第 19 卷第 4 期　　广西科学院学报　　Vol. 19, No. 4
2003 年 11 月　　Journal of Guangxi Academy of Sciences　　November 2003

RMI 在 Mobile agent 系统中的应用
Application of RMI in Mobile Agent System

蔡启先
Cai Qixian

（广西工学院计算机工程系　柳州　545006）
(Dept. of Computer Engineering, Guangxi Univ. of Technology, Liuzhou, 545006)

摘要　在介绍 RMI 技术的基础上，提出利用 RMI 作为通信中间件，构建一个通用远程接口，实现 Mobile agent 作为客户机来调用远程对象的目的。

关键词　Mobile agent　RMI　中间件　通用远程接口
中图法分类号　TP311.13

Abstract　A common remote interface based on RMI middleware is constructed with the mobile agent as a client machine, then the purpose of calling remote object is obtained.

Key words　Mobile agent, RMI, middleware, common remote interface

Mobile agent 是一种能在异构网络中移动，自主决定其行为的程序[1]。Mobile agent 具有移动性与自治性，使得它在电子商务、个人软件助理、分布式信息检索、电信网络服务、监视和通告、信息发布、移动设备计算、网络管理、并行任务求解、工作流管理和协作、动态网络等方面应用日益广泛[2]。由于 IT 厂商产品之间的差异，各个用户操作环境不尽相同，要真正实现 Mobile agent 的功能，就必须解决跨平台运行和分布式应用问题，因此，人们提出了中间件技术。中间件通常位于客户机服务器的操作系统之上，管理计算资源和网络通信，从而屏蔽不同厂商产品之间的差异，使不同技术之间共享资源，实现了分布式应用[3]。

目前较为流行的 3 种中间件技术为：OMG 组织制定的 CORBA 标准、Microsoft 的 COM/DCOM 标准和 SUN 公司的 Java RMI。3 种技术各有所长[4]。

RMI（Remote Method Invocation，远程方法调用）将 Java 特性扩展到分布式系统的领域中。由于 Java 本地模型的易用性，RMI 成为一种较简单和快捷的实现分布式对象结构的方式。RMI 的优势除了对象传递外，就是其绝对的平台无关性。由于目前 Java 语言已成为 Internet 上占绝对优势的程序语言，所以基于 Internet 的多操作系统平台应用系统来说，其优势是 DCOM 和 CORBA 所不能及的。但 RMI 要求编程人员必需熟知 Java 语言，而不象 DCOM 和 CORBA 那样，编程人员可以有较多地自主选择，这是 RMI 的缺陷。

本文在简介 RMI 技术的基础上，阐述如何具体应用 RMI 来支持 Mobile agent 技术。

2003-07-03 收稿。

1　RMI 技术简介

RMI[5]是建立分布式 Java 应用程序的有效途径。RMI 调用的是同一台或不同的计算机上另一个 JVM 中的 Java 对象。同时，RMI 是虚拟的无缝连接，可以象调用本地对象一样，通过 RMI 调用远程对象。

为实现 RMI 对远程对象的调用，客户机需要定义一个远程接口，该接口遵守下列规则[6]：（1）远程接口必须为 public 属性；（2）远程接口必须扩展接口 java. rmi. Remote。

用远程接口调用远程方法时，RMI 也通过这个远程接口来隐藏底层的实施细节，具体来说，RMI 在客户机端生成一个桩对象，当客户机调用远程方法时，利用桩将方法调用请求传输到服务器的 JVM 中，在这里另一个称为框架的特殊对象解释这个请求，并调用正确的方法。如果这个远程方法返回数值或抛出异常，则框架封装这个结果信息并将其发送回桩中，然后桩再将信息发送给客户机。图 1 显示客户机、桩、框架和服务器之间的关系。

