基于. NET反射技术的规约插件实现原理

朱有产,李玉凯,李自强

(华北电力大学信息与网络管理中心,河北 保定 071003)

摘要: 针对 SCADA系统中远动规约标准的不统一,在分析现有解决方案的基础上,提出了一种基于. NET反射技术的规约插件设计与实现原理。通过将规约处理模块封装成程序集,通信主程序利用反射技术动态加载该程序集,调用其中相关对象的接口函数以完成远动规约的转换。这种设计模式不仅简化了规约插件的开发和部署,而且避免了在 Windows环境下所遇到的"DLL地狱"问题。这为 SCADA系统中多厂家 RTU的集成、多种远动规约的集成提供了灵活方便的解决方案。文中最后给出了用 C#实现的具体方法和步骤。

关键词: . NET反射技术; 规约插件; SCADA; COM

中图分类号: TM76 **文献标识码**: A **文章编号**: 1003-4897(2006)22-0060-04

0 引言

在电网 SCADA (Supervisory Control and Data Acquisition)系统中,监控主站系统依靠通信处理机与现场的 RTU(远程终端)通信,以完成命令下发和数据采集的功能[1]。因此,通信处理机是系统中上传下达的通道,起着非常重要的作用。目前,国内对 SCADA系统中设备之间的通信还未制定统一的接口标准[2],通信规约种类繁多,如: SC1801规约、部颁标准 Polling规约、部颁 CDT规约等。由于系统中不同厂家的 RTU遵循不同的通讯规约,这给 SCADA系统的维护和扩展等带来不便,通信规约的转换成了系统中必不可少的环节[2]。

针对上述情况,就要求通信处理机能够支持不同的规约。现有的 SCADA商品软件的一般做法是:把它支持的各种通信规约全部封装在通信处理程序内部,通过配置程序选择适用的规约;或者,用基于 COM技术的规约插件设计思想,将规约处理程序设计为插件[3]。

这些做法有以下不足: a. 把规约处理程序封装在 SCADA系统内部,必然会加大系统对其提供商的依赖性,造成系统不灵活,不能动态加入其它新的规约处理程序,使整个系统的扩展受到限制。b. 基于 COM技术的规约插件设计思想虽然有很多优点,但是, COM技术仍然面临很多难以解决的问题。首先, COM组件并不容易编写,它提供的功能取决于编写它所使用的语言。重要的是, COM的部署和维护比较困难,新旧版本必须保持兼容,否则会产生所谓的"DLL地狱"问题[4]。

为此,本文提出了一种新的规约插件解决方案:在 Windows环境下,利用. NET框架提供的反射技术代替 COM技术,将规约处理程序设计为规约插件。这样,既降低了规约插件的实现技术难度,又保持了其原有的灵活方便性,同时避免了因采用 COM技术而可能带来的"DLL地狱"问题。

1 . NET反射技术

. NET框架是微软公司大力推广的新一代软件平台,它为用户提供了更为方便的开发平台和更为丰富的类库资源,并且提出了程序集的概念。而反射技术正是. NET提供的程序集高级技术。

1.1 . NET程序集技术[5]

在. Net框架中,程序集是自我描述的单元,软件(包括控件、窗体和其他运算代码)是以程序集的方式存在的。它构成了部署、版本控制、重复使用、激活范围控制和安全权限的基本单元,并为公共语言运行库提供它要用以识别类型实现的信息。其主要功能为:包含公共语言运行库执行的代码、形成安全边界、形成类型边界、形成引用范围、形成版本边界、形成部署单元。

程序集的优点:程序集旨在简化应用程序部署并解决在基于组件的应用程序中可能出现的版本控制问题。由于程序集是不依赖于注册表的自述组件,它的安装仅是复制(使用 xcopy命令)所用的文件而已。所以程序集能使无相互影响的应用程序安装成为可能,使应用程序的卸载得以简化,解决了软件版本控制问题以及导致 DLL冲突的问题。

1.2 . NET反射技术

反射技术是指:使用程序通过检查程序集的单

个模块来查看其内容的能力。程序集包含模块,而模块包含类,类又包含成员,由于程序集是自描述的,因此通过反射技术,可以在运行时获得程序集中每一个类型(包括类、结构、委托、接口和枚举等)的成员,包括方法、属性、事件,以及构造函数等。还可以获得每个成员的名称、限定符和参数等。有了反射技术,我们可以动态地创建类型的对象,即使这个对象的类型在编译时还不知道。

