中 国 科 学 院 上 海 天 文 台 年 刊
2012 年 总第 33 期 ANNALS OF SHANGHAI OBSERVATORY ACADEMIA SINICA No. 33, 2012

上海 65 m 射电望远镜单天线观测系统分布式中间件技术研究

董 健，赵融冰，左秀婷，范庆元

（中国科学院 上海天文台，上海 200030）

摘 要：位于上海市松江区的 65 m 射电望远镜是一架在建的大型射电望远镜。本文对 65 m 射电望远镜单天线观测系统中使用的分布式中间件——网络通信引擎 (Internet Communications Engine，ICE) 进行了研究。首先，对常用的分布式中间件进行分析和比较，选择 ICE 作为开发基础；接着，分析了基于 ICE 的 65 m 单天线观测系统的体系结构；最后，对观测系统中使用的 ICE 服务及其实现进行了阐述。结果表明，在应用中，ICE 能够满足需求。

关 键 词：上海 65 m 射电望远镜；单天线观测系统；分布式中间件；ICE

中图分类号：TP317；P228.6

1 引 言

上海 65 m 射电望远镜是一架在建的、全方位可动的大型射电望远镜，位于上海市松江区，其口径有 65 m，高达 70 m，重约 2700 t，采用修正型卡塞格伦抛物面设计，可工作在 L、S、C、X、Ku、K、Ka 和 Q 波段。建成后，它将在我国各项深空探测和天文学研究中发挥重要作用[1]。

上海 65 m 射电望远镜的观测模式主要分为 VLBI 观测和单天线观测两种。对于 VLBI 观测，世界上绝大多数射电望远镜都使用 FS(field system) 系统。FS 系统最初由 NASA 开发，随着需求的变化，FS 系统不断升级，目前版本是 9.10.4。FS 系统支持大多数 VLBI 终端系统，并通过挂靠本地台站程序来支持各台站特有的各种硬件和天线、接收机系统[2]。上海 65 m 射电望远镜的 VLBI 观测同样使用 FS 系统。对于单天线观测，目前还缺乏统一的观测软件系统，各个台站都是依据自身的观测需求单独开发的。本项目组依据上海 65m 射电望远镜的科学目标和软、硬件系统，正在开发单天线观测系统。

上海 65 m 射电望远镜拥有众多系统，例如终端系统、接收机系统、主动面系统、气象系统、时间频率系统和数据处理系统等。这些系统位于不同的位置，物理位置的隔离自然要求单天线观测系统是一个分布式的系统，并且，各个子系统的控制软件采用不同的技术来实现，运行在不同的操作系统之上，因此，设计和实现这样一个分布式的单天线软件系统是非常困难的。

如图 1 所示，中间件是在操作系统的编程接口之上提供的一个高层的应用程序接口 (appliaction programming interface，API)，以简化分布式软件开发人员的开发工作。分布式

收稿日期：2012-09-10； 修回日期：2012-09-20

软件系统开发和集成首先面对的就是多重异构问题，包括硬件和网络的异构、操作系统和编程语言的异构等。中间件作为一层共性的支撑软件，其主要的作用就是为用户屏蔽这些异构。除此之外，中间件还提供一组共性支撑的服务，如支持位置透明的命名服务，支持失效透明的容错服务，以及安全、实时、事务等相关服务[3]。

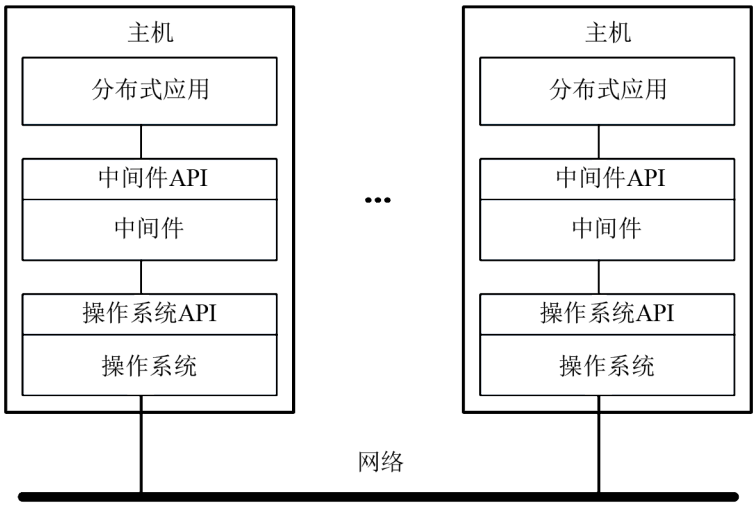

图 1　分布式中间件示意图

目前，几乎所有的望远镜观测系统的开发都基于分布式中间件。采用中间件可以缩短开发周期，增强系统的可靠性，使开发人员只需要关注于观测系统的业务逻辑，降低了开发难度。因此，上海 65 m 单天线观测系统也采用分布式中间件作为开发基础。