使用桩的好处在于客户机并不知道自己是在与桩交流，而以为是通过接口调用本地方法；同样服务器也不知道有框架，而是把框架看成与使用服务器类的其他类一样。桩与框架之间利用 JRMP（Java Remote Method Protocol，Java 远程方法协议）进行通信。实际上，JRMP 只是桩与框架可以使用的协议之一。在 Sun 公司提供的 JDK1. 3 及以后版本中还可利用 RMI _ IIOP 协议[7]。

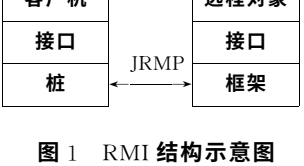

图 1　RMI 结构示意图

2　Mobile agent 技术与 RMI 的结合

RMI 包括客户端程序与服务器端程序，这决定 Mobile agent 与 RMI 结合时可能扮演的 2 种角色：作为客户机来调用远程对象或是作为对远程对象的具体实现。这里只讨论支持 Mobile agent 作为客户机来调用远程对象。

2. 1　Mobile agent 中 RMI 注册表对象的重载

RMI 客户注册就是通过 Java. rmi. Naming. bind（）方法或 Java. rmi. Naming. rebind（）方法，将 OperateImpl 实例绑定到指定的 URL 上。Naming 类提供了 RMI 注册表对象的接口，它的 bind、rebind 方法用于远程对象与对象名的绑定，lookup 方法用于从远程服务器返回远程对象的句柄。标准 Java 2 提供的 Naming 方法中用 java. lang. String 数据类型描述的形如“//host：port/name”的 URL 格式表示注册表服务器所在地址，但在 Mobile agent 系统中，agent 名、服务器名以及它们与网络地址间的对应关系取决于系统的命名规则，而且不同 Mobile agent 系统往往有不同的命名规则。因此，在 Mobile agent 系统中对 Naming 类的方法进行重载是必须的。

重载 Naming 类方法的关键是将 agent 名、服务器名解析为与其当前位置对应的网络地址和接口。不同 Mobile agent 系统有不同命名规则，重载 Naming 类的具体代码也不完全相同。

2. 2　RMI 服务器对象的生成

RMI 服务器的 Main 方法必须首先建立安全管理器，然后建立远程对象的一个或多个实例。但是在 RMI 中，客户端程序必须通过远程接口才能调用远程对象，Mobile agent 调用到的远程公用接口也必须随 agent 一起传递，这极大地增加了 agent 传输的复杂性（接收机需要

增加远程接口存在判定和传输的函数），也限制了 agent 调用 RMI 的灵活性（必须事先确定需调用的远程接口）。因此，本文在系统中利用 Java 的反射机制设计了一个通用的 RMI 远程接口，它包含于 Agent Server 软件包中，Mobile agent 无论在哪个 Agent Server 中都可直接对它进行调用。

图 2 是通用接口调用过程，客户名义上调用的是通用接口类，实际返回的却是调用参数中指明的远程对象。也就是说，实际的远程服务对象由通用远程接口统一管理，Mobile agent 通过远程对象名而不是对象句柄实现远程调用。因此，Main 程序只需建立通用远程对象的一个实例即可。程序如下：

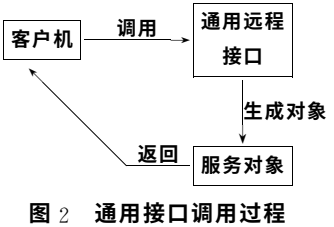

图 2 通用接口调用过程

```
public static void main(String[] args)
{ try
  { if (System.getSecurityManager()=null)
    {  System.setSecurityManager(new RMISecurityManager());//建立安全管理器
    }
    CommonFactoryImpl impl=new CommonFactoryImpl();//建立通用远程对象的一个实例
    Naming.rebind("//localhost/CommonFactory",impl);//在 RMI 的注册表中绑定通用远程对象
  }
catch(Exception exc)
  {  System.out.println(exc);
  }
}
```

