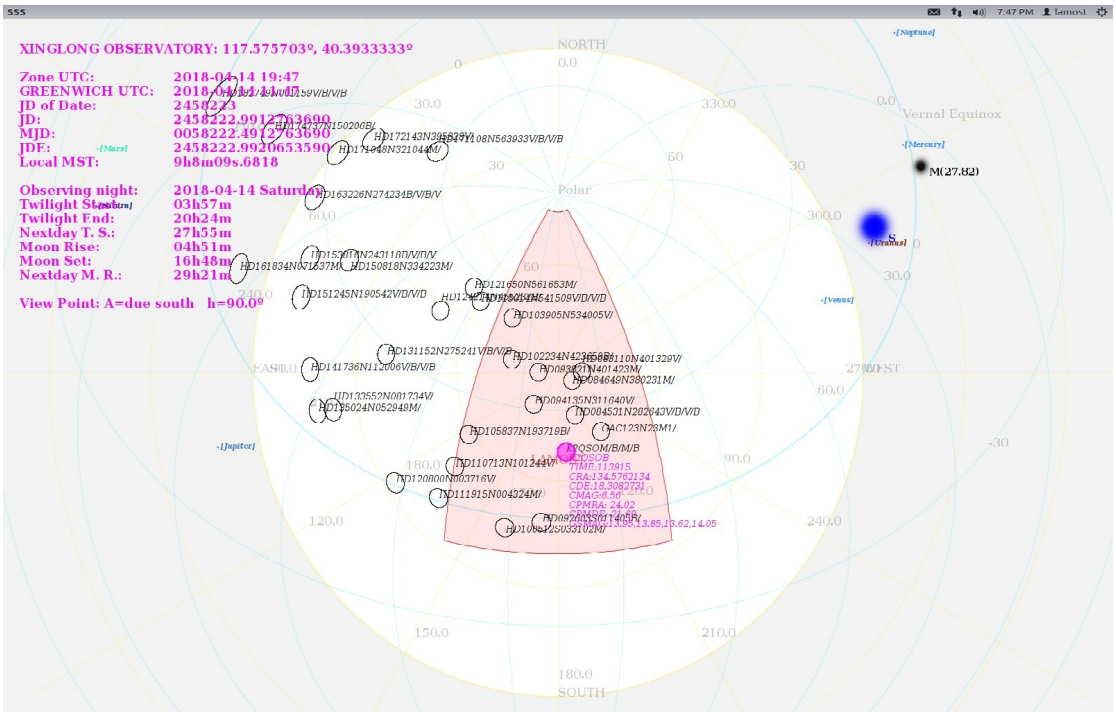

图 5.1 SSS系统运行时GUI截图

目前，SSS系统自动化运行，利用高效的选择算法结合诸多实际约束条件为LAMOST每夜的巡天观测产生多个观测计划文件包。图5.1所示为SSS运行界面。图中，白色圈显示的是当夜备选观测计划文件所对应的天区，紫色圈则表示当前正在执行观测的天区，而粉色扇形区域则表示当前LAMOST望远镜所能观测到的天区投影。

从图中可以看出，相同观测时间段内不同纬度都有备选观测计划文件，且相同天区也尽量产生不同亮度等级的观测计划文件以适应实际观测气象以及仪器运转效率。

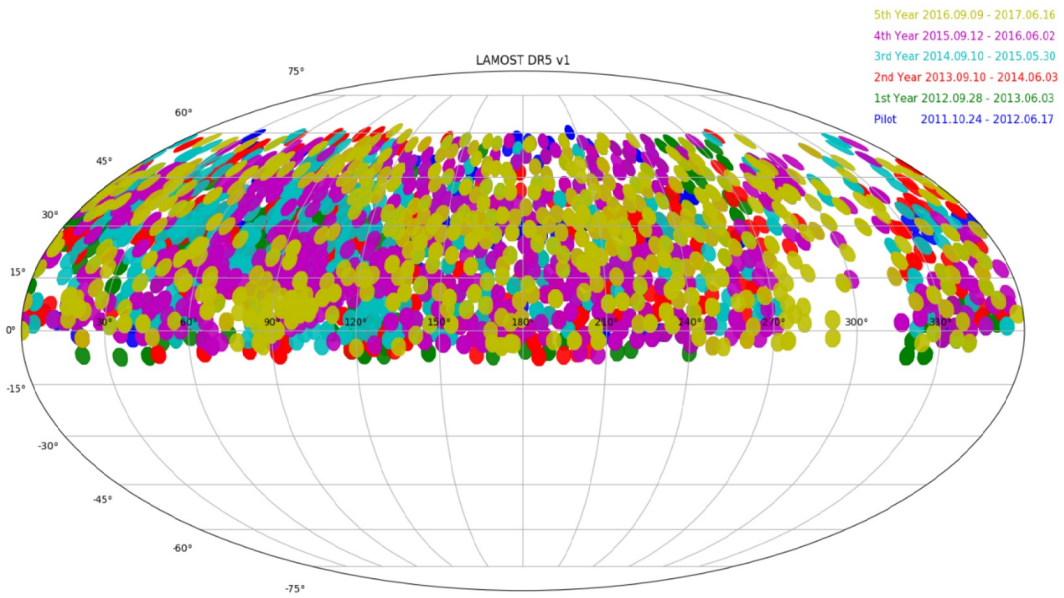

图 5.2 LAMOST已观测天区在天球上的分布图

图5.2描述的是LAMOST自2011年10月先导巡天开始到2017年6月第一个五年巡天计划圆满完成并成功释放数据发布版本5（Data Release 5，DR5）以来已经观测和处理过的天区分布情况。

5.1.2 光纤定位软件

LAMOST作为国际领先的光谱巡天望远镜，其中一项重要技术就是利用并行技术同时驱动焦面板上的4000个光纤定位单元对准同一天区内不同的目标，从而达到极高的数据获取率。LAMOST焦面仪器设备如图5.3所示。

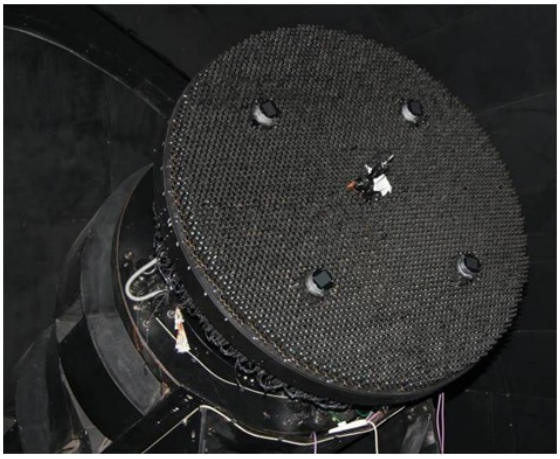

图 5.3 LAMOST焦面板实物图

相对于国际上传统的磁扣法和孔板法，LAMOST光纤定位采用了创新性的分区并行可控方案和双回转光纤定位单元技术。该新方法的使用降低了人工操

作强度，减少了光纤定位运行时间，从而提高了望远镜观测运行效率(邢晓正 等, 2007)。

LAMOST光纤定位装置的软硬件研发工作是由中国科技大学来承担的。其主要解决的问题为(Li et al., 2016)：

1）每根光纤定位误差小于40微米。

2）光纤头端面不能离焦。

3）4000根光纤在整个直径1.75米的焦面板上无盲区。

4）4000根光纤定位单元运行过程中有防碰撞保障。

5）每次4000根光纤重新定位是时间小于10分钟。

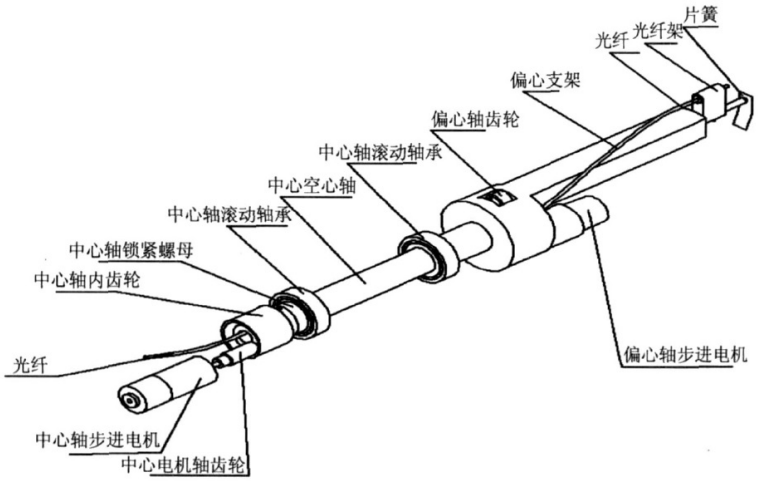

图 5.4 LAMOST双回转光纤定位单元机械结构示意图

图5.4为单个双回转光纤定位单元机械结构示意图。

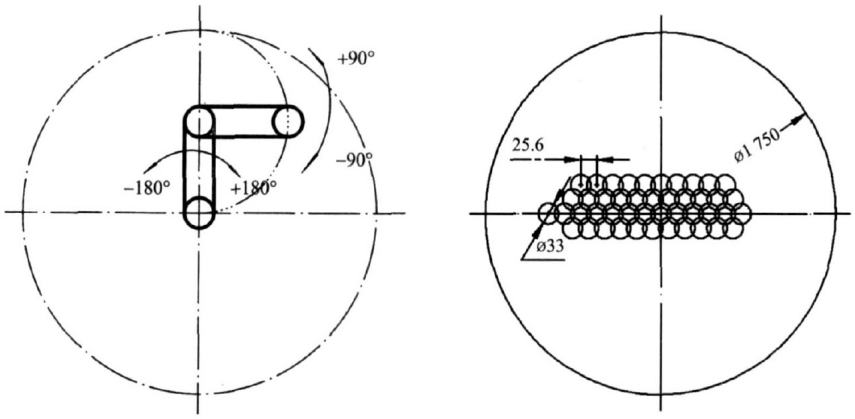

图 5.5 光纤定位单元运行轨迹（左）和排布图（右）

图5.5描述了单个光纤定位单元在焦面上的运动轨迹和范围以及4000个光纤单元在焦面板上的排布。

光纤定位系统有自己的主控软件，如图5.6所示，可以通过本机图形界面主控软件进行总体控制和任选单元的单步调试(刘志刚 等, 2011)。

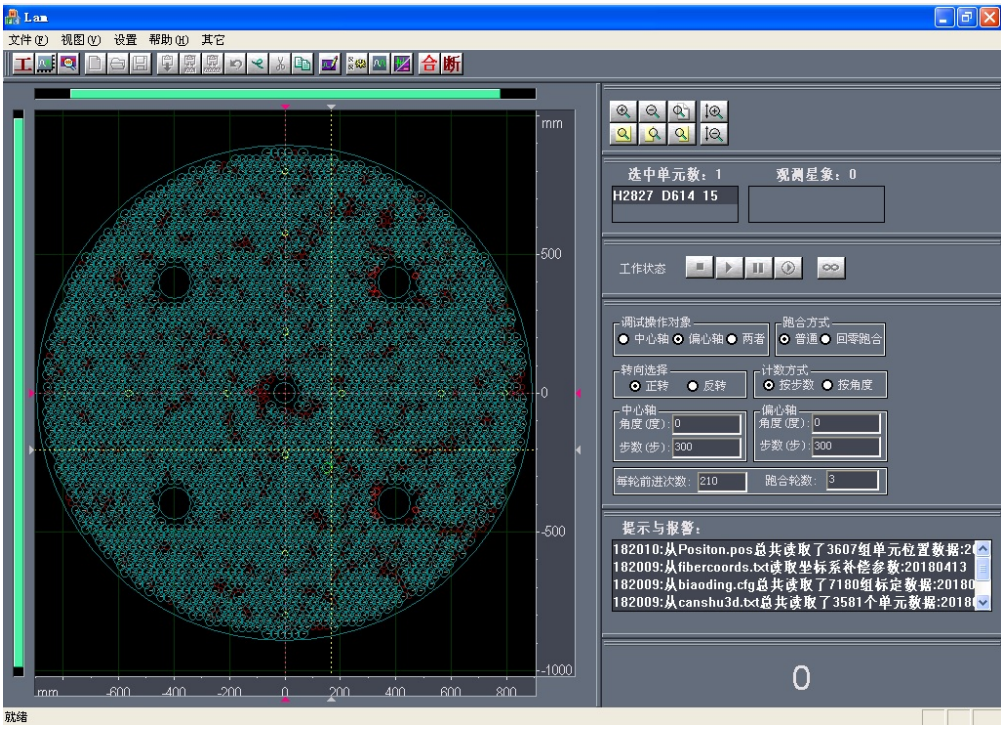

图 5.6 光纤定位软件GUI截图

光纤定位系统的主控软件根据LAMOST通信协议与OCS系统进行交互。目前，夜间观测过程中，由OCS将面向观测计划文件传输给光纤定位主控软件，并根据命令协议驱动其完成光纤定位操作，并从光纤定位主控软件处获取单元运行信息。