2　分布式中间件选择

分布式中间件产品很多，在天文观测控制领域，应用较多的有 CORBA、EPICS 和 ICE 三种。

CORBA(Common Object Request Broker Architecture) 是 1990 年代由 OMG(Object Management Group) 组织制订的一种为解决分布式处理环境中硬件和软件系统的互连而提出的解决方案。CORBA 定义了一系列 API、通信协议和信息模型，使各种不同的应用程序能够互操作[4]。1990 年代修建的 Gemini 使用 CORBA 开发了观测系统。CORBA 庞大而复杂，致使 CORBA 编写和调试应用程序非常困难，开发人员需要将大部分精力集中于 CORBA 技术本身。

EPICS(Experimental Physics and Industrial Control System) 是 1990 年代初由美国洛斯阿拉莫斯国家实验室 (Los Alamos National Laboratory，LANL) 和阿贡国家实验室 (Argonne National Laboratory，ANL) 等联合开发的大型控制软件系统。EPICS 软件系统中的两个基本机制是通道访问和分布式动态数据库。EPICS 系统采用了客户端–服务端模型，基于 TCP/IP 协议，建立了通道访问机制，并为客户和服务器分别提供了应用接口。EPICS 主要应用于底层硬件设备的监控，对于上层应用程序没有提供完整的支持。

网络通信引擎 (Internet Communications Engine，ICE) 是一款近些年由 ZeroC 公司出品的、现代的面向对象中间件，可用于替代 CORBA 这样的复杂中间件。它易于学习，同时提供了强大的网络基础功能，拥有足够的可伸缩性和安全性。它提供了一组完整的特性，支持广泛领域中的实际的分布式应用开发，避免了不必要的复杂性，使平台更易于学习和使用，提供了一种在网络带宽、内存使用和 CPU 开销方面都很高效的实现[5]。ICE 支持 C++、Java、Python、PHP、C# 和 Visual Basic 等多种语言影射，非常适合于在异构的环境中使用。因此，本项目组采用 ICE 作为观测软件的开发基础。