2.3 远程通用接口的实现

RMI 中的远程接口必须是 java.rmi.server.UnicastRemoteObject 类的子类，并包含一个或几个远程方法。此外，还必须为远程对象明确定义构造器，并掷出 RemoteException 违例。

通用远程接口的定义如下：

```
public interface CommonFactory extends Remote
{ public Object getremoteObject(String servicename)throws RemoteException;
  public Object getremoteMethod(String servicename,String methodname,Class[]
  paramclass,Object[] params)throws RemoteException;
  public Object getremoteField(String servicename,String fieldname) throws
RemoteException;
}
```

整个接口主要包括 3 个方法：getremoteObject、getremoteMethod 和 getremoteField。通过 getremoteObject，用户可以在输入服务名的情况下获得远程对象对该种服务的一个具体实现；getremoteMethod 方法能够帮助用户实现对特定服务的特定方法的动态调用，它有 4 个参数，分别代表远程对象的服务名、用户要获取的方法的方法名、方法参数数据类型的 Class 对象数组（因为存在同名但参数不同的方法，所以必须指定参数类型）以及调用远程方法的具体参

数值；getremoteField 方法用于帮助用户通过属性名获取远程对象的属性值，它的 2 个参数分别代表远程对象的服务名与用户要获取的属性的属性名。

通用远程接口定义好后，还要在服务器端加以实现，在此我们只以 CommonFactory 接口中最具代表性，也是最困难的 getremoteMethod 方法为例对实现加以说明。见如下程序：

```
Public class CommonFactoryImpl extends UnicastRemoteObject implements
CommonFactory
{     ……
   public Object getMethod(String servicename,String methodname,Class[] paramclass,
   Object[] params)throws RemoteException;
      { Method m = servicetable.get(servicename).getclass.getMethod(methodname,
      paramclass);//通过方法名及所带参数确定需要服务的类
      Object o=m.invoke(servicetable.get(servicename),params);//动态生成类的实例
      return o;
      }
……
}
```

首先根据 Java RMI 的规定，实现程序必须继承自 UnicastRemoteObject 对象。其次，为实现对不同对象的调用，引入了 Hashtable 类型的变量 servicetable，作为 Hashtable 类型的变量，servicetable 允许保存键（key）—值（value）对的链表，在此键为服务名，值为对该服务的具体实现。最后是 getremoteMethod 方法的实现，即先利用远程对象的服务名从 servicetable 表中查出实现该服务的对象，再通过方法名和方法参数数据类型数组获取要调用的方法，最后调用该方法，返回结果。

3　结束语

在移动代理系统中使用中间件技术，一方面有利于 agent 在异构平台上的交互，另一方面可以为移动代理的开发这带来更为灵活的机制，扩大 Mobile agent 的适用范围。本文目前仅实现了 Mobile agent 作为客户端来调用 RMI 远程服务器，并为此设计了通用远程接口，在今后的工作当中，希望进一步研究 Mobile agent 作为 RMI 远程服务器的实现，以及由此带来的服务程序定位与安全问题。

参考文献

1　Baumann J,Hohl F,Rothermel K. Mole-concepts of a Mobile agent system. WWW Journal,1998,1(3):123～127.

2　朱淼良,邱　瑜. 移动代理系统综述. 计算机研究与发展,2001,38(1):16～25.

3　Bellavista P,Corradi A,Stefanelli C. Mobile agent middleware for Mobile computing. The World Wide Web Journal,2001,1(3):73～81.

4　易　平,陈福生,李　　. 三种主流中间件之比较. 计算机应用与软件,2002,19(12):52～54.

5　Wollrath A,Waldo J. RMI. Http://java.sun.com/docs/tutorial/rmi/index.html. 2003.

6　Cay S,Horstmann,Gary Cornell. Java 2 核心技术(卷 2:高级特性). 朱　志等译. 北京:机械工业出版社,2000.11.

7　Mark Wutka. Java 2 企业版实用全书. 北京:电子工业出版社,2001.

(责任编辑:邓大玉　曾蔚茹)