一种 CORBA/DCOM 互连网关的设计

赵颖 潘理 沈金龙 (南京邮电学院计算机科学与技术系 210003)

摘要:CORBA 和 DCOM 是当今两种较为成熟的面向对象的构件技术,然而作为两种不同的技术标准,其相互之间并不兼容。因此,如何实现两种系统之间的互通,是众多软件开发人员所关心的问题。本文从两种技术的内部运作机制入手,讨论了两者互通的可能性,并提出了一种合理的 CORBA/DCOM 互连网关的设计思路和方法。

关键词:CORBA DCOM 对象构件 网关 面向对象

公共对象请求代理体系结构 CORBA (Common Object Request Broker Architecture)是 OMG 于 1991 年提出的一个工业标准,它将面向对象技术和网络通信技术有机结合起来,用于在分布式环境下实现应用的集成,使得基于对象的软件构件可以在分布式环境中可重用、可移植和可互操作。随着人们对构件技术的关注,CORBA 技术已日趋成熟,并得到了世界范围内许多计算机厂商和科研部门的产品支持。同时,Microsoft 公司于 1993 年提出了它的一种分布式对象技术——构件对象模型 COM (Component Object Model),现已进一步改进为分布式构件对象模型 DCOM (Distributed COM)。因为 Windows 系列及其应用产品的普及,DCOM 技术也得到了长足的发展。这两种技术之间存在着许多类似之处,但彼此并不兼容。因此,如何实现 CORBA /DCOM 之间的互操作,是众多软件开发人员和用户所关心的一个关键问题。下面我们将提出一种 CORBA/DCOM 互连网关实现的基本思路和方法。

1. CORBA /DCOM 互连网关的设计基础

OMG 组织为 ORB(Object Management Architecture)制定了 CORBA 规范,其主要内容包括 ORB 核心、IDL (Interface Definition Language)的定义和映射、ORB 的体系结构以及 ORB 间互操作的机制。其中,IDL 语言是 CORBA 规范中用于描述对象界面的中性语言,CORBA 规范制定了 IDL 语言与多种具体编程语言的映射关系(单一的 ORB 体系结构如图 1 所示)。

CORBA 支持客户动态或静态地调用远程对象所封装的方法。静态调用方式,即客户采用的对象构件明确定义了调用接口,客户运行的应用程序可以直接访问到某远程对象及其方法。具体来说,静态调用主要利用两端的 IDL 接头(Stub)和 IDL 框架(Skeleton)机制,当客户在某一具体的进程中调用某一远程对象,实际上只是启动了客户端的 IDL 接头。IDL 接头可以看作是远程对象在本地机里的一个代理,它和客户端的 ORB 一起将客户的请求从具体的程式代码转换为可传输 IDL 的形式。当该形式的请求到达对象一方,远程对象服务器的 ORB 和 IDL 框架一同将请求再次转化为具体的编程代码形式,为远程对象实现部分识别。当对象完成请求,响应的回送采用同样的机制。由此可见,IDL 接口和 IDL 框架主要是用于将两端的具体的编程代码转化为底层的 ORB 可识别的形式。

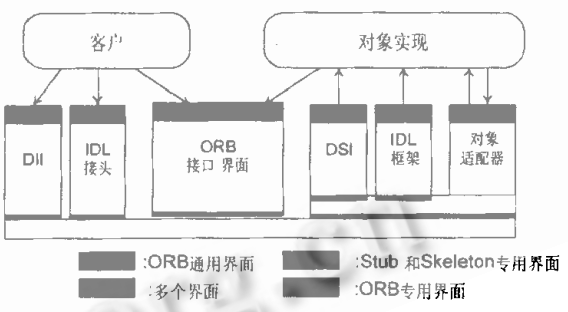

图 1 ORB 的体系结构

动态调用方式,则是客户的应用程序在构造和生成的时刻并未确切规定或调用远程对象,而是在运行期间,根据当时的需求条件,通过 ORB 动态地定向所需访问的对象和途径。具体地说,CORBA 提供了两种界面来进行动态请求:在客户端是动态调用界面 DII(Dynamic Invocation Interface)以及远程对象服务器一端的动态框架界面 DSI(Dynamic Skeleton Interface),DII 和 DSI 都是由 ORB 直接提供的与对象的 IDL 无关的界面。

CORBA 的动态请求机制为我们建立 CORBA/DCOM 互连网关提供了可能性。

同时,COM 规范制定了一组规范(COM 核心、结构化存储、统一数据传输、智能命名)和系统级的实现(COM 库)。COM 核心定义了构件对象与客户通过二进

制接口标准进行交互的规格，结构化存储定义了复合文档的存储格式以及创建文档的接口。统一数据传输定义了构件之间数据交换的标准接口。智能命名给予对象一个系统可识别的唯一标识。

在 COM 系统中，客户对构件对象功能的访问通过调用构件接口函数完成。在实际使用中，接口的定义多采用 COM IDL 来描述。COM 定义两类服务器：进程内服务器(In-process)和进程外服务器(Out of process)。