具体面向光纤定位软件的观测计划文件格式参见附录B，OCS系统与光纤定位主控软件之间的通信协议见附录C。

5.2 观测计划文件入库

RTS2自动观测能力主要来自于两个方面：

1）设备运行自动化，主要涉及到设备内部状态机转换以及设备命令队列缓存。这部分在前文中已有详细论述。

2）观测目标选择自动化。这也是LAMOST自动化观测控制非常需要借鉴的地方。

5.2.1 RTS2目标选择机制

RTS2中目标选择分为两种方式：通过客户端（例如RTS2的Monitor）直接选择单个目标、采用RTS2调度方法选择目标。前者适合于人工控制观测，后者则提供了自动化观测的基础。

RTS2的调度方法又进一步分为三种：单队列调度（queue scheduling）、派发调度（dispatch scheduling）和元队列组调度（meta-queues scheduling）(Kubánek, 2012)。

单队列调度：由观测者预先制定好一个观测目标序列提交到RTS2的plan表中，并指定序列开始和结束时间。然后当开始时间到达时，RTS2根据这个序列逐一观测目标，直到完成整个序列观测（或者到达结束时间，或者天亮了）为止。这种方式属于定制自动化观测。

派发调度：由RTS2根据价值（merit）函数计算出数据库中所有目标的价值，选最高者进行观测。这种方式属于随机自动化观测。

单队列调度更直观，但由于排序方式单一，使用不够灵活。派发调度全自动化，但每次均为纯随机选择，无法人为干预。而且，对于大型望远镜来讲，由于需要考虑的条件较多，合理的价值函数不好定义。

元队列组调度则结合了单队列调度和派发调度的优点。它允许有多个备选的先入先出（FIFO）队列，且每个队列可以有不同的排序方式。在每次自动选择时，先更新每个队列（将各队列中已过期目标、无法观测目标剔除掉），然后只用价值函数计算每个队列的队首项（第一个元素），选择其中最高者进行观测。

元队列组调度方式的核心算法思想其实就是数据分组（分级）与缓存。它保留了单队列调度的直观特点（便于理解和维护）。它因为每次选星时由于不用像派发调度那样计算所有目标的价值，从而极大的降低了算法复杂度和运算量(Kubanek, 2010)。因此，元队列组调度已成为RTS2中最常用的调度算法。该算法的实现与执行是由RTS2的selector服务模块来完成的。

目前，元队列组调度算法中支持的每个备选队列的排序方式主要有：

1）FIFO，备选队列按插入顺序待选。

2）CIRCULAR，与FIFO相似，但是观测完的目标会再次插入队尾。

3）HIGHEST，按目标高度角排序。

4）WEST-EAST，按目标由西到东排序（由于地球自转，越西侧的目标可

获得的观测时间越长，优先级越高）。其实这种方式也适合与LAMOST，但是，由于LAMOST观测范围是中天前后两小时（见图5.1），如果选择这种排序方式，需要修改RTS2的目标过滤函数filterUnobservable()。因为该函数默认过滤条件是地平线。

5）WEST-EAST-MERIDIAN，按目标是否接近中天排序，越接近中天优先级越高。

从源码分析，RTS2的各备选队列就是其数据表中的信息映射到内存变量后所组成的队列（扩展了STL的vector）。而各种排序方法也是由STL标准排序算法对各表不同字段排序而得。RTS2相关数据表见5.2.2节。

RTS2作为小型望远镜自动化控制软件，其所提供的调度算法和队列排序方式并不完全适用于LAMOST。它与SSS系统的选星功能有部分重合，但显然它所提供的选星功能本身不能满足LAMOST巡天需求。

另一方面，如果将SSS系统生成的观测计划与RTS2的观测目标进行对比，会发现两者间有诸多相似之处。如果能将观测计划映射成RTS2的观测目标，则无疑是在RTS2引入LAMOST观测控制这一课题上迈出了具有重要意义的一步。

5.2.2 OCS数据库与RTS2数据库

虽然由SSS系统生成的面向光纤定位系统的观测计划文件内容很多、信息量巨大（参见附录B），但是结构清晰，采用XML格式记录这些信息使OCS系统、光纤定位系统以及DHS系统等软件可以较为方便的进行解析、存储和转发。

OCS系统在获得当夜备选的一系列观测计划文件包后，逐一扫描包中各天区的XML格式观测计划文件，并将各种关键信息记入后台MySQL数据库中。OCS系统后台数据库中与观测计划信息相关数据表关系简图如图5.7所示。从图中可以看出，虽然各数据表内容较多，但是表结构清晰、表间关系也较好理解。

另一方面，要将RTS2软件系统完全引入LAMOST观测控制领域，也必须理解RTS2相关数据库知识。图5.8展示了RTS2系统中数据库内各数据表的结构以及表间关系。

其中，核心表名叫targets，表内包含了一条一条的观测目标记录。scripts表内记录着相对于每个观测目标人工制定设备执行脚本（设备脚本举例见5.4节）。observations表中记录了RTS2已经观测的目标日志记录。images表中记录的是各目标观测后所获得的图像的相关信息。plan表记录着观测者指定的需要观测的序列（详情参见5.2.1节）。

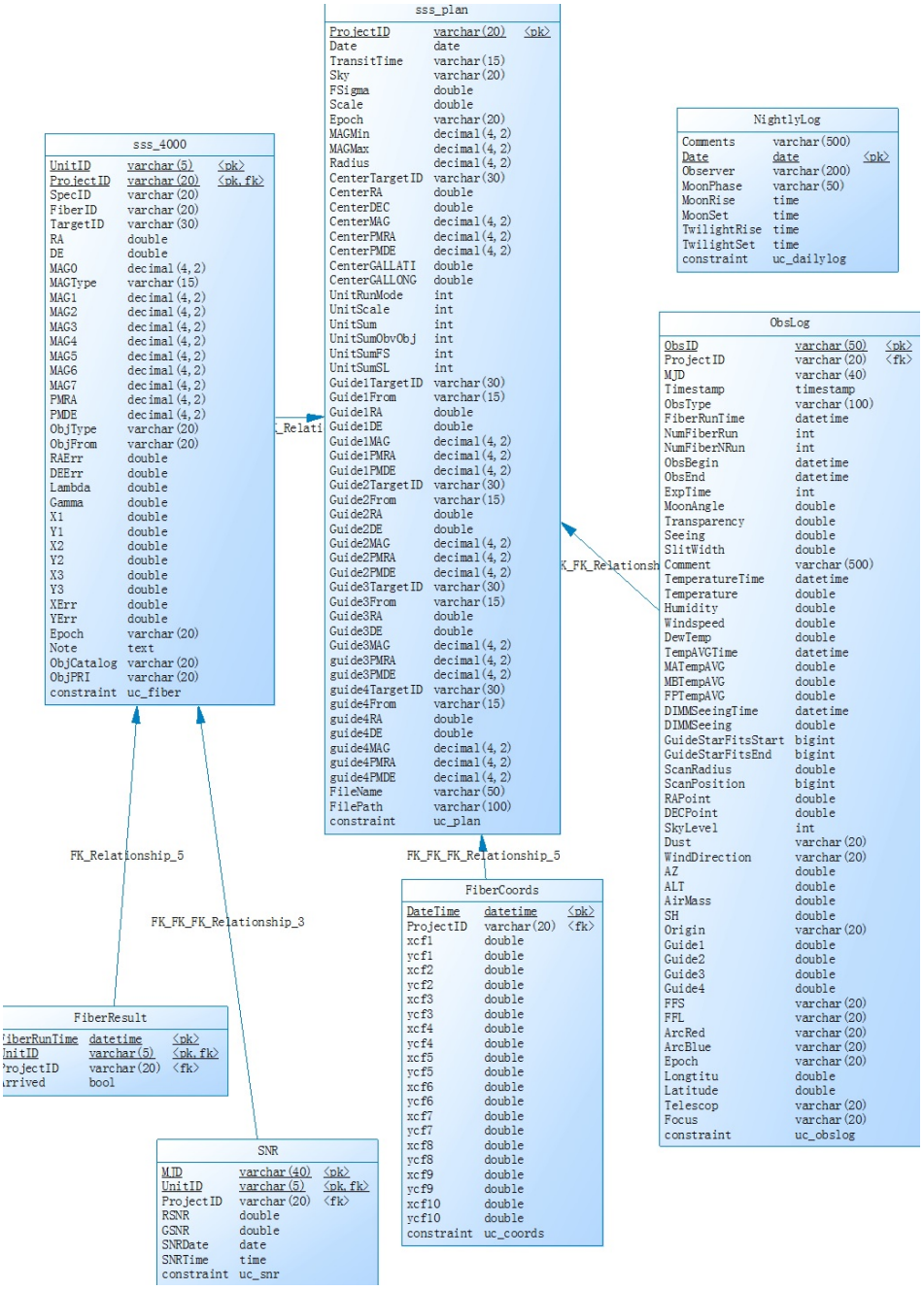

图 5.7 OCS系统数据库关系简图

将两者进行对比后发现，OCS数据库虽然比RTS2数据库复杂，但其复杂性来自于观测计划内部细节信息以及各子系统运行时细节信息。站在宏观角度分析，RTS2并不关心4000个目标单元的天文信息和运行信息。

如果仅以观测计划中的中央星作为RTS2的观测目标，则一切问题就迎刃而解了。经过认真分析两者之间的异同，结合RTS2的自动化目标选择原理以及LAMOST实际工程环境，我们得到了两者之间的逻辑对应关系。表5.1列举了部分的信息映射关系。

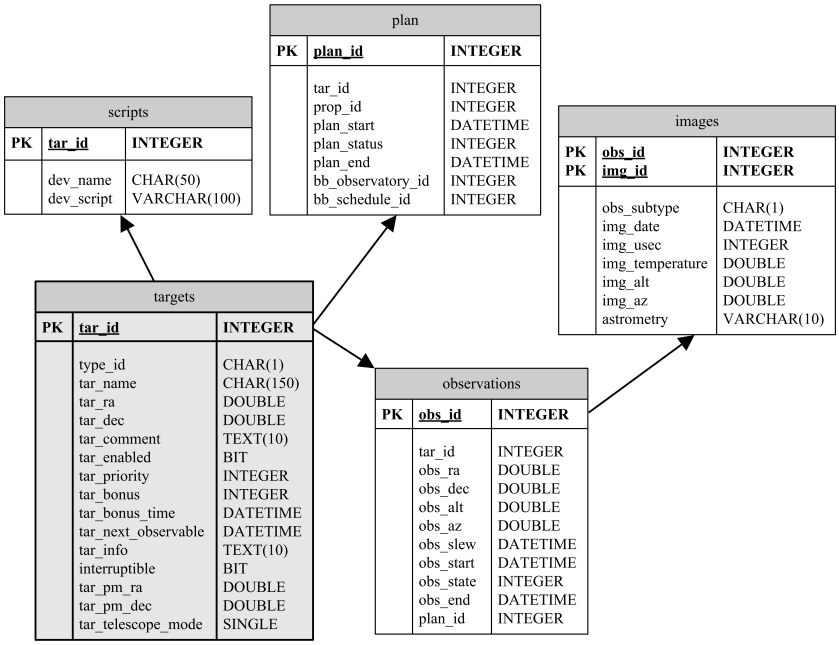

图 5.8 RTS2系统数据库中目标相关的数据表关系简图

表 5.1 LAMOST观测计划信息与RTS2的targets数据表部分对应关系

LAMOST	RTS2	说明
观测计划ID	tar_id	目标ID
观测计划名称	tar_name	目标名称
中央星赤经	tar_ra	目标赤经
中央星赤纬	tar_dec	目标赤纬
无	tar_info	用于向光纤定位传输文件
观测计划文件路径	tar_comment	用于向光纤定位传输文件
观测计划待执行日期	无	目前，只考虑当夜
天区中天时间	无	属于冗余信息，根据中央星赤经、赤纬以及观测点的三维地理位置RTS2可以计算出来
中央星星等	无	RTS2作为小型望远镜控制软件，暂时未考虑不同星等目标的直接选择，但是通过tar_bonus和tar_priority组合可以达到星等选择效果