3 65 m 单天线观测系统体系结构

65 m 单天线观测系统体系结构如图 2 所示，主要分为三个部分，分别是：监视和控制系统、仪器监控系统和集成天文观测系统。

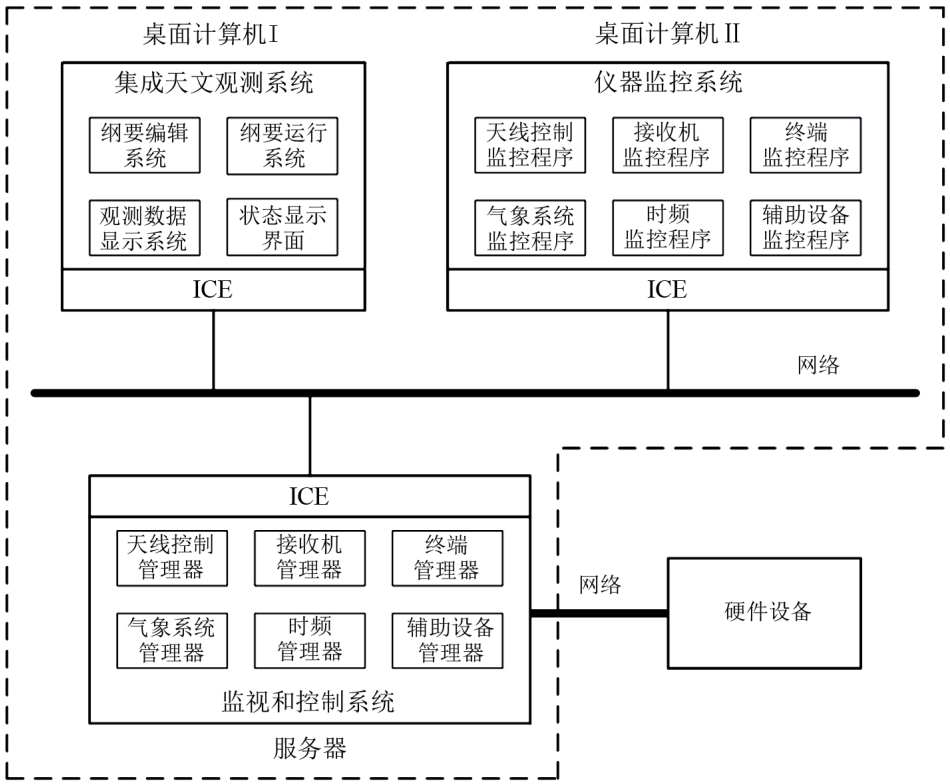

图 2 上海 65 m 单天线观测系统体系结构

监视和控制系统用于对底层设备进行控制，它包括天线控制管理器、接收机管理器、终端管理器、气象设备管理器、时频管理器以及辅助设备管理器等，这些管理器通过网络与实际设备相连。底层设备的通信协议有很大差别，监视和控制系统封装了这些差异，将设备封装成一个管理器类，对外提供远程函数调用接口。

工程师使用仪器监控系统对底层设备状态进行监控。仪器监控系统包括的模块有：天线控制监控程序、接收机监控程序、终端监控程序等，这些程序通过监视和控制系统中的管理

器远程接口对底层设备进行监控。

观测员和天文学家使用集成天文观测系统进行天文观测。此系统包含的模块主要有：纲要编辑系统、纲要运行系统、观测数据显示和状态显示系统等。集成天文观测系统功能复杂，它包含了所有的天文观测业务逻辑，并进行封装，对外提供简单的操作界面。

监视和控制系统运行在位于机房的服务器上，集成天文观测系统和仪器监控系统分别运行在观测室的两台台式机上，它们之间通过 ICE 进行连接。

4　分布式中间件 ICE 的使用

在单天线观测系统中，主要使用了 ICE 的三种服务：远程调用服务、消息发布和订阅服务以及版本化服务。从图 2 可知，监视和控制系统与其他两个系统位于不同的机器上，使用 ICE 的远程调用来实现接口调用。仪器设备需要将状态不断汇报给上层应用程序，上层应用程序获取状态主要通过两种方式：一种是“拉模式”，即通过不断调用状态获取函数接口来获取状态；另一种方式是“推模式”，即底层设备有新的状态时，通过 ICE 的消息发布和订阅服务将状态发送给相关的程序。目前，这两种方式在观测系统中均得到应用。观测系统的不断完善，会导致软件模块的版本不断变化，使用 ICE 的版本服务能便于软件模块的升级和版本维护。下面具体介绍三种服务的实现方式。

4.1　远程调用实现

远程调用实现主要分为三个步骤：首先，使用 Slice 语言定义远程调用接口并编写接口文件，使用 slice2cpp 编译程序将接口文件编译成.h 和.cpp 两个 C++ 文件 (根据接口文件生成的语言不同，可以选择不同的编译程序，例如 slice2java 将 Slice 文件编译成 Java 文件，slice2py 将 Slice 文件编译成 Python 文件等)；接着，依据编译生成的.h 和.cpp 文件，实现管理器对象，完成服务器端程序；最后，完成客户端程序，调用远程接口。

下面以辅助设备管理器中的微波矩阵管理器为例，具体描述远程调用的实现过程，结构图如图 3 所示。微波矩阵由 Dow-Key 公司生产，型号为 4104-ENET，它有 4 个 10 选 1 开关，每个开关用于将 10 路输入信号选择 1 路输出，微波矩阵有远程和本地两种控制方式，远程方式通过网口实现，并提供了通讯控制协议。微波矩阵管理器需要实现微波矩阵的初始化、重置、设置输出、获取状态等功能，并为外部模块提供远程调用接口。微波矩阵管理器位于图 2 中的服务器中，客户端程序位于桌面计算机 II 中。