进程内服务器即本地机上的 DLL。进程外服务器分为两类，一是本地机上的 EXE 可执行程序；二是远程机上的 DLL 或 EXE 程序。服务器内部包括构件接口的实现体和类工厂，类工厂生产构件对象，将对象的接口指针返回给客户。COM 库完成构件服务器的定位并返回对象指针。COM 对象位置的透明性处理由 COM 的服务控制机制 SCM 完成。进程外的对象必须先调用 SCM 提供的 Proxy，Proxy 生成服务对象的 RPC(COM 的系统运作图如图 2)。

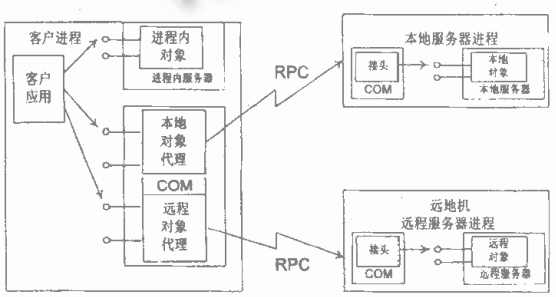

图 2　COM 的系统运作图

DCOM 对 COM 网络技术进行改进，它支持多种通信协议，为 Java 提供分布能力，同时加强了构件通信的安全保障，把基于认证 Internet 安全机制同基于 Windows NT 的 C2 级安全机制集成在一起。但从系统内部的实现机制而言，DCOM 所采用的技术仍符合图 2 的 COM 的模式。然而，从设计互连网关的角度来看，使用 DCOM 技术更灵活和方便。

2. CORBA/DCOM 互连网关的设计

我们需要在 CORBA 和 DCOM 之间设计一种机制，能够实现 CORBA、DCOM 对象之间的互相访问，这就是 CORBA/DCOM 互连网关。网关首先使 DCOM 和 CORBA 中的请求相互识别，一个 DCOM 客户发出请求，DCOM 可以发现它请求的对象不在本系统内，从而将该请求转发给网关，将其转化为 ORB 可以识别的形式，发往相应的 ORB。反之亦然。同时，客户对对象的请求是动态的，要实现两种系统的互通，必须保证网关对动态请求的支持。CORBA 支持客户对象的静态和基于 DII 和 DSI 的动态访问方式。在 DCOM 中，客户和对象之间的交互是通过二进制的接口来进行的，也是一种动态的访问方式。由此可见，CORBA/DCOM 网关无需对请求响应做太多的工作。当完成了相应请求的转换以后，对象服务者动态地处理客户的请求，客户对象无需进行任何改动和编译。所以设计 CORBA/DCOM 的互连网关中最关键的问题是解决请求的转换，从两种系统的体系结构来看，请求转换的实现应是在 DCOM 和 ORB 之间设计一个 IDL 转换器。整个转换的实现结构如图 3 所示：

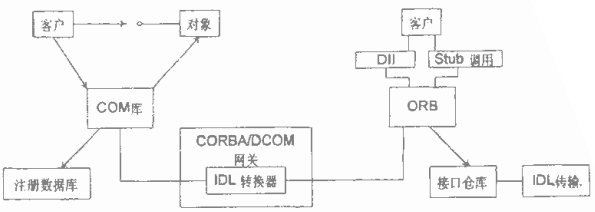

图 3　CORBA/DCOM 互连网关的实现结构

COM 中的客户启用 CORBA 中的对象，向 COM 库发出“创建构件对象”的请求，COM 库查找注册数据库，若在注册数据库中无法查找到所需的构件服务器，则认为该请求是发往一个 CORBA 对象的，它就被发往 CORBA/DCOM 网关，两种 IDL 语言的转换器将 COM 请求转化为 CORBA 请求经 ORB 发往 CORBA 远程对象，远程对象服务器的动态框架界面允许远程对象接收没有经过框架转换的动态请求。

CORBA 客户利用 DII 机制请求 COM 中的对象，由 CORBA::Object 接口提供的 create－Request 操作来创建 Request 伪对象。所有的 IDL 接口都是从 CORBA::Object 继承而来，每个对象都支持 create－Request 操作。客户对象利用该 Request 伪对象向目标对象发送参数，该请求所需的参数类型及返回值类型由 ORB 启动接口库查找。CORBA/DCOM 将该请求转换为 DCOM 中可识别的请求，完成在 CORBA 客户和 DCOM 对象之间建立连接。

考虑面向对象系统的特性，因此在设计该网关程序的时候，利用 OOA 技术，在网关系统内部构造对象，用于完成 COM IDL 和 OMG IDL 的映射。该网关的实现所要做的工作集中在设计一个新的转换器对象，可以为两