在理清映射关系的基础上，我们需要编写程序来完成具体映射工作，程序

中还需要如下问题：

1）PostgreSQL数据库接口（RTS2采用的是PostgreSQL数据库）

2）RTS2数据库中各字段数据类型、大小与LAMOST观测计划信息的兼容性问题。

为了加快自动化模拟测试进度，我们采用Python编写程序，扫描单夜所有观测计划文件，从每个观测计划文件中提取信息，并按照逻辑对应关系逐一将观测计划信息映射到RTS2数据表中的对应项上，从而顺利的完成LAMOST的观测计划在RTS2软件系统中的入库工作了。

5.3 光纤定位软件接口的RTS2映射

如前文所述，作为世界领先的光谱巡天望远镜，LAMOST的光纤定位技术是其几大核心技术之一。在为小型望远镜自动化观测而设计的RTS2软件框架中显然没有直接对应的设备模块。

经过分析，光纤定位系统与RTS2中的滤光片设备类Rts2Filterd的操作逻辑以及与相机设备的关系极为相似，因此，我们尝试将LAMOST的光纤定位系统虚拟化后映射成RTS2的滤光片设备。

Rts2Filterd类拥有全一位全局状态位（0或1），其所能表示的状态只有两种，分别为FILTERD_MOVE（滤光片移动中，不允许曝光）和FILTERD_IDLE（滤光片已到位，允许曝光）。这正好符合LAMOST中光纤定位系统与CCD相机集群之间的操作逻辑。

Rts2Filterd类实现比较简单。由于继承自设备类Device，该类具备了以守护进程方式长时间运行的能力。在commandAuthorized()函数内部规定了其可以接收并执行（由EXEC发来的）“filter 参数”命令。

Rts2Filterd类中提供了两个最关键的虚函数：

1）int addFilter(int new_filter)，用于给自己的数据成员filters添加新的滤光片（通过继承和覆盖我们可以将观测计划虚拟成单个滤光片）

2）int setFilterNum(int new_filter)，用于在接到“filter”命令时修改当前选中滤光片信息、并可以作为子类实际驱动滤光片设备的函数使用。

在第三章和第四章虚拟化LAMOST设备时，我们遇到了长时间运行命令阻塞RTS2设备类本身循环的问题。为了解决这个问题，当时分别采用了两种方法：多进程方法、自定义Rts2连接类并将之融入到RTS2多路监听机制中的方法。

映射光纤定位接口其实要面对这个问题。但是，由于Rts2Filterd作为独立模块运行（独立进程），且只做更换滤光片这一件事，因此，这种阻塞并不影响整个RTS2软件的运行。所以，在映射光纤定位接口时我们并未采用以上的复杂方式，而是采用了更为简单的方法：直接继承Filterd类产生LAMOST-Fiber子类，子类覆盖int addFilter()函数和setFilterNum()函数。

软件工作流程如下：

1）EXEC发来“filter 123”命令，从而调用LAMOST-Fiber的setFilter(123)，参数123为上一节中介绍的targets表的tar_info项的值，该值为int型。

2）setFilter()函数内，先调用已覆盖写的addFilter(123)。addFilter()根据123查找RTS2数据库，找到tar_id、tar_name等一系列值，以此更新自身数据成员。

3）setFilter()函数，与光纤定位软件创建连接，并按照OCS与光纤定位协议（附录C）逐一发送命令IcsFiberReady、IcsFiberSelfCheck、IcsFiberPOST_RT、IcsFiberRun和IcsFiberOffline，从而完成了光纤定位驱动工作。其中，在传输观测计划命令IcsFiberPOST_RT中，需要根据2）中查到的文件路径读取解析观测计划XML文件，然后按规定格式填入命令中。

4）setFilter()函数完成并返回，LAMOST-Filterd继续执行自己的循环。

通过继承Rts2Filterd类并覆盖重写相应虚函数，我们完成了LAMOST光纤定位系统接口对RTS2系统的映射。

5.4 基于RTS2框架的LAMOST模拟控制

在完成了资源监控映射、CCD集群映射、光纤定位映射、观测计划入库等工作后，我们可以尝试建立基于RTS2系统的LAMOST观测控制环境了。

由于我们尚未完成LAMOST的TCS子系统的虚拟化工作，因此，相应模块我们采用RTS2提供的伪设备类（dummy）来代替。RTS2框架为每一种设备类都提供dummy实现，可模拟运行，这也是RTS2软件框架的优点之一。

采用Rts2Teld的dummy来代替TCS子系统后，虽然我们还不能RTS2控制望远镜指向跟踪，但是搭建出来的模拟环境已经足可以展示RTS2系统的自动控制能力了。RTS2模拟环境中各模块之间的关系如图5.9所示。

在模拟测试中，Rts2Teld的dummy模块取设备名为“T0”、LAMOST相机模块取名为“C0”、光纤定位系统取名为“F0”。资源监控系统由于并不参与观测过程，因此暂时省略。

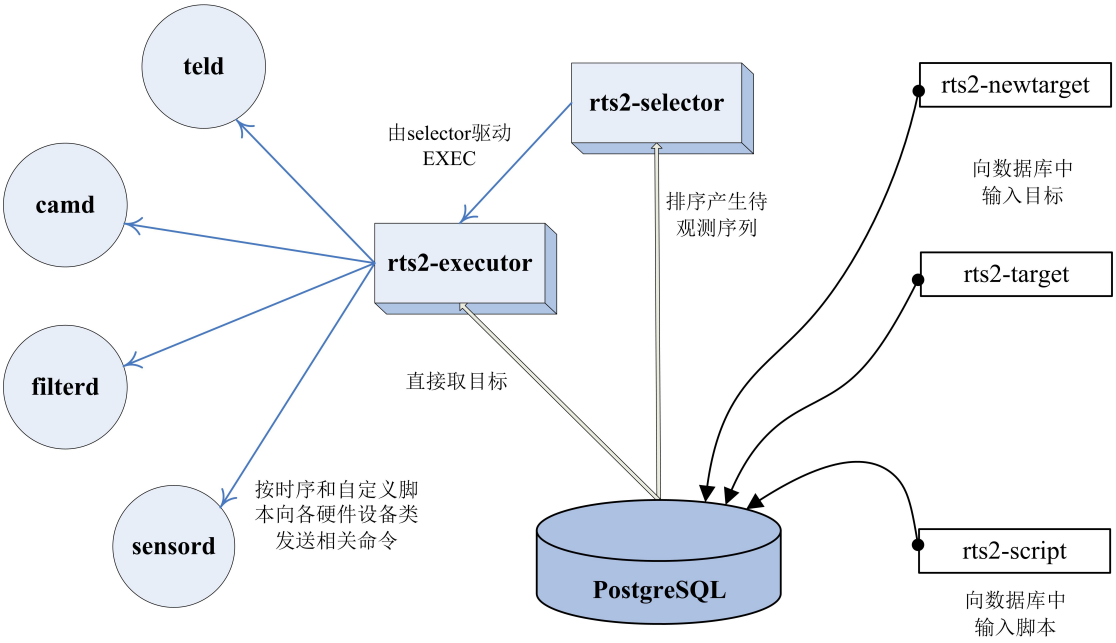

图 5.9 基于RTS2的LAMOST自动化观测模拟测试组件关系图

我们选择某夜SSS产生的全部观测计划包，用我们开发的Python程序读取这些观测计划，映射后加入RTS2数据库中。为了模拟实际观测效果，我们还针对每个观测计划（RTS2中的目标）给出相应的硬件设备脚本。

假设观测计划名为NGC123，添加设备脚本为：C0 F 123 loops 2 E 5 。即：当NGC123要进行观测时，对C0相机需要首先进行光纤定位（RTS2库中设置了NGC123对应的tar_info为123），然后开始2轮观测，每轮曝光1次时长5秒。

开始运行整个RTS2软件后，目标选择器selector会自动地逐个选择我们之前已经导入RTS2数据库中的各个观测计划观测计划，然后驱动执行器EXEC进行自动化观测。执行器EXEC和中控服务Centrald（图5.9中并未画出）会驱动和协调各个设备模块的操作，最终完成整个观测流程。

图5.10展示了自动化观测过程中T0、F0、C0三个设备模块之间以及与选择器Selector、执行器EXEC和中空服务Centrald之间的交互时序逻辑图。

从图中可以清晰的看出，当望远镜移动或滤光片切换（光纤定位运行）过程中，相机会自动被阻塞，EXEC发给相机的曝光命令会被缓存，直到条件允许后才自动执行。另一方面，当相机正在曝光过程中，望远镜不允许移动，滤光片不允许切换。

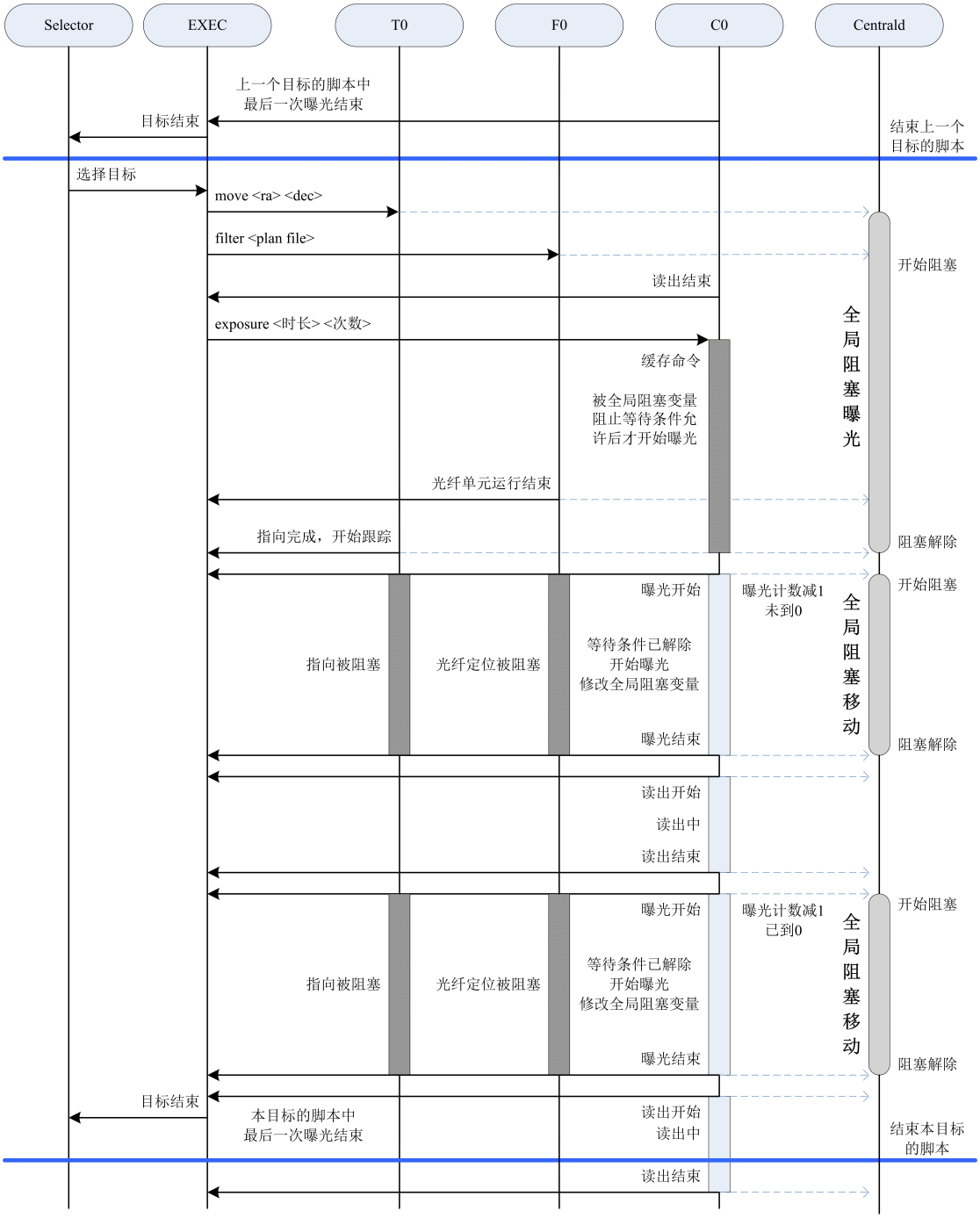

图 5.10 自动化观测模拟测试中模块交互时序图

需要注意的是，当相机读出时并不阻塞望远镜移动和滤光片切换，也就是说，当相机正在进行当前天区最后一次读出过程中，望远镜和滤光片已经可以开始进行下一个天区的准备工作了。RTS2的这种设计思路非常有意义，阻塞设计保证了各模块间操作的协调性，而并行设计有利于提高望远镜的观测效率。

