一种增强 CORBA/ORB 消息通信机制的研究与实现*)

Research and Implementation of an Enhanced CORBA/ORB Message Communication Mechanism

陈 松 王敏毅 周明天

(电子科技大学计算机科学与工程学院 成都610054)

Abstract CORBA is the current popular interoperability technology which makes the development of the distributed object application more convenient and rapid. As we well know, Message-Oriented Middleware assures the reliability, high-effectiveness, security, and real-time quality in the message transmission and supports the "stop-and-forward" of the message. This article presents a communication mechanism conforming to the CORBA specification, which can embed the MOM into ORB, complete core migration from MOM to Object-Oriented middleware, and compatible of the traditional applications.

Keywords Message-Oriented Middleware, Distributed Object, CORBA

1 前言

目前传统中间件在技术上已基本发展成熟,在国内外占有最大的市场份额。但传统中间件是一种以"过程"为基础的编程技术。新一代面向对象中间件在符合 CORBA 规范2.X的前提下,提出了一种"构件化"的思想,它保证不同产品相互之间能够实现互操作[1,2]。为满足新的面向对象应用的需要,同时保持向前兼容传统中间件的应用,本文提出了一种全新的思路和实现,将消息中间件作为 ORB 的底层通信协议,即保持应用不变,"核心"向前推移,提供给用户一个高性能的分布式计算平台。

2 常规 ORB 的消息处理流程

ORB 对应用程序完全屏蔽了网络实现的细节,其中用 CDR 格式来封装消息,用 GIOP 协议完成消息的发送和接收。作为通用的互操作协议,GIOP 并不是一个可直接用于 ORB 之间进行通信的具体网络协议。GIOP 描述了客户和服务器之间进行通信所需的大部分协议细节,其中包括对各种具体传输协议所抽象出的通用消息传送接口。例如,基于 TCP/IP 的 IIOP[1,2]就是 GIOP 的一种具体实现。图1为常规 ORB 规范所遵循的标准通信模型。

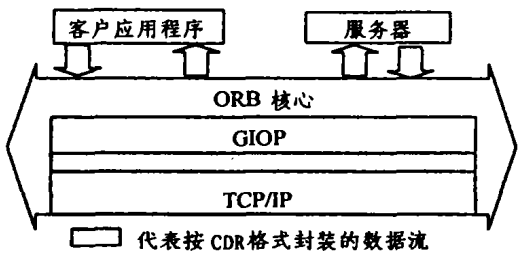

图1 常规 ORB 客户消息发送流程

IIOP 指定了如何根据具体的网络协议进行编码和解码 IOR,对于 IIOP 而言,定位一个对象的主要信息有端点(IP 地址,端口号)、接口池 ID、对象键。这样客户端得到服务对象的 IOR,就可以建立一个到服务端的连接来发送消息,并引发服务对象完成请求和作出相应的应答。

在实际应用中,常需要根据具体的通信协议定制不同的协议构件[4],来设计一个同步[5]或异步[6]的 ORB 核心体系结构。中间件虽不是通信协议,但和传输密切相关并符合某些前提的中间件,如消息和安全中间件,可以看作是一种特殊的通信协议,来完成 ORB 的底层通信。因此可以按照异步的方式设计一个可插御多通信协议的扩展 ORB。

3 消息中间件满足 GIOP 所做的几点假设

消息中间件在消息传递方面具备特殊的能力,能够很好地加强消息的高效、可靠的传递和接收;同时消息中间件还能够实现"存储-转发"机制,实现与时间无关消息的发送和接收,因此具有广阔的应用范围与前景。为此,我们遵循 GIOP 协议的传输映射机制,将其作为 ORB 的底层通信设施,来提高 ORB 在消息传送方面的能力,最终实现应用程序对消息的有效控制和对传送的可靠管理。消息中间件的核心部分由消息队列[3]实现,创建异步、单向的相互连接,并满足 GIOP 抽象协议对传输层所作出的几点假设:①传输是面向连接的(应用和中间件之间创建逻辑连接);②传输是可靠的;③传输是全双工的;④传输提供了一个字节流抽象;⑤传输当连接发生混乱时,提供一些合理的提示。

4 设计思想

根据 GIOP 提出的传输映射机制和 ORB 所提供的动态协议槽接口,我们可以扩展 ORB 所提供的基于 TCP/IP 的标准通信方式,加入自己定义的协议构件[5]来实现 GIOP 协议。图2为扩展的 ORB 客户消息发送流程。由于每种通信协议的特性参数各不相同,系统可以选择最优的通信协议进行通信。