反射技术具体用途[5]:a. 使用 Assembly定义和加载程序集,加载在程序集清单中列出的模块,以及从此程序集中查找类型并创建该类型的对象。b. 使用 Module了解包含模块的程序集以及模块中的类等,还可获取在模块上定义的所有全局方法或其他特定的非全局方法。c. 使用 ConstructorInfo了解构造函数的名称、参数、访问修饰符(如 pulic、private)和实现详细信息(如 abstract、virtual)等。使用 Type的 GetConstructors或 GetConstructor方法来调用特定的构造函数。d. 使用 MethodInfo了解方法的名称、返回类型、参数、访问修饰符和实现详细信息等。使用 Type的 GetMethods或 GetMethod方法来调用特定的方法。e. 使用 FieldInfo了解字段的名称、访问修饰符和实现详细信息等;获取或设置字段值。f. 使用 EventInfo了解事件的名称、事件处理程序、数据类型、自定义属性、声明类型和反射类型等;添加或移除事件处理程序。g. 使用 PropertyInfo了解属性的名称、数据类型、声明类型、反射类型和只读或可写状态等;获取或设置属性值。h. 使用 ParameterInfo了解参数的名称、数据类型、是输入参数还是输出参数,以及参数在方法签名中的位置等。

2　基于.NET反射技术的规约插件实现原理

2.1　插件设计技术

软件插件是近年来十分常见的一种技术。插件结构有助于编写具有良好的扩充和定制功能的应用程序。插件的本质是在不修改程序主体的情况下对软件功能进行加强或用于在创建程序之后为其添加额外的功能。当插件的接口被公开时,任何人都可以自己制作插件来解决一些操作上的不便或增加一些功能。一个插件框架包括两个部分:主程序(host)和插件(plug-in)。主程序即是"包含"插件的程序。插件通过若干标准接口,与主程序进行信息的交互。基于插件技术的软件开发可以使产品标准化、系列化。通过不同规格的插件的组合,可快速完成应用系统集成,满足客户的需求和升级。

利用.NET反射技术能方便地创建这种软件构架,将 SCADA系统中的规约处理程序设计成规约插件,可方便扩充系统的规约转换功能。

2.2　规约插件的设计[3]

通信主程序通过规约插件与 RTU之间进行数据交换。因此,规约插件应具有以下功能:一是向下,实现与各种规约的 RTU接口,接收 RTU上传的实时数据、并且向 RTU发送命令;二是向上,为通信主程序提供标准的数据格式,使得通信主程序完全不必关心使用何种 RTU及通信规约。

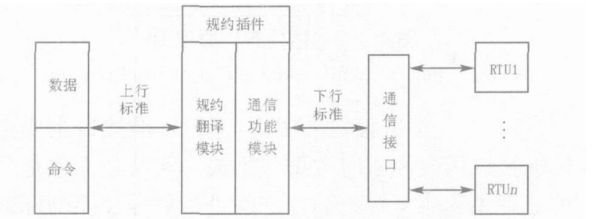

图 1　规约插件的框架图

Fig 1　Structure of protocol plug-in

规约插件框架图如图 1所示。a. 规约翻译模块:针对各种具体的通信规约,将 RTU上传的实时数据进行语义解释,取出相应的遥信、遥测等实时量;另一方面,将通信处理机下传的遥控、遥调命令按照各种规约的具体要求组帧,再通过通信模块传达到 RTU上。b. 通信功能模块:负责数据帧的发送与接收。c. 上行标准接口:是规约插件和通信处理主程序之间的标准接口,将实时数据、命令数据等以统一格式存放于标准的实时数据包中,通信主程序与规约插件通过接口来传递标准的实时数据包以完成数据和命令的交换。d. 下行标准接口:是规约插件和各种串行通信之间的标准接口,实现规约插件与 RTU的二进制位流的传递。