图5.11是模拟测试过程中RTS2的Monitor的截图。该截图显示的是RTS2自动选择目标227，并进行3轮观测，每轮先曝光3秒，再曝光5秒。

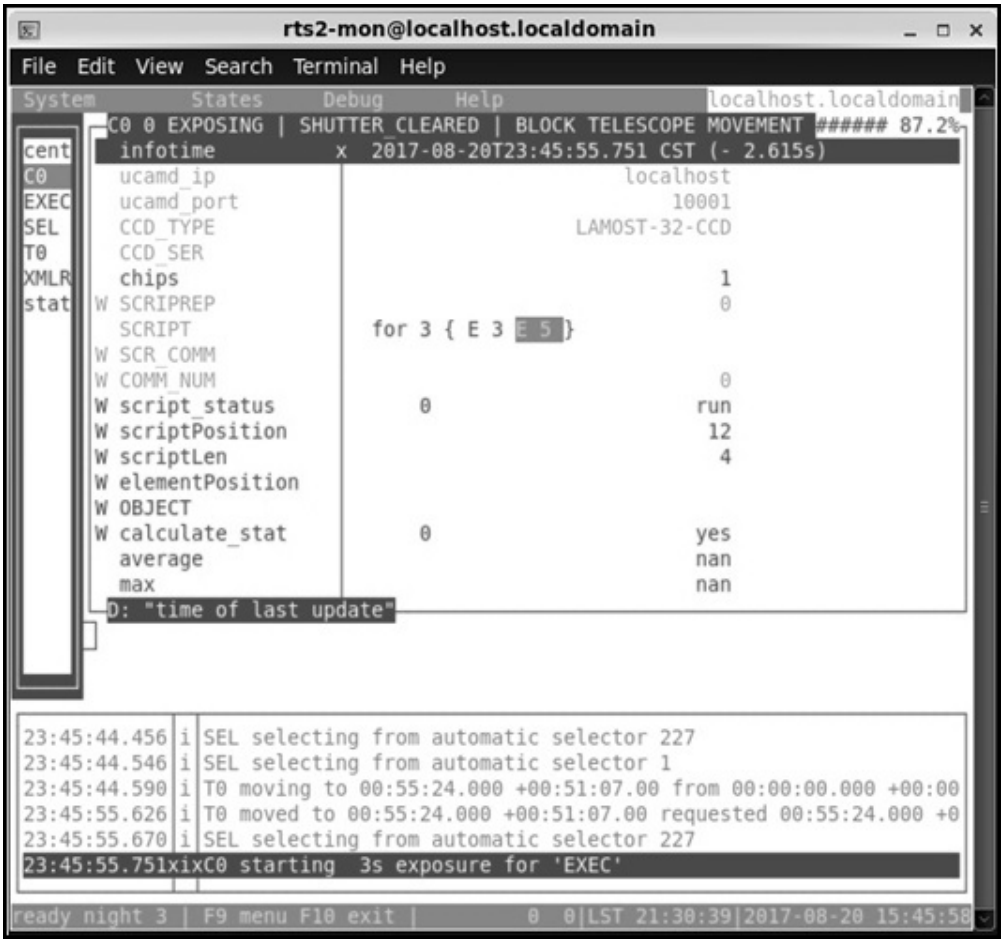

图 5.11 自动化观测模拟测试中RTS2的Monitor模块运行截图

模拟控制实验顺利完成，除了在开始测试前需要人为控制（向RTS2数据库录入观测计划以及设备脚本）之外，整个测试过程无需人工干预。

5.5 基于RTS2框架的LAMOST设备全面映射的预研

如前文2.6节所述，在大型望远镜设备虚拟化思路的指导下，我们对现有的LAMOST子系统或软件进行合理化的分类，并逐一映射到RTS2软件框架中相应的设备或服务模块上去。整体映射思路如图5.12所示。

本文中，在完成了Sensord-2、Camera、Selector（目标入库）以及Filter的虚拟化映射（图5.12中蓝色背景模块）后，搭建了基于RTS2的LAMOST自动化观测控制的测试环境，并进行了模拟测试。测试结果进一步证明了RTS2软件完全有能力控制如LAMOST的大型望远镜自动化观测。

在本文接近尾声之时，我们将对下一步工作设想进行简单描述。为全面虚拟化LAMOST子系统从而实现RTS2全自动控制展开预研。

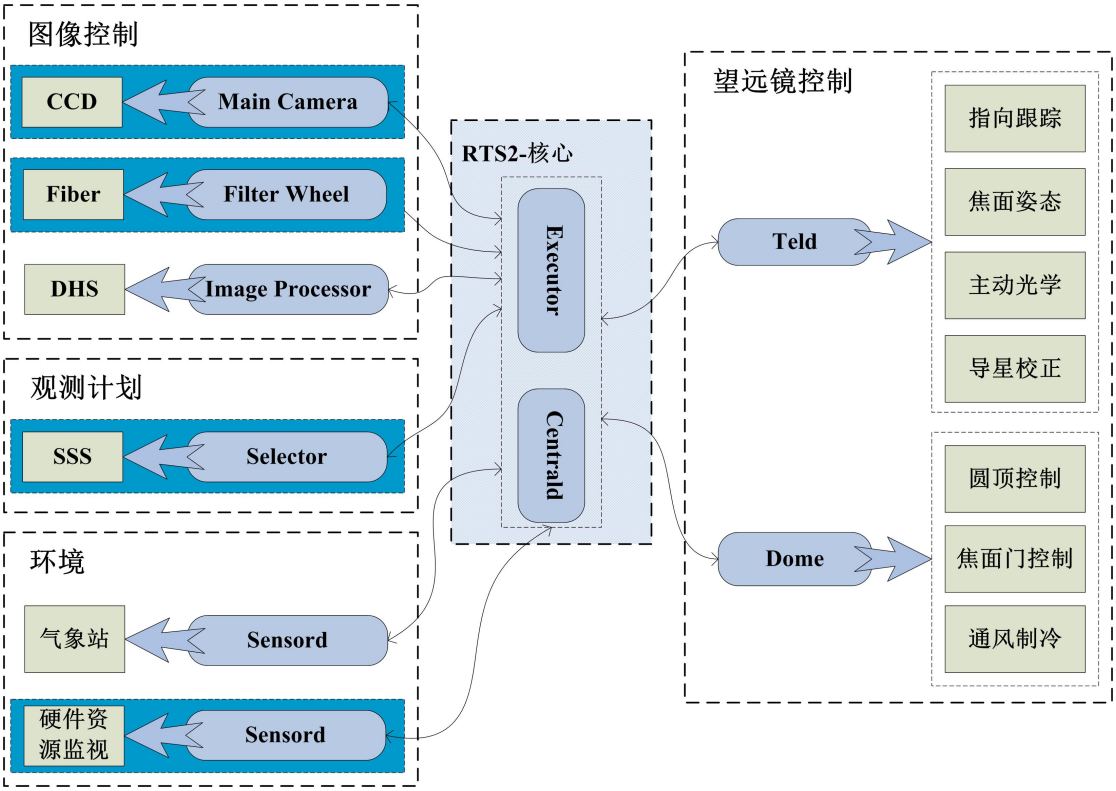

图 5.12 基于RTS2的LAMOST设备虚拟化映射总图

5.5.1 RTS2框架Dome设备模块的映射

结合表2.5和图5.12，LAMOST的TCS子系统的部分功能例如：圆顶控制、焦面门控制以及通风制冷控制将映射成RTS2中的Dome模块。在RTS2框架中，Rts2Domed类是圆顶控制模块的基类，其中提供了OpneDome()、CloseDome()等虚函数接口，同时，该类还会根据圆顶状态自动修改RTS2全局状态变量并广播给其他组件。RTS2为Rts2Domed分配的全局状态有：DOME_OPENING（圆顶正在打开）、DOME_OPEND（圆顶已打开到位）、DOME_CLOSING（圆顶正在关闭）、DOME_CLOSED（圆顶已关闭到位）等。

目前，OCS系统利用LAMOST通信协议可以控制TCS系统这部分功能，并获得状态反馈。TCS这部分功能相对于RTS2提供的功能更为复杂，开关圆顶、开关焦面门等操作之间有着时间顺序制约关系。但是对于望远镜整体控制来讲，只需要了解整套操作是否完成即可。因此，可以通过添加中间层的方法，将这部分功能的实际业务逻辑进行封装，对RTS2Dome类只提供最终运行结果。

5.5.2 RTS2框架Mount设备模块的映射

如图5.12所示，LAMOST的机架指向跟踪、焦面姿态调整、主动光学以及导星校正等功能需要合并映射成RTS2框架的Rts2Mountd模块。Rts2Mountd基类

提供了如：move()、park()、stop()等一系列函数以适应小型望远镜镜筒控制的操作函数，并且自动根据当前接收到的命令以及望远镜镜筒是否在转动等状态修改RTS2全局状态变量并广播给其他组件。RTS2Teld所分配的全局状态相对较多，最重要的是TEL_MOVING（望远镜正在指向调整中）、TEL_TRACKING（望远镜跟踪）、TEL_CORRECTING（望远镜校正）、TEL_PARKING（望远镜正在停靠中）、TEL_PARKED（望远镜停靠）等。

LAMOST设计新颖，采用的不是传统镜筒转动方式，而是设计了MA反射镜。作为一台中星仪式的望远镜，在观测过程中，LAMOST将观测天区位置（赤经、赤纬）、当前时刻以及地理位置等信息换算成MA机架方位角和俯仰角数据，利用这些数据进行望远镜指向和跟踪。MA机架实物图如图5.13（左）所示，望远镜光路的原理如图5.13（右）所示。

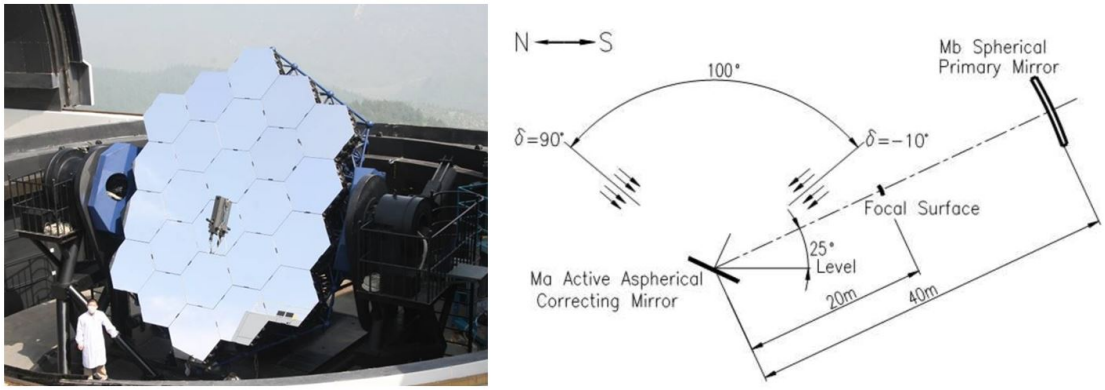

图 5.13 机架实物图（左）和机架控制光路图（右）

由于LAMOST设计的特殊性，望远镜指向、跟踪过程中涉及到一系列复杂命令状态交互过程。但经过抽象后，可以对应于RTS2的move命令和trace命令。

另一方面，在LAMOST望远镜整体完成指向操作后，需要进行主动光学面形校正以及导星调焦校正操作。

主动光学子系统通过对中央星光线进行分光和拍摄获得点阵图，然后各分光点光度强弱差异计算面形校正值，并驱动MA子镜背后众多力促动器和位移促动器来调节MA整体面形，使望远镜获得最好的聚光效果。