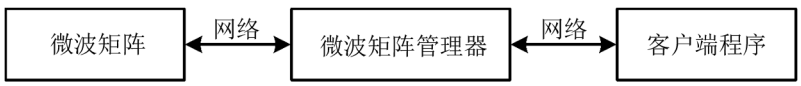

图 3　微波矩阵远程控制结构图

首先，在 microwavemanager.ice 文件中定义接口，如下：

```
Interface rpcmicrowavemanager
{
```

```
    bool init();                          //初始化
    bool reset();                         //重置
    string setPos(string x, string n);    //设置开关输出
    string getPos(string x);              //获取开关状态
};
```

运行 slice2cpp microwavemanager.ice 命令，编译生成 microwavemanager.h 和 microwavemanager.cpp 两个文件。

接着，实现 rpcmicrowavemanager 接口。在 microwavemanager.h 文件中包含了生成的类定义文件，在实现接口时，需要包含这个文件，并继承对应的类，如下：

```
class iceMicrowareManager : public rpcmicrowavemanager
{
    virtual bool init(const :: Ice :: Current&);
    virtual bool reset(const :: Ice :: Current&);
    virtual string setPos(string&x, string&n, const :: Ice :: Current&);
    virtual string getPos(string&x, string&n, const :: Ice :: Current&);
};
```

依据通讯协议，实现以上接口，从而实现微波矩阵管理器，并编写服务器端程序 (将微波矩阵管理器对象注册到对象适配器上，开启服务，等待外部调用)。服务器端的核心程序如下：

```
int main(int argc, char* argv[ ])
{
    int status = 0;
    Ice::CommunicatorPtr ic;
    //初始化
    ic = Ice::initialize(argc, argv);
    //创建对象适配器，并指定对象适配器的 ID、协议、IP 和端口
    Ice::ObjectAdapterPtr adapter = ic->createObjectAdapterWithEndpoints
                (“MicrowareManagerAdapter”, “ tcp -h xxx -p 10000 ”);
    //创建微波矩阵管理器对象
    Ice::ObjectPtr object = new iceMicrowareManager;
    //指定微波矩阵管理器对象的 ID 并将它添加到对象适配器中
    adapter->add(object, ic->stringToIdentity(“MicrowareManager”));
    //激活对象适配器
    adapter->activate();
    //监听、等待外部调用
    ic->waitForShutdown();
    return status;
```

```
}
```

最后，实现客户端程序，对微波矩阵远程控制，核心程序如下：

```
int main(int argc, char* argv[ ])
{
        int status = 0;
        Ice::CommunicatorPtr ic;
        ic = Ice::initialize(argc, argv);
        //通过微波矩阵管理器的 ID、IP 和端口号获取引用
        Ice::ObjectPrx base=ic->stringToProxy("MicrowareManager:: tcp -h xxx -p 10000");
        //类型转换
        rpcmicrowavemanagerPrx manager= rpcmicrowavemanagerPrx::checkedCast(base);
        //调用远程初始化接口
        manager-> init();
        return status;
}
```

4.2　消息发布和订阅实现

消息发布和订阅服务主要通过 ICE 的 IceStorm 服务来实现。IceStrom 简化了消息的传递，将消息发布对象和消息订阅对象解耦合，增强了扩展性。

消息发布和订阅服务的实现主要分为三个步骤：首先，定义主题 (topic) 对象接口 (topic 接口是发布端和订阅端的通信接口)；接着，实现发布端，包括创建主题和发布消息；最后，实现订阅端，包括实现 topic 接口，获取主题和注册。以时频管理器中的 GPS 管理器的时间信息发布为例，具体描述消息发布和订阅的实现过程。如图 4 所示，GPS 工作站测量并收集 GPS 时间数据，通过网络将数据发送给 GPS 管理器，GPS 管理器将数据发送给 IceStrom 服务，IceStrom 将信息发送给订阅者。

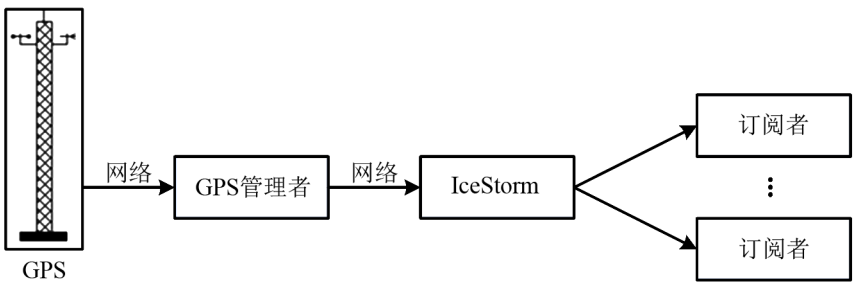

图 4　GPS 时间数据发送过程

首先，在 gpsinfo.ice 中，定义 topic 接口，如下：

```
Interface gpsMonitor
{
```

```
    void report(string time);
};
```