2.3　基于反射技术的规约插件实现原理

在.NET框架下,实现基于反射技术的规约插件的步骤是:首先根据不同的规约和通信标准,开发人员设计具体的翻译类和通信类来实现图 1中规约翻译模块和通信功能模块的功能,这两个类需要分别继承相应的接口并实现接口中的函数,并将这两个类封装成一个程序集(DLL 文件),即为规约插件,将其 copy到相应目录下便完成插件的部署;通信主程序根据文本配置,利用.NET的反射技术进行动态加载每个通信通道的规约插件程序集,动态生成每个通道的翻译对象和通信对象。通信主程序通过调用翻译对象和通信对象继承的接口函数完成规约翻译及通信功能。规约插件实现原理图如图 2

所示。具体实现如下:

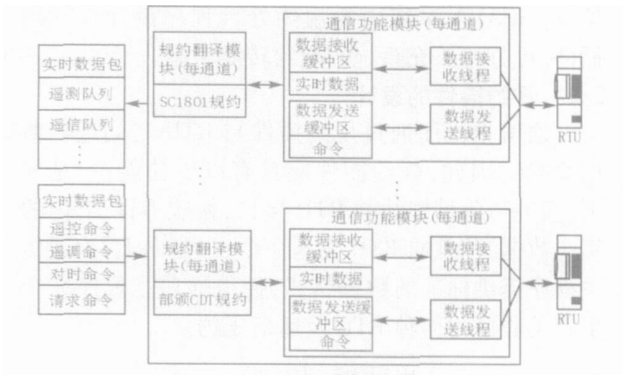

图 2 规约插件实现原理图

Fig 2 Implementation principle of protocol plug-in

1) 上行标准接口的实现:上行标准接口主要由标准接口函数和实时数据包组成。实时数据包是通信处理主程序与规约插件之间进行数据交换的标准数据结构,主要包含数据的相关属性、遥测数据队列、遥信数据队列等,以用于装载各种实时数据和下发的命令。实时数据包的功能通过设计一个类来实现,将该类封装成独立的程序集,以便在主程序和规约插件中引用。主要代码如下:

```
public class TransData //数据包装类
{ public TransData(){} //构造函数
private ArrayList SignalList = new ArrayList(); //遥信数据队列
private ArrayList MeasureList = new ArrayList(); //遥测数据队列
private ArrayList Command = new ArrayList(); //遥控命令队列
... }
```

标准接口函数是通信主程序和规约插件信息交互的接口。将接口函数进行封装并定义成.NET框架中的 interface接口类型,用于被后面的模块继承和实现。规约通信和翻译接口的主要代码:

```
public interface ITranslator//翻译接口
{ //初始化接口函数
bool IniTranslator(string settings);
bool FinTranslator(); //终结处理函数
//将实时数据包翻译为字节流
byte[] GetBytes(TransData data);
//从字节流拼装实时数据包
TransData GetPackage(byte[] bytes);
... }
public interface ICommunicator//通信接口
{ //初始化接口函数
bool IniCommunicator(string settings);
bool FinCommunicator(); //终结处理函数
bool Send(byte[] bytes); //发送字节流
byte[] Receive(); //读取接收到的新字节数据
... }
```

2) 规约翻译模块和通信功能模块的实现:基于面向对象的编程思想,开发人员根据不同的规约和通信标准设计具体的翻译类和通信类,其中,翻译类必需继承和实现 ITranslator接口,通信类必需继承和实现 ICommunicator接口。将设计好的两个模块编译成程序集,如命名为 protocol dll,既为规约插件。翻译类主要代码如下:

```
public class Translator: ITranslator
{ //实现所继承接口的函数
byte[] GetBytes(TransData data) {... };
TransData GetPackage(byte[] bytes) {... };
... }
```

通信类主要代码:

```
public class Communicator: ICommunicator
{ //实现所继承接口的函数
bool Send(byte[] bytes){... };
byte[] Receive(){... };
... }
```

3) 下行接口标准的实现:下行接口是规约插件与 RTU之间的通信接口,实现面向数据流、面向字节的传输控制。根据不同的接口标准(TCP/IP、串口等),可在通信类的相应函数中编写代码实现与RTU通信的功能。

4) 通信主程序的插件处理机制:主程序根据配置找到相应目录下的规约插件程序集(protocol dll),通过 .NET框架提供的反射技术,使用 Assembly动态加载规约插件的程序集:

```
Assembly translatorAssembly = Assembly.LoadFile("protocol dll");
```

使用 GetType方法得到翻译类和通信类的类型:

```
Type translatorType = translatorAssembly.GetType("translator");
Type communicatorType = translatorAssembly.GetType("communicator");
```