四台导星CCD相机拍摄较亮的引导星，根据图像进行一系列天测运算获得校正数据，对望远镜进行高度、方位、旋转和调焦操作。

图5.14（左）为导星在焦面板上的分布图，0至3号为1K×1K像素正在使用的导星CCD相机，4至7号为4K×4K的新型相机，新型相机目前正在测试不久后将会融入原有导星系统，从而为望远镜提供更精确的校正数据。

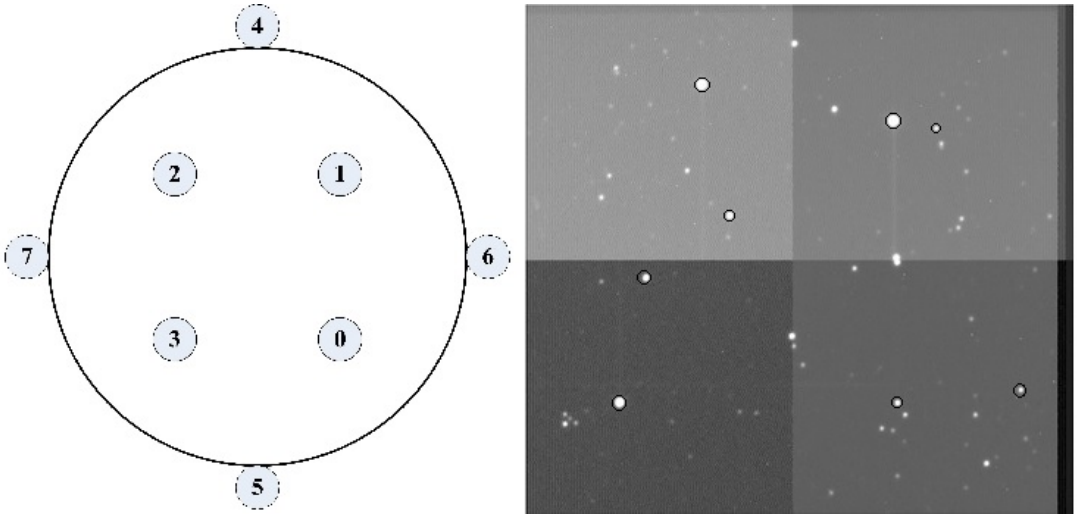

图 5.14 导星相机在焦面分布图（左）和导星天测计算界面图（右）

图5.14（右）为导星校正过程中天测运算界面截图。通过四幅图像上指定导星位置的计算结合焦面框架数据，就可以获得实时的校正信息来修正望远镜跟踪误差以及焦面失焦调节。

经过分析，总结RTS2与OCS的异同后得出初步设想：可以将主动光学校正操作和导星调焦操作合并映射成Rts2Teld的correct命令，在两者进行操作过程中，Rts2Teld处于TEL_CORRECTING状态。

5.5.3 RTS2框架气象传感器设备模块的映射

在LAMOST焦面楼楼顶西侧平台上安装着一系列室外测量设备，为了保护这些辅助测量设备还加设了矮墙和可移动顶棚，我们称整个土木建筑以及其内的测量设备为“小圆顶”。小圆顶内部设备如图5.15所示。

小圆顶内主要包括用于测量气象环境信息的气象站设备（包括：风速仪、湿度仪、粉尘仪等）、用于测量室外视宁度的DIMM、以及控制小圆顶顶棚开启和闭合的机电设备等。

目前，这些设备均采用计算机控制，通过LAMOST内网与OCS系统的WIS组件相连，为OCS提供实时气象信息、视宁度信息。小圆顶控制软件也提供了基于网络的远程控制接口。

在RTS2框架中，气象信息是通过Rts2Sensord模块采集获取的。RTS2对气象传感器设备没有定义任何命令，只提供了两种全局状态：GOOD_WEATHER（好天气，允许观测）和BAD_WEATHER（坏天气，不能观测）。

显然，LAMOST小圆顶可以映射成RTS2气象传感器设备模块，但是RTS2气

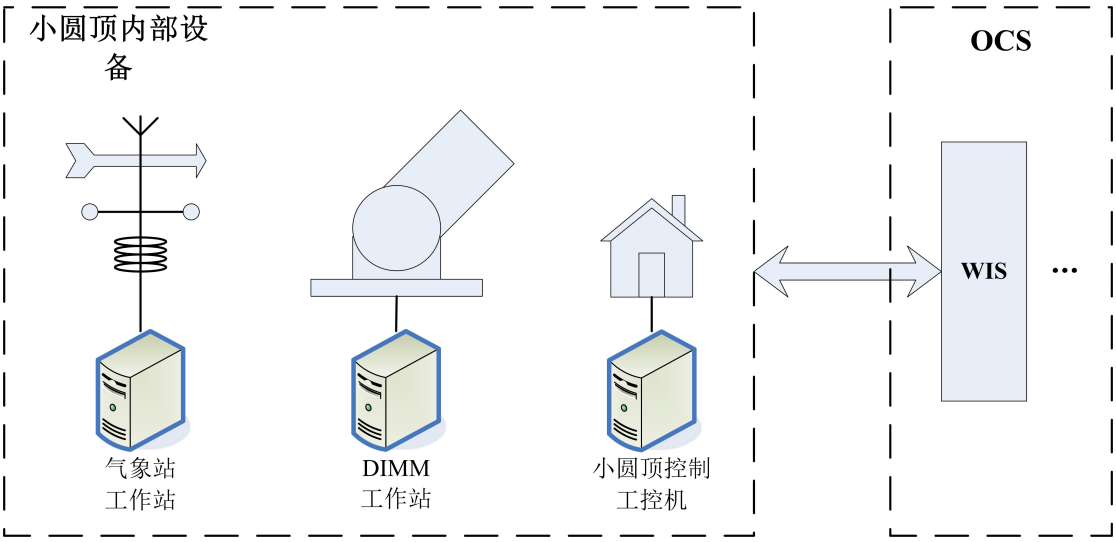

图 5.15 小圆顶内部设备示意图

象传感器模块所提供的状态过于简单，因此后期小圆顶虚拟化和映射过程中会有较大的工作量。另外，气象传感器模块不具备命令处理能力，这也是需要注意的地方。

不过，如前文所述，在深入理解了RTS2设备类运行机制的基础上，可以覆盖重写commandAuthorized()函数，扩充出属于小圆顶设备类自己专用的命令和相应处理程序。进行精心设计和细心测试，基于Rts2Sensord的LAMOST小圆顶类就可以满足映射和集成需求。

5.5.4 RTS2框架ImageProc服务模块的映射

LAMOST的DHS子系统主要功能有两个：

（1）向已拍摄完成的32图像填写FITS头信息。这些信息包括：观测计划中的天文相关信息、OCS采集的望远镜各子系统运行时信息、气象环境信息等。

（2）进行在线数据处理，根据图像快速计算出上一次观测的效果。

DHS的功能（1）为后期数据处理提供全面的原始记录信息。而功能（2）是一种反馈机制，只有包含了这种反馈机制才能使自动化控制实现闭环。

图5.16显示的是以相机为单位计算信噪比的界面截图，图5.17显示的是单次观测总体信噪比分布图。

RTS2框架中提供了图像处理服务（称为IamgeProc），人工提前写好图像处理脚本，采用Linux系统提供的文件监视接口（inotify）来实现对指定路径下文件变化的监控。当RTS2获得拍摄图像文件后，自动驱动图像处理脚本运行。

在下一步工作中，可以将DHS系统虚拟化映射成RTS2框架的IamgeProc模

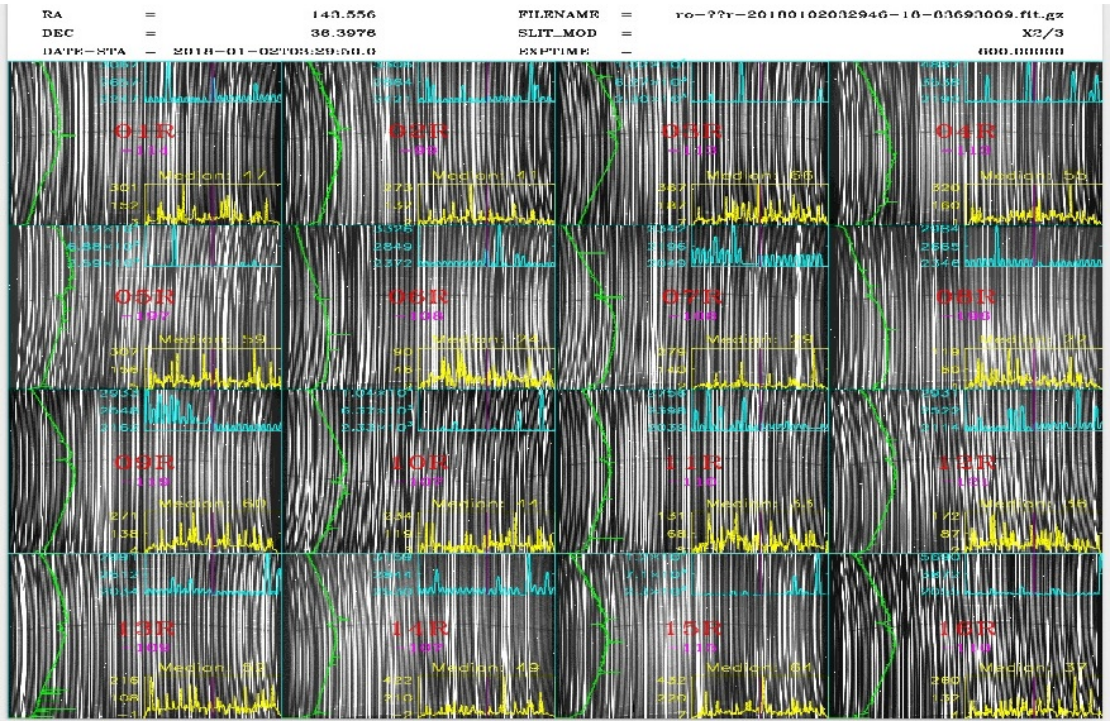

图 5.16 以CCD为单位的在线信噪比计算结果截图

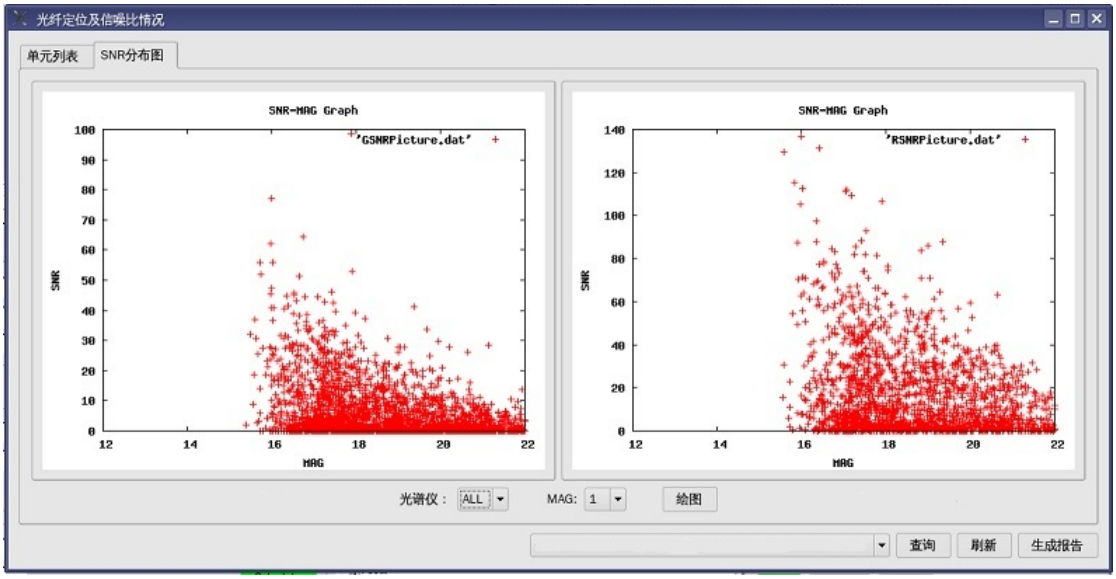

图 5.17 在线信噪比计算结果在红蓝两端的分布图

块。但是，在这个过程中，如何保证本次观测全部图像获取完成后再驱动处理脚本这一问题需要认真考虑。另外，还需要仔细设计图像处理脚本，在保证处理结果正确的前提下减少处理时间。如果脚本运行时间过长，就无法达到实时反馈的效果了。