种系统所识别。

DCOM 的实现包括一个系统级的 COM 库，因此在新的转换器对象生成以后，还要完成 COM 库对该对象的注册。可以把该对象看作是一种异构系统中被访问对象的代理，客户通过网关将其对异构系统中对象的访问请求交给该转换器对象，由它完成用户与对象之间的连接。至于转换器对象内部对两种 IDL 语言的转换的实现，因为两种 IDL 语言都存在着与各种编程语言的映射关系，实现两者的转换是可行的，且与平台无关。

3. CORBA/DCOM 互连网关的功能

设计 CORBA/DCOM 互连网关的主要目的就是实现两个系统内对象之间相互透明的访问。另外，该网关是建立在分布式环境下应用级的应用，如何协调好与系统以及下层网络协议的关系，也是构造网关是需要考虑的问题。因此，网关还具有安全认证、出错处理等一些辅助功能。

(1)网关提供两种形式访问：静态访问和动态访问。前者是指用户在访问前已经明确知道被访问对象不在本系统中，因此在应用开发中直接将请求发往网关，由网关对请求进行转换并将请求转发给相应的对象。比如 DCOM 中的客户启用 CORBA 中的对象，向 COM 库发出“创建构件对象”的请求，COM 库查找注册数据库，若在注册数据库中无法查找到所需的构件服务器，则认为该请求是发往一个 CORBA 对象的，它就被发往 CORBA/DCOM 网关，两种 IDL 语言的转换器将 DCOM 请求转化为 CORBA 请求经 ORB 发往 CORBA 远程对象。对于这种形式的访问，网关所做的工作是实现两种系统 IDL 接口的映射。CORBA 2.1 中已经给出了这两种语言具体的对应关系。当两系统间采用动态的访问方式时，网关中存在着 COM IDL 与 OMG DII(DSI)之间的映射关系。CORBA 客户利用 DII 机制请求 COM 中的对象，由 CORBA::Object 接口提供的 create - Request 操作来创建 Request 伪对象。所有的 IDL 接口都是从 CORBA::Object 继承而来，每个对象都支持 create - Request 操作。客户对象利用该 Request 伪对象向目标对象发送参数，该请求所需的参数类型及返回值类型由 ORB 启动接口库查找。CORBA/DCOM 网关将该请求转换为 DCOM 中可识别的请求，完成在 CORBA 客户和 DCOM 对象之间建立连接。利用这种动态方式访问速度更快一些，并且便于访问一些未知的对象。缺点是 DII(DSI)接口并不是与语言无关的，因此映射也不是与语言无关的。

(2)DCOM 对 COM 技术进行改进，加强了构件通信的安全保障，把基于认证 Internet 安全机制同基于 Windows NT 的 C2 级安全机制集成在一起。而 CORBA 系统对构件的安全性考虑较欠缺。该网关的设计参考了 DCOM 系统的安全认证的方法，采用集中管理的机制，对于 DCOM 一方的安全认证交由 Windows NT 的系统级认证来完成，而 CORBA 一方的安全认证由网关集中管理。网关根据 DCOM 发送的请求中的用户姓名和权限检索一个认证数据库来对用户的权限进行检查和验证。利用这种机制进行安全认证，既减少了对象对请求权限的处理，又保证了与已有对象构件的兼容。

(3)在系统的健壮性方面，DCOM 使用了 DeltaPing 方法。即客户对象每隔一点时间向对象发一 Ping 信息，当对象在一定时间内无法收到该消息时就认为客户已经退出，于是中断与客户的通信。另外，对于通信过程中出现的各种异常情况，CORBA 系统中有一种异常机制将通信过程中的错误封装在 Enviroment 对象中，DCOM 中采用 HRESULT 的值来判定通信质量。在两个系统中增加了网关以后，两个系统对象之间的通信所需时间增长，可能会出现对象方对请求尚未作出响应，而用户认为已经超时而中断连接，这种异常中断对象是无法知晓的。为了保证整个系统的健壮性，网关必须能够及时对两端对象的状况进行了解，比如接收客户的 Ping 信息，向对象发送客户的状态。同时我们在网关中加入一些缓冲区，对客户的多个请求进行缓存，以减少客户传送请求的次数。

CORBA/DCOM 互连网关完成的功能较多，需要考虑的情况较复杂，对于网关内部的事务处理、并发执行等功能这里不再作过多的讨论。

CORBA 和 DCOM 的目标都是建立一种开放的、基于客户/服务器模式的面向对象的分布式处理系统。实现两种系统的互通，不仅可以充分利用两者各自的技术优势，而且可以给软件开发者提供更大的灵活性，为建立一种更广意义上的分布式对象系统提供保障。

参考文献

[1] Common Object Request Broker Architecture, version 2.0, http://www.omg.org.

[2] The COM Specification, http://www.microsoft.com..

[3] Steve Vinoski, CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments, IEEEE Communications Magazine, February 1997, p46 - 55.

[4] A. S. Tanenbaum, Computer Networks, 3rd Ed, Prentice Hall Inc., 1996 .

(来稿时间：1998 年 3 月)