根据得到的类型,动态创建翻译对象和通信对象:

```
object translatorObj = Activator.CreateInstance(translatorType);
Object communicatorObj = Activator.CreateInstance(communicatorType);
```

主程序在生成对象后首先调用对象继承的接口

初始化函数,以便完成相关初始化操作,如调用翻译对象的 IniTranslator函数:

```
bool iniSuccess = translatorObj.GetType().
InvokeMember("IniTranslator", BindingFlags.
InvokeMethod, null, translatorObj, new object[] {set-
tings});
```

每个规约插件可支持一个与 RTU相连的通信通道,接着通信主程序为每个通道创建接收和发送线程。接收线程调用通信对象继承的 Receive()函数,从通信接口得到数据帧的字节流,通过调用规约翻译对象的 GetPackage(byte[] bytes)函数将字节流进行翻译成实时数据包,而后将它上送给数据处理系统,到此完成了上行实时数据的翻译。同理,主程序通过发送线程先调用翻译对象继承的 GetBytes(TransData data)函数,并将下发的命令实时数据包作为调用参数,得到该函数返回的翻译后的字节流,接着调用通信对象的 Send(byte[] bytes)函数,完成对 RTU数据帧的发送。

在多通道的情况下,每个通道根据需要可配置不同的规约插件。结合. NET框架的多线程技术,通信处理程序可方便地实现并发与多个不同规约的 RTU通信。

3　结束语

将每一种规约翻译模块与通信功能模块封装成一个程序集,开发人员通过对翻译类和通信类以及其所继承接口的不同实现来创建不同的规约插件,利用. NET反射技术动态加载规约插件的程序集,以完成通信和规约翻译的功能。采用这种技术,简化了规约插件的实现,使用 copy命令即可完成插件的部署,并且不会引发规约插件版本冲突的问题。

在工程实际应用中,基于. NET反射技术的规约插件所具备的优秀特性,为用户在 SCADA系统中实现多种远动规约的集成提供了方便、灵活的解决方案。

参考文献:

[1] 王常力,罗安. 分布式控制系统 (DCS)设计与应用实例 [M]. 北京:电子工业出版社, 2004.
WANG Chang-li, LUO An. Distributed Control System Design and Its Applications [M]. Beijing: Publishing House of Electronics Industry, 2004.

[2] 张岳匀,何志伟. SCADA系统通信规约的标准化建议 [J]. 电力系统及其自动化学报, 2000, 12(5): 42-44.
ZHANG Yue-yun, HE Zhi-wei. Protocol Standardization of SCADA System and Advice to It[J]. Proceedings of the EPSA, 2000, 12(5): 42-44.

[3] 贾春娟,郭桂同. COM技术及其规约插件的实现原理 [J]. 继电器, 2001, 29(3): 49-51.
JIA Chun-juan, GUO Gui-tong. Technology of COM and Implementation Principle of Its Protocol Card[J]. Relay, 2001, 29(3): 49-51.

[4] 花振峰,杨伟民,张 生. NET组件和 COM组件的应用集合研究 [J]. 电子工程师, 2005, 31(2): 71-73.
HUA Zhen-feng, YANG Wei-min, ZHANG Sheng. A Study on the Application of. NET and COM Components [J]. Electronic Engineer, 2005, 31(2): 71-73.

[5] 徐卫东,何江,张峻. 基于. Net反射技术的动态界面的实现 [J]. 计算机工程与设计, 2003, 24(10): 57-59.
XU Wei-dong, HE Jiang, ZHANG Jun. Design and Implement Method for Dynamic User Interface Based on. Net Reflection Technology [J]. Computer Engineering and Design, 2003, 24(10): 57-59.

收稿日期: 2006-06-21;　**修回日期**: 2006-08-01

作者简介:

朱有产 (1963 -),男,教授,硕士研究生导师,主要研究方向为网格计算,网络管理;

李玉凯 (1975 -),男,硕士研究生,主要研究方向为分布式系统、网格计算; E-mail: wslyk@sohu.com

李自强 (1982 -),男,硕士研究生,主要研究方向为分布式系统,数据挖掘。

An implementation principle of protocol plug-in based on technology of. NET reflection

ZHU You-chan, LI Yu-kai, LI Zi-qiang

(Center of Information and Network Management, North China Electric Power University, Baoding 071003, China)