5.5.5 RTS2框架XML-RPCd服务模块的映射

本地GUI控制与网络访问相结合是当今软件的流行趋势。作为天文望远镜，提供网络访问功能，使天文学家和天文爱好者更好的了解望远镜相关情况，这一思路对望远镜的运维和未来发展也有极大的益处。但是，作为国家大科学工程，网络开放等级、内容开放程度都应有严格而详细的规范，这种信息分级制度也是LAMOST运维的重要组成部分。现行OCS系统在设计之初更多考虑的是客户端/服务器框架，因此，只提供了本机GUI访问和控制功能，未提供远程访问功能和网站服务。

作为自动化控制软件，RTS2从设计之初就考虑了远程访问功能。RTS2框架提供的Rts2XmlRpcd模块（最新版RTS2软件中已改名为Rts2httpd）就是远程访问服务模块，如图5.18所示。最初，Rts2XmlRpcd模块只提供基于JSON格式的信息交互功能(冉凡辉 等, 2013)。经过不断的升级和扩展，以WebSocket为代表的一大批新技术和新功能已成功引入到Rts2XmlRpcd模块中(卫守林 等, 2014a)，为RTS2提供了更加完善和强大的WEB功能。

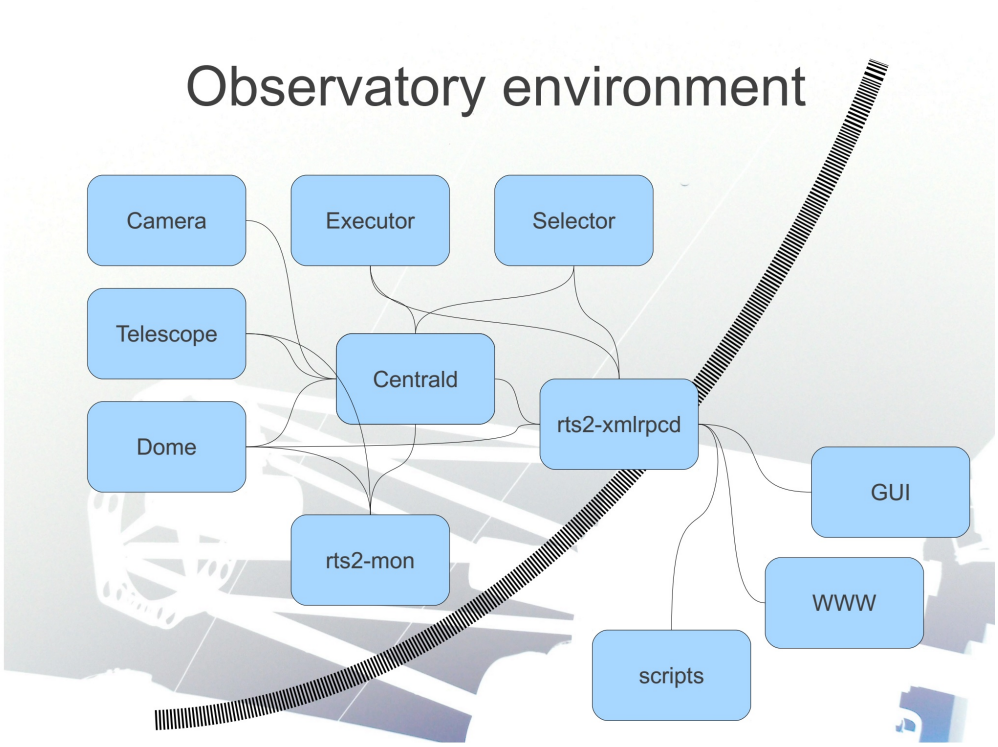

图 5.18 基于RTS2的LAMOST自动化观测模拟测试组件关系图

需要注意的是，Rts2XmlRpcd模块本身的用户身份验证功能较薄弱，采用的是简单的用户名密码登录机制。而且，目前实现中信息并未分级、命令与状态也未分离，因此，一旦用户登录成功，就拥有了整个望远镜的控制权。对

于LAMOST望远镜，这种情况显然是不能被允许的。

因此，在下一步工作中，如何在保证望远镜绝对安全的前提下，适度开放信息成为一项需要研究的子课题。对现有OCS信息分级管理功能的保留，以及对Rts2XmlRpcd内部的改造和扩展需要我们进行大量工作。

通过本节的预研与描述，RTS2全面控制LAMOST动化观测的蓝图已清晰的呈现在我们面前。

虽然，将LAMOST已经稳定运行十年的OCS系统向着RTS2方向进行升级和扩展十分困难，这个过程中所涉及到的知识和技术也十分庞杂。但是，我们坚信在大型望远镜设备虚拟化思路的指导下，随着研究人员的不断增加，研究力度的不断增大，研究经验的不断积累，本文所未完成的工作将很快得到完善，并最终能够实现LAMOST观测控制的全面自动化。

第6章　总结与展望

6.1　本文的内容总结

LAMOST是我国自主研发的具有世界先进水平的新一代大型望远镜。由于大口径与大视场的兼备，使之成为光谱获取效率最高的望远镜。LAMOST是一个由多个子系统组成的复杂系统，加之观测行为的复杂性，决定了它必须依靠观测控制系统（OCS）来完成望远镜观测控制。现行OCS系统已稳定运行十年，为LAMOST巡天做出了重要的保证。随着光谱巡天计划的不断展开，对于OCS自动化控制的需求也越来越迫切。对现有OCS系统的改造和升级已刻不容缓。本文正是围绕这一课题进行了多方面的研究和探讨。

本文深入分析了OCS系统的分层式软件体系结构、组件设计原理、系统运行机制以及所采用的关键技术等问题，总结了现行OCS系统的优点和不足。

在广泛调研国内外望远镜观测控制系统软件的基础上，选择了具有极强自动化控制能力的RTS2软件系统作为OCS升级的方向。在深入理解RTS2软件系统设计原理和运行机制的基础上，与现行OCS系统进行对比，充分思考两者之间的异同，最终提出了基于LAMOST的大型望远镜设备虚拟化概念。

在基于LAMOST的大型望远镜设备虚拟化思路的指导下，分别完成了基于Python异步协程技术的LAMOST控制节点分布状态采集与监视系统软件的开发和虚拟化，现有多相机集群控制软件的重构和虚拟化，LAMOST观测计划文件的RTS2入库以及对OCS现有光纤定位系统接口的虚拟化。并将以上软件分别映射到RTS2软件体系中的不同设备(或服务)模块上。

搭建测试环境，利用RTS2提供的伪设备模块技术模拟LAMOST的TCS系统，结合已完成的映射设备，进行基于RTS2的LAMOST自动化观测模拟测试。测试结果证明了将RTS2引入LAMOST的可行性，并为完成“基于RTS2的LAMOST自动化观测运行”这一课题打下了坚实的理论基础，提供了参考方法。

6.2　本文的内容总结

大型望远镜自动化控制是一个复杂而又巨大的系统工程。现行OCS系统也是经过多方共同努力历时近十年才基本完成的。本文对OCS系统的升级和改造进行了开拓性的研究，但仍有大量工作需要实现和进一步完善：

1）为了尽快完成基于RTS2的LAMOST自动化观测实验，文中第五章对观

测计划信息的RTS2入库以及对光纤定位接口的映射工作均进行了简化处理。无论是在异常处理机制方面，还是在复杂情况分析方面都远未达到工程使用的要求，今后这方面的工作需要进一步完善。

2）LAMOST的TCS子系统以及GSS子系统尚未进行虚拟化。因此，暂时还无法实现用RTS2来进行完全控制LAMOST望远镜。在文中我们已经提出了实现这一虚拟化的大体思路，但由于子系统的复杂性，想要实现完美的映射和兼容并不简单，仍有大量的工作需要完成。

3）LAMOST的DHS子系统也尚未完成RTS2映射。DHS子系统本身提供了在线快速图像处理功能。这一功能是自动化闭环控制中不可或缺的反馈环节。将DHS映射为RTS2数据处理服务模块的过程中，需要对RTS2内部逻辑进行完善和扩展。

4）如文中所述，LAMOST小圆顶设备的接入、RTS2的网络服务模块的改造仍然需要进行大量工作。

虽然后期工作量巨大困难多多，但是，在大型望远镜设备虚拟化思路的指导下，随着研究人员的不断增加，研究力度的不断增大，研究经验的不断积累，我们相信本文所未完成的工作将很快得到完善，并最终实现LAMOST观测控制的全面自动化。使观测控制系统为巡天计划的进一步实施提供保障，并为提高LAMOST测运行效率做出重要贡献。

附录 A　OCS命令、状态与执行反馈

基本命令XML格式表达如下所示：

```
<?xml version=’1.0’ encoding=’UTF-8’?>
<Command>
    <CUID>2.7.0.2.1.20180409.201812.1</CUID>
    <Parameter><DomPos>85</DomPos></Parameter>
    <PRI>Normal</PRI>
    <Sync>0</Sync>
</Command>
```

一般状态XML格式表达如下所示：

```
<Status>
    <RelatedCmdCUID>2.7.0.2.1.20180409.201812.1</RelatedCmdCUID>
    <StatusCUID>2.7.0.2.2.20180409.201814</StatusCUID>
    <Severity>Info</Severity>
    <StatusAttribute><value>65.234</value></StatusAttribute>
    <bExeStatus>false</bExeStatus>
</Status>
```

命令执行反馈的XML格式表达如下所示：

```
<Status>
    <RelatedCmdCUID>2.7.0.2.1.20180409.201812.1</RelatedCmdCUID>
    <StatusCUID>2.7.0.2.2.20180409.201812</StatusCUID>
    <Severity>Info</Severity>
    <StatusAttribute><value>ACTIVE</value></StatusAttribute>
    <bExeStatus>true</bExeStatus>
</Status>
```

一般状态的“StatusAttribute”节点内以子节点方式携带不同的信息，且“bExeStatus”节点值为false。而命令执行反馈的“StatusAttribute”节点的值始终为START、ACTIVE、DONE、ERROR之一，而“bExeStatus”节点值为true。

命令流则是由各种基本命令按照串并联方式组合成的树状结构，其XML格式定义如下：

1) 根元素标识为“CommandPipeLine”。

2) 标识为“SerialCommand”的节点表示其下子节点均为串行执行。

3) 表示为“ParallelCommand”的节点表示下子节点均为并行执行。

4) 串并行节点可以相互嵌套。

5) 叶节点必须是基本命令。

附录 B　LAMOST观测计划文件

观测计划文件包是LAMOST进行观测的依据，由SSS子系统生成，其内包含多种格式且面向不同子系统的文件。例如：面向光纤定位软件和DHS子系统的XML文件、面向机架指向跟踪的TXT文件，面向导星子系统的TXT文件等。

由于本文只涉及到光纤定位接口，因此，只对面向光纤定位的XML格式的观测计划加以说明。

面向光纤定位的观测计划文件采用ASCII字符记录信息，采用XML格式树状结构组织信息，根节点为“ObservationProject”，根节点的属性“ProjectID”的值为观测文件唯一标识，观测文件总结构为：

<ObservationProject ProjectID="xxxx" version="v2.63">

<!– 观测计划总体信息 –>

<!– 中央星信息 –>

<!– 各引导星信息 –>

<!– 目标总体分配信息 –>

<!– 各目标信息 –>

</ObservationProject>

其中，观测计划总体信息主要记录了整个观测计划的天文观测相关信息，其各节点为：

<Date>20180101</Date> <!– 待观测日期 –>

<SKY>HIP10462</SKY> <!– 天区名 –>

<AlternateNo>1</AlternateNo> <!– 同一天区内分配方案序号 –>

<PLANID>hip1046201</PLANID> <!– 观测计划名–>

<Time>114203.9</Time> <!– 中天UTC时间 –>

<F_SIGMA>-0.0223592000</F_SIGMA> <!– 焦面转角 –>

<SCALE>1.0000000000</SCALE> <!– 焦面缩放因子 –>

<EPOCH>J2000.0</EPOCH> <!– 历元 –>

<MAGMIN>0.0</MAGMIN> <!– 最亮星等 –>

<MAGMAX>100.0</MAGMAX> <!– 最暗星等 –>

<RADIUS>2.5</RADIUS> <!– 视场半径 –>

... ...