运行 slice2cpp，将 gpsinfo.ice 编译成 gpsinfo.h 和 gpsinfo.cpp 两个文件。
接着，实现发布端，创建 topic，再发送消息，核心程序如下：

```
int main(int argc, char* argv[ ])
{
    ...
    //获取 Topic Manager 的引用
    Ice::ObjectPrx obj = communicator->stringToProxy
                    ("IceStorm/TopicManager:tcp -h xxx -p 9999");
    IceStorm::TopicManagerPrx topicManager
    = IceStorm::TopicManagerPrx::checkedCast(obj);
    IceStorm::TopicPrx topic;
    //创建主题
    topic = topicManager->create("GPS");
    Ice::ObjectPrx pub = topic->getPublisher()->ice_oneway();
    gpsMonitorPrx monitor = gpsMonitorPrx::uncheckedCast(pub);
    //发送信息
    monitor->report(time);
    ...
}
```

最后，实现订阅端，与远程调用过程类似，实现 gpsMonitor 接口 (代码略)，通过接口处理消息，核心程序如下：

```
int main(int argc, char* argv[ ])
{
   ...
   Ice::ObjectPrx obj = communicator->stringToProxy
                   ("IceStorm/TopicManager:tcp -h xxx -p 9999");
   IceStorm::TopicManagerPrx topicManager
   = IceStorm::TopicManagerPrx::checkedCast(obj);
   Ice::ObjectAdapterPtr adapter
   = communicator->createObjectAdapter("MonitorAdapter");
   //实例化 topic 对象
   gpsMonitorPtr monitor = new gpsMonitorI;
   //添加 topic 对象到对象适配器中
   Ice::ObjectPrx proxy = adapter->addWithUUID(monitor)->ice_oneway();
   IceStorm::TopicPrx topic;
```

```
    //获取主题
    topic = topicManager->retrieve("GPS");
    IceStorm::QoS qos;
    //订阅主题
    topic->subscribeAndGetPublisher(qos, proxy);
    adapter->activate();
    communicator->waitForShutdown();
    topic->unsubscribe(proxy);
    ...
}
```

4.3　版本化实现

版本化通过 ICE 的 Facets 来实现，不同的实现对象可以通过 Facets 加入到适配器对象中，从而最小化软件系统的修改。客户端根据需要选择相应的版本即可。

5　结　论

本文对上海 65 m 射电望远镜单天线观测系统中使用的分布式中间件 ICE 进行了研究。ICE 在系统的设计和实现中发挥了重要作用。在初步应用中，ICE 达到了预期效果。下一步的工作中，将依据需求的变化，选择相关的 ICE 服务，并与整个系统融合起来。

参考文献：

[1] http://65m.shao.cas.cn/

[2] 薛祝和. 中国科学院上海天文台年刊，2011，32：154

[3] Chiozzi G, Wallander A. Trends in Software for Large Astronomy Projects, Proc of ICALEPCS07, Knoxville, Tennessee, USA: SNS and TJNAF, 2007: 13

[4] 董健. 博士论文，合肥：中国科学技术大学，2011

[5] http://www.zeroc.com/Ice-Manual.pdf

The Research of Distributed Middleware Technique in Single-dish Observation System of Shanghai 65-Meter Radio Telescope

DONG Jian, ZHAO Rong-bing, ZUO Xiu-ting, FAN Qing-yuan

(Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030)

Abstract: Shanghai 65-meter radio telescope is a telescope under construction at Songjiang District of Shanghai. The application of distributed middleware-ICE (Internet Communications Engine) is researched in this paper. Firstly, the common distributed middleware libraries are analyzed and compared, the ICE is selected as development foundation. Then, the architecture of 65-meter single-dish observation system is analyzed. Finally, the ICE services and its implementation methods are described. The results show that the ICE can satisfy the needs.

Key words: Shanghai 65-meter radio telescope; single-dish observation system; distributed middleware; ICE