Abstract: In allusion to the different remote protocol standards in SCADA system, the paper analyzes available solutions and proposes a kind of new design and implementation principle of protocol plug-in based on. NET reflection technology. By using the reflection technology, main communication program loads the assembly dynamically in which the implementation module of protocol has been encapsulated and calls the interface procedure of object in the assembly to implement the task of translation of remote protocol. This design not only simplifies the complexity of development and deployment of protocol plug-in, but also avoids the 'DLL Hell' problem which is

(下转第 83页　continued on page 83)

期允许通过电流为多大。现场由于条件限制没有进行试验。也就是说短路电流至少需要达到 14 A,自动开关才能跳开。而印制电路板的设计通流能力为 8 A左右,这样才出现装置烧毁的现象。根据《火力发电厂、变电所二次极限设计技术规程》要求,当运行电压为 90%额定电压时,二次电压回路末端经过渡电阻短路,加于(保护)继电器线圈上的电压低于 70%额定电压时,自动开关应瞬时动作。而当时发生电压互感器反充电时,相当于回路末端经过渡电阻短路。按照上述关系换算,运行电压为 105%额定电压,加于继电器线圈上的电压低于 81.3%额定电压(即 179 kV)时,自动开关应该跳闸。

对于后来检查隔离开关辅助控制触点输入正常,有多种可能。一、有可能因为机械应力,隔离开关操作完以后,辅助机构没有立即到位,但是经过一段时候自动到位,辅助触点转换正常。这种情况在以前出现过,也符合力学原理。二、现场机构调整,由于该地区为煤炭开采地区,地表受到一定影响,加之隔离开关的底座地面基础没有固定在一起,所以,每次操作完设备,都要进行机构调整。

综合以上分析可得出结论:由于操作过程中两路电压切换继电器同时动作,导致断开母联断路器时,北母电压互感器通过二次回路被反充电,电压切换回路过负荷烧毁。

4　总结

根据多年来从事继电保护行业工作的总结,现场曾多次发生类似的电压切换回路烧毁现象。其故障原因主要有两条:一、隔离开关辅助触点转换不可靠;二、电压互感器二次回路自动开关选型不合理。因此,建议在条件允许的情况下,将隔离开关辅助触点采用两副触点相互串联或并联后使用;隔离开关选择转换可靠的辅助机构。自动开关的选择一定要进行分断电流校验,选择合适的自动开关。另外,电压切换继电器采用双位置磁保持继电器同样是个值得讨论的问题。双位置继电器虽然可以保证在控制直流电源消失时二次电压的正常切换,但是,在切换过程中,要求两组控制触点必须同时正确动作才能保证切换的可靠。这就对隔离开关的辅助转换触点提出了更高的要求。

参考文献:

[1] 卓乐友.电力工程电气设计手册[M].北京:中国电力出版社,1989.
ZHUO Le-you. Manual for Designing of Electrical in Power Project[M]. Beijing: China Electric Power Press, 1989.

[2] DL/T 5136-2001,火力发电厂、变电所二次接线设计技术规程[S].
DL/T 5136-2001, Technical Code for Designing of Electrical Secondary Wiring in Fossil Fuel Plants and Substations[S].

收稿日期:　2006-05-24

作者简介:

郭占伟(1974-),男,工程师,从事电力系统继电保护工作设计与研究;guo7412@126.com

魏晓强(1977-),男,助理工程师,从事电力系统继电保护工作与研究;

肖志刚(1977-),男,助理工程师,从事电力系统继电保护工作与研究。

Analysis of busbar secondary voltage selection route faults

GUO Zhan-wei, WEI Xiao-gang, XIAO Zhi-qiang, LIU Yan-dong
(XJ Electric Protection & Automation Business Department, Xuchang 461000, China)

Abstract: The paper depicts an accident about voltage selection circuit in power plant. The principle and reason of accident about the voltage selection circuiting are analyzed. Measures of preventing similar accident in design and operation are presented.

Key words: busbar; selection relay; automatic switch; voltage transformer

(上接第 63页　continued from page 63)

existing under Windows environment. As a result, an easy flexible solution scheme for integrating different remote protocols and different RTUs from different manufacturers in SCADA system is available. Finally, the paper presents its implementation method and procedure with C# in detail.

Key words: .NET Reflection; protocol plug-in; SCADA; COM