中央星信息记录当前观测计划的中央星相关信息，除了用于LAMOST主动光学校正外，也作为整个观测计划所对应天区的标识。其节点结构为：

<CentralLocation> <!- 中央星子节点 ->

<TargetID>HIP10462</TargetID> <!– 中央星ID –>

<RA>33.7188157000</RA> <!– 中央星赤经 –>

<DE>48.7823879000</DE> <!– 中央星赤纬 –>

<MAG>6.73</MAG> <!– 中央星星等 –>

<PMRA>4.20</PMRA> <!– 赤经岁动 –>

<PMDE>-16.86</PMDE> <!– 赤纬岁动 –>

<GALLONG>136.8017722902</GALLONG> <!– 焦面投影经度坐标 –>

<GALLATI>-11.8293261767</GALLATI> <!– 焦面投影纬度坐标 –>

</CentralLocation>

目前LAMOST观测中使用4颗引导星，为了提升引星校正精度正在研究和测试8颗引导星，相信未来8颗引导星成功引入LAMOST观测后会极大的提升望远镜跟踪精度。观测计划文件中逐一列出各引导星信息，其中单个引导星相关信息节点结构为：

<Guide CCDID="1"> <!- 1号引导星子节点 ->

<TargetID>N330233100000021639</TargetID> <!– 引导星ID –>

<FROM>star</FROM> <!– 引导星类型 –>

<RA>31.9870823000</RA> <!– 引导星赤经 –>

<DE>49.9264530000</DE> <!– 引导星赤纬 –>

<MAG>15.90</MAG> <!– 引导星星等 –>

<PMRA>-0.59</PMRA> <!– 赤经岁动 –>

<PMDE>4.76</PMDE> <!– 赤纬岁动 –>

</Guide>

目标总体分配信息节点主要描述了本观测计划中对4000个光纤单元所要定位目标的总体信息，其结构为：

<UNIT_SUM>3410</UNIT_SUM> <!– 分配单元总数 –>

<UNIT_SUMOBVOBJ>2924</UNIT_SUM_OBVOBJ> <!– 分配目标数 –>

<UNIT_SUM_SL>406</UNIT_SUM_SL> <!– 分配天光背景数 –>

<UNIT_SUM_FS>80</UNIT_SUM_FS> <!– 分配流量标准星数 –>

<INTILE_OBJ>7650</INTILE_OBJ> <!– 视场内可观测的目标数 –>

<INTILE_SKY>4743</INTILE_SKY> <!– 视场内可分配的天光背景数 –>

<INTILE_STD>1588</INTILE_STD> <!– 视场内可分配的流量标准星数 –>

... ...

目标信息包含了单个光纤单元所要定位的天体目标的信息，SSS系统制定观测计划时会尽量分配满所有4000个光纤单元。但也会受到多种条件的制约，例如：整体考虑LAMOST巡天计划、平衡整个观测计划星等、光纤单元损坏信息反馈等。单个目标信息节点结构为：

<ObvObj>

<UNIT_ID>F0311</UNIT_ID> <!– 目标对应的光纤定位单元编号 –>

<PLATE_ID>1</PLATE_ID> <!– 目标对应的光谱仪号 –>

<FIBER_ID>1</FIBER_ID> <!– 目标对应的光纤编号 –>

<TargetID>UCAC4_683-013088</TargetID> <!– 目标ID –>

<RA>33.7772386000</RA> <!– 目标赤经 –>

<DE>46.4180934000</DE> <!– 目标赤纬 –>

<MAG0>12.63</MAG0> <!– 目标MAG0波段星等 –>

<MAGTYPE>JHKVGRI</MAGTYPE> <!– 目标MAG类型 –>

... ... <!– MAG1至MAG7星 –>

<PMRA>0.00</PMRA> <!–赤经岁动–>

<PMDE>0.00</PMDE> <!–赤纬岁动–>

<OBJTYPE>STAR</OBJTYPE> <!– 目标类型 –>

<FROM>LAMOST</FROM> <!– 目标所属 –>

<LAMBDA>180.9091227000</LAMBDA> <!– 目标焦面投影坐标一 –>

<GAMMA>2.3640526000</GAMMA> <!– 目标焦面投影坐标二 –>

<OBJ_CATALOG>UCAC4</OBJ_CATALOG> <!– 目标来源 –>

<OBJ_PRI>10</OBJ_PRI> <!– 目标优先级 –>

<EPOCH>J2000.0</EPOCH> <!– 目标历元 –>

... ...

</ObvObj>

SSS系统分配的目标主要分为三种：观测目标、流量标准星、天光背景。三种目标均采用上述统一格式记录信息，区别主要在于“OBJTYPE”子节点的值，分别定义为：STAR、FS、SL。另外，天光背景目标的“MAGTYPE”值设为“XXX”，各波段MAG星等均设为99。

XML格式的观测计划文件主要面向光纤定位系统和DHS系统，其中光纤定位系统主要关注于各目标在焦面上的投影坐标，从而转换计算并确定每个光纤定位单元的中心轴和偏心轴需要转动的角度以及电机步进数。DHS系统则需要将观测计划中的各种信息结合OCS运行时提供的各种子系统信息记录到最终观测图像文件FITS头中。

附录 C　OCS与光纤定位接口

光纤定位软件与OCS系统之间采用LAMOST通信协议进行交互。通过多次协商并反复测试后确定命令内容。下面列举几种观测过程中常见的命令。

1）光纤定位预备

表 C.1 光纤定位预备命令

属性	内容
命令ID	3.1.1.1.1
英文名称	IcsFiberReady
中文名称	光纤定位系统预备
描述	光纤定位系统准备好，返回0后系统处于可接收命令状态，OCS获得光纤定位系统控制权
输入参数	无
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等 2、RV=0：准备完成 RV>0：准备失败，不同值表示不同原因
优先级	正常
同步/异步	同步/异步

2）光纤定位观测计划传输

表 C.2 光纤定位观测计划传输命令

属性	内容
命令ID	3.1.2.5.1
英文名称	IcsFiberPOSI_TR
中文名称	光纤定位坐标传输
描述	OCS系统以XML格式向光纤定位系统发送本次观测所需的观测计划文件

接下页

接上页

属性	内容
输入参数	<Data><![CDATA[xxxxxxx]]></Data>
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等
	2、RV=0：运转完成
	RV>0：传输失败，不同值表示不同原因
优先级	正常
同步/异步	同步/异步

每次换天区重走光纤时，由OCS系统通过此命令向光纤定位系统发送需要运行的观测计划文件。OCS发送的XML格式命令(前文已述)保持不变，只是需要将整个XML观测计划文件放入到命令的“Parameter”节点下的“localfile”子节点中。由于观测计划文件本身也是XML格式，为了防止出现转义错误，采用了XML的字符数据(Character Data，CDATA)段区来保存观测计划文件。命令结构举例如下：

```
<Command>
    <CUID>3.1.2.5.1.20180101.000000.5</CUID>
    <Parameter>
        <localfile>
            <![[CDATA[
                ... ... <!– 观测计划文件原始内容 –>
            ]]>
        <localfile>
    </Parameter>
    <PRI>Normal</PRI>
    <Sync>false</Sync>
</Command>
```

3）光纤定位运行

表 C.3 光纤定位运行命令

属性	内容
命令ID	3.1.2.1.1
英文名称	IcsFiberRun
中文名称	光纤定位运转
描述	在光纤定位已接收并缓存OCS发来的观测计划的前提下，光纤定位运行
输入参数	1、bZero：是否运行开始时先进行单元回零运作 0表示不回零 1表示回零 2、Percent：允许运行的比例，值为1至99
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等 2、RV=0：运转完成 RV>0：准备失败，不同值表示不同原因，其中 RV=1表示可运转目标不足 RV=2表示没有观测计划 RV=3表示没有Percent参数
优先级	正常
同步/异步	同步/异步

光纤定位软件根据自身配置文件的记录和防碰撞算法剔除部分单元后，剩余可运行单元数与观测计划分配的总单元数的比值称为运行比例。当运行比例大于Percent值时才可运行。当运行比例小于Percent值时，光纤定位软件视本次操作为运作错误并反馈给OCS系统。

4）光纤定位离线命令

表 C.4 光纤定位离线命令

属性	内容
命令ID	3.1.1.5.1
英文名称	IcsFiberOffline

接下页

接上页

属性	内容
中文名称	光纤定位离线
描述	OCS主动放弃控制器，光纤定位系统接收到此命令后不再响应的任何OCS命令(除IcsFiberReady以外)
输入参数	无
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等 2、RV=0：离线完成 RV>0：准备失败，不同值表示不同原因
优先级	正常
同步/异步	同步/异步

以上命令为正常巡天观测过程中常用命令，下面列举几种非观测过程中常见的命令。这些命令涉及到整夜观测开始前后操作或紧急情况操作。

1）光纤定位自检

表 C.5 光纤定位自检命令

属性	内容
命令ID	3.1.1.3.1
英文名称	IcsFiberSelfCheck
中文名称	光纤定位自检
描述	光纤定位系统进行内部CAN通信总线以及单元通信等自检
输入参数	无
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等 2、RV=0：自检完成 RV>0：自检失败，不同值表示不同原因
优先级	正常
同步/异步	同步/异步

2）光纤定位停止

表 C.6 光纤定位停止命令

属性	内容
命令ID	3.1.2.4.1
英文名称	IcsFibeStop
中文名称	光纤定位停止
描述	紧急停止正在运行的光纤定位
输入参数	1、无参数，停止所有正在运行的光纤定位单元
	2、FiberID，停止指定的正在运行的光纤定位单元
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等
	2、RV=0：该单元(或所有单元)已经运行到位
	RV=1：该单元(或所有单元)已经紧急停止
	RV>0：准备失败，不同值表示不同原因
优先级	高
同步/异步	同步/异步

3）光纤定位回零

表 C.7 光纤定位回零命令

属性	内容
命令ID	3.1.2.3.1
英文名称	IcsFiberBackZero
中文名称	光纤定位回零
描述	光纤定位回零
输入参数	1、无参数，停止所有正在运行的光纤定位单元
	2、FiberID，停止指定的正在运行的光纤定位单元
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等
	2、RV=0：该单元(或所有单元)回零运转成功
	RV>0：回零失败，不同值表示不同原因
优先级	正常
同步/异步	同步/异步

4）光纤定位上电

表 C.8 光纤定位上电命令

属性	内容
命令ID	3.1.2.6.1
英文名称	IcsFiberPowerOn
中文名称	光纤定位上电
描述	光纤定位控制机柜接通电源
输入参数	无
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等 2、RV=0：上电完成 RV>0：上电失败，不同值表示不同原因
优先级	高
同步/异步	同步/异步

5）光纤定位断电

表 C.9 光纤定位断电命令

属性	内容
命令ID	3.1.2.7.1
英文名称	IcsFiberPowerOff
中文名称	光纤定位断电
描述	光纤定位控制机柜断开电源
输入参数	无
输出参数	1、执行反馈：<value="xxx">, xxx为Start、Active、Done等 2、RV=0：断电完成 RV>0：断电失败，不同值表示不同原因
优先级	高
同步/异步	同步/异步

参考文献

MARTINFOWLER, 2015. 重构：改善既有代码的设计(Refactoring:improving the design of existing code).第2版[M]. [北京]: 人民邮电出版社.

OMG, 孟洛明, 韦乐平, 等, 2002. Corba服务[M]. [北京]: 电子工业出版社.

W.RICHARDSTEVENS, BILLFENNER, ANDREWM.RUDOFF, 2006. Unix网络编程.第1卷,套接口api[M]. [北京]: 清华大学出版社.

任间, 金革, 王坚, 等, 2007. Lamost选星关键约束条件分析与建模[J]. 中国科学技术大学学报, 37(6): 657–661.

冉凡辉, 邓辉, 梁波, 等, 2013. 基于xml-rpc的rts2自主观测系统远程访问技术[J]. 天文研究与技术, 10(4): 372–377.

刘志刚, 谢志林, 江晖, 等, 2011. 基于qt的lamost光纤定位软件界面设计及开发[J]. 中国科学技术大学学报, 41(1): 50–55.

卫守林, 曹子皇, 王锋, 等, 2014a. 基于websocket的rts2web控制研究[J]. 天文研究与技术, 11(4): 404–409.

卫守林, 陈亚杰, 梁波, 等, 2014b. Rts2中新ccd类型扩展方法[J]. 天文研究与技术, 11(3): 000281–286.

姚仰光, 2008. Lamost观测控制系统的建立与测试[D]. [合肥]: 中国科学技术大学.

姚仰光, 王坚, 黄鲲, 等, 2007. Ocs命令流解析器的设计与实现[J]. 中国科学技术大学学报, 37(6): 675–679.

姚仰光, 王坚, 刘光曹, 等, 2008. 基于分布式环境的lamost控制系统通信机制的研究[J]. 核电子学与探测技术, 28(2): 226–229.

崔向群, 2001. Lamost项目及进展[J]. 天文学进展, 19(2): 123–128.

崔辰州, 李建, 蔡栩, 等, 2013. 程控自主天文台网络的发展[J]. 天文学进展, 31(2): 141–159.

张光宇, 刘佳靖, 唐鹏毅, 等, 2015. 基于rts2和ep ics的成像控制软件的设计[J]. 天文研究与技术(3): 342–348.

朱利平, 王坚, 黄鲲, 等, 2007. Ocs命令调度的设计与实现[J]. 中国科学技术大学学报, 37(6): 670–674.

李建, 崔辰州, 赵永恒, 等, 2013. 基于rts2的rao管理系统二次开发[J]. 天文研究与技术, 10(3): 264–272.

梁波, 袁智, 邓辉, 等, 2015. 一种基于异构操作系统的 rts2 ccd相机扩展方法[J]. 天文研究与技术, 12(4): 466–472.

王坚, 2003. 大天区面积多目标光纤光谱天文望远镜(lamost)观测控制系统的研究和设计[D]. [合肥]: 中国科学技术大学.

王坚, 金革, 黄鲲, 等, 2005. Lamost观测控制系统体系结构设计[J]. 天文研究与技术:国家天文台台刊, 2(2): 114–121.

王坚, 金革, 黄鲲, 等, 2006. Lamost观测控制系统消息总线的设计和实现[J]. 核电子学与探测技术, 26(3): 268–271.

罗阿理, 田园, 宋静, 等, 2012. Lamost观测控制系统设计与实现[J]. 科研信息化技术与应用, 3(4): 76–85.

苏定强, 2008. 望远镜和天文学 400年的回顾与展望[J]. 物理, 37(12): 836–843.

苏定强, 崔向群, 2001. 光学/红外望远镜和技术的进展[J]. 天文学进展, 19(2): 121–122.

董健, 王坚, 邓小超, 等, 2009. 基于udp协议的ccd数据可靠传输的研究和实现[J]. 核电子学与探测技术, 29(2): 369–372.

袁海龙, 2011. Sss巡天星表系统设计和光纤分配算法优化[D]. [合肥]: 中国科学技术大学.

褚耀泉, 2007. Lamost科学观测计划[J]. 中国科学技术大学学报, 37(6): 591–595.

赵哲, 谭海波, 赵赫, 等, 2018. 基于zabbix的网络监控系统[J]. 计算机技术与发展(1): 144–149.

赵文瑞, 卢志刚, 姜政伟, 等, 2014. 网络安全综合监控系统的设计与实现[J]. 核电子学与探测技术(4): 419–424.

赵永恒, 2000. Lamost观测控制系统[J]. 天体物理学报(B12): 89–95.

赵永恒, 2012. 天文望远镜的自动观测技术[J]. 科研信息化技术与应用, 3(4): 11–16.

赵永恒, 2015. Lamost天体光谱巡天[J]. 物理(4): 205–212.

邓小超, 王坚, 董健, 等, 2011. Lamost ccd相机总控的设计与实现[J]. 天文研究与技术, 08(3): 279–284.

邢晓正, 胡红专, 褚家如, 2007. Lamost光纤定位技术[J]. 中国科学技术大学学报, 37(6): 596–600.

郭晓慧, 李润知, 张茜, 等, 2013. 基于zabbix的分布式服务器监控应用研究[J]. 通信学报(s2): 94–98.

陆文周, 2015. Qt 5开发及实例[M]. [北京]: 电子工业出版社.

黄鲲, 王坚, 姚仰光, 等, 2007. Ocs消息总线的测试与分析[J]. 中国科学技术大学学报, 37(6): 662–669.

AXELROD T A, DE M D, La Peña, 2004. An overview of the large binocular telescope control system[J]. Astronomical Data Analysis Software & Systems XIII, 314: 704.

CASTROTIRADO A J, 2014. Robotic autonomous observatories: A historical perspective[J]. Advances in Astronomy, 2010(1687-7969): 8.

DALY P N, SCHUMACHER G, DELGADO F, et al., 2016. Lsst ocs status and plans[C]//Software and Cyberinfrastructure for Astronomy IV. [S.l.: s.n.]: 99132O.

DENG X, WANG J, DONG J, et al., 2010. A control system for lamost ccd cameras[C]//Software and Cyberinfrastructure for Astronomy. [S.l.: s.n.]: 774036–774036–6.

GILLIES K, WALKER S, 2007. The design of the gemini observatory control system[Z]. [S.l.: s.n.].

HENNING M, VINOSKI S, 1999. Advanced corba programming with c++[M]. [S.l.]: Addison-Wesley Longman Publishing Co., Inc.

HOU Y, ZHU Y, HU Z, et al., 2010. Performance and sensitivity of low-resolution spectrographs for lamost[J]. Proc Spie, 7735(1): 54–58.

JESCHKE E, BON B, INAGAKI T, et al., 2008. A framework for the subaru telescope observation control system based on the command design pattern[C]//Advanced Software and Control for Astronomy II. [S.l.: s.n.]: 1563–70.

KUBANEK P, 2010. Genetic algorithm for robotic telescope scheduling[J]. Physics.

KUBáNEK P, 2008. Rts2: Lessons learned from a widely distributed telescope network[J]. Astronomische Nachrichten, 329(3): 271–274.

KUBáNEK P, 2010. Rts2 - the remote telescope system[J]. Advances in Astronomy, 2010: 9.

KUBáNEK P, 2012. Rts2: meta-queues scheduling and its realisation for flwo 1.2m telescope[J]. Proc Spie, 8448(8): 11.

KUBáNEK P, VíTEK S, 2006. Rts2: a powerful robotic observatory manager[C]//SPIE Astronomical Telescopes + Instrumentation. [S.l.: s.n.].

KUBáNEK P, JELíNEK M, FRENCH J, et al., 2008. The rts2 protocol[C]//Advanced Software and Control for Astronomy II. [S.l.: s.n.]: 70192S–70192S–12.

KUBáNEK P, PROUZA M, KOTOV I, et al., 2012. Use of rts2 for lsst multiple channel ccd characterisation[J]. Proceedings of SPIE - The International Society for Optical Engineering, 8451: 84512T–84512T–12.

LI T, GU Y, GUO L, et al., 2016. Design of multi-motor distributed control system for optical fibers positioning based on can bus[C]//SPIE Astronomical Telescopes + Instrumentation. [S.l.: s.n.]: 99125I.

MILLS D, SCHUMACHER G, LOTZ P, 2016. Lsst communications middleware implementation [C]//Ground-based and Airborne Telescopes VI. [S.l.: s.n.]: 99065C.

Q. C X, H. Z Y, Q. C Y, et al., 2012. The large sky area multi-object fiber spectroscope (lamost)[J]. RAA(4): 1197–1224.

SCHMIDT D C, 2005. Tao developer's guide[M]. [S.l.]: Object Computing Inc.

SHI J R, 2015. The large sky area multi-object fiber spectroscopic telescope[J]. Bulletin of the Chinese Academy of Sciences, 12(3): 257–260.

TANENBAUM A S, STEEN. M V, 2007. Distribute systems principles and paradiamsg[M]. [S.l.]: Person Prentice Hall.

WANG S, SU D, CHU Y, et al., 1996. Special configuration of a very large schmidt telescope for extensive astronomical spectroscopic observation[J]. Applied Optics, 35(25): 5155.

ZHANG G, WANG J, TANG P, et al., 2015. An autonomous observation and control system based on epics and rts2 for antarctic telescopes[J]. Monthly Notices of the Royal Astronomical Society, 455(2): 1654–1664.

ZHAO Y, 2014. Spectroscopic survey of lamost[C]//SPIE Astronomical Telescopes + Instrumentation. [S.l.: s.n.]: 914517.

ZOU S, WANG G, 2006. The ccd imaging systems for lamost[C]//Society of Photo-Optical Instrumentation Engineers. [S.l.: s.n.]: 626922–626922–8.

作者简历及攻读学位期间发表的学术论文与研究成果

作者简历

田园，河北省定兴县人，中国科学院国家天文台博士研究生。

教育经历

2001年9月–2005年7月，燕山大学，电子信息工程，学士
2006年9月–2009年6月，燕山大学，信号与信息处理，硕士
2012年9月–2018年6月，中国科学院国家天文台，天文技术与方法，博士

工作经历

2009年7月至今，中国科学院国家天文台

已发表(或正式接受)的学术论文:

[1] Yuan Tian, Zheng Wang, Jian Li, Zi-Huang Cao, Dai Wei, Wei Shou-lin, Yong-Heng Zhao. LAMOST CCD camera-control system based on RTS2[J] RAA 2018 Vol.18 No.5, 54(14pp). doi: https://doi.org/10.1088/1674–4527/18/5/54
[2] 田园, 王锋, 李建, 王政, 赵永恒. 基于Python协程技术的LAMOST控制节点分布状态采集与监视系统[J] 天文研究与技术 2018 (已接收)

攻读博士学位期间参加的科研项目:

[1] 国家自然科学基金青年科学基金项目，“基于新型中间件的大型望远镜控制计算机集群资源监控系统”(批准号 11603044)，2017年1月至2019年12月，项目责任人

致　谢

值此论文即将完成之际，请允许我对工作九年来所有关心和帮助过我的老师、同学和同事表示真心的感谢!

首先，感谢我的导师赵永恒研究员！在论文工作的各个阶段，赵老师都花费了巨大的经历和心血。正是在赵老师的精心指导下，我才能够顺利地完成这篇论文。赵老师学识渊博，为人谦虚诚恳，待人亲切宽厚。作为我学业上的导师和工作中的领导，在各方面都给予我极大的帮助，也始终是我学习的榜样。

在论文进行过程中，王锋教授也给予了我很大的帮助，王老师严谨的治学作风、忘我的工作态度和丰富的工作经验都使我受益匪浅。

感谢施建荣研究员，作为我工作中的领导，对于我完成学业给予了极大的支持和充分的肯定。

感谢罗阿理研究员，是您带我入门并为我打开了天文技术这扇大门，使我领略到其中的乐趣并决定继续向前。

感谢郭守敬（LAMOST）运行和发展中心的所有老师和同事以及南京天光所的同事，正是大家共同的努力才使LAMOST越来越好。

感谢中国科技大学的金革老师、王坚老师以及已经毕业走上工作岗位的董健和邓小超同学。在你们的无私帮助下，我才能以最快的速度理解OCS系统。

感谢师弟王政和同事李建、曹子皇在代码开发过程中给予我极大的帮助。

感谢师兄白仲瑞在论文格式和排版方面给予我很大的帮助。

还要感谢国家天文台教育处的艾华老师和马怀宇老师，在我攻读博士期间给予我的极大的帮助。

最后，还要感谢我的父母、妻子和儿子！感谢你们为我解决了生活上的后顾之忧，使我可以全身心投入到科研工作中来。