利用消息中间件实现 ORB 的传输功能,关键是实现从 GIOP 到消息中间件的映射(如图3所示)。用 OO 设计实现时,可采用若干抽象基类定义服务原语的接口。具体实现协议则为抽象映射协议在具体网络协议上的具体实现,表现为上

*)本文得到四川省重点科技攻关项目(SG95.17.1)和电科院预研基金项目(DJ8.1.3)的支持。陈 松 硕士,主要研究领域为面向对象技术,Web 技术;王敏毅 博士研究生,主要研究领域为面向对象技术,智能 Agent,网络管理等;周明天 教授,博士生导师,主要研究领域为计算机网络,分布对象技术,并行处理等。

述抽象类的子类，针对每种协议都有相应的若干子类实现，构成相对独立的模块。并且，此模块是可以动态绑定到系统核心的，核心相应有专门的协议槽。

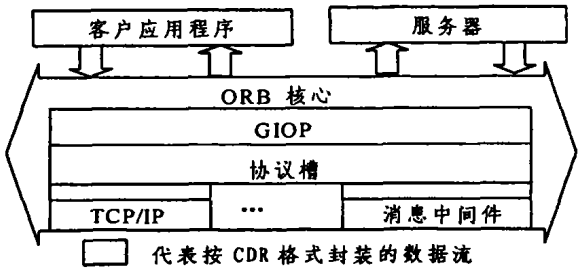

图2 扩展的 ORB 客户消息发送流程

·GIOP 层 CORBA 规范中定义了 GIOP 协议来保证 ORB 的互操作性，其中包括传输的语义和消息语法。GIOP 能够支持象满足传输假设前提下的任何面向连接协议的映射。而下层具体实现协议的实现细节对 GIOP 层透明，GIOP 只使用抽象协议接口完成其功能。

·协议槽 为能够实现对多种网络协议的动态加载，以适应各种不同的网络应用环境，在 ORB 核心中包括专门的协议槽，可以绑定和具体协议相关的通信设施。针对网络通信协议的不同特点，在具体的环境中通过选择算法选择最优的通信协议进行通信。

·Wrapper（对消息中间件的封装） 由于不能直接实现从 GIOP 层到传输层的映射，也不能直接将协议模块直接加入到核心协议槽，因此必须进行额外封装，才能够实现上述机制。对 IIOP 而言，就是 TCPWrapper，而对我们所封装的消息中间件，就是 TMiddlewareWrapper。

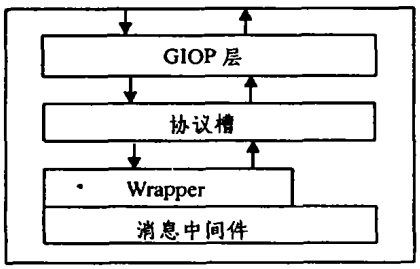

图3 GIOP 和消息中间件的消息映射

5 系统实现

以上阐述了如何在 ORB 核心支持消息中间件的思路，本节我们将具体地从动态协议槽、协议标记映像和 Wrapper 三个方面来介绍具体的系统实现。

5.1 可动态加载核心协议槽

协议槽不属于 GIOP 协议，也不属于下层消息传输层，它是位于 ORB 核心中的一个构件插槽，完成核心对底层通信协议（如 IIOP 协议）的登记。由核心的通信协议选择算法产生默认的通信协议，即最优的通信协议，在对象请求引发时，用默认的通信协议完成客户端与服务器的通信。

在协议槽中登记了一些和具体协议无关的逻辑元素，这些元素起着从 GIOP 抽象协议到具体实现协议之间桥梁的作用。图4就表明了这几种逻辑元素所处的位置以及它们之间的关系。

倾听器：服务端 ORB 一开始运行，守护线程开始启动，按照并发服务器模式，监听客户所发起的各种请求，并进行请求分派。

端点：标记目标程序的唯一地址（如 IP 地址、端口号），连接请求通常针对某个端点。

连接工厂：用于客户端创建连接，至于其所创建的细节均被封装在连接工厂中。

映像工厂：根据标准标记映像，产生协议的标记映像，即将服务端的对象地址信息从网络形态转变为能被客户用以创建连接的主机形态。

连接：即借助下层通信设施实现通信的逻辑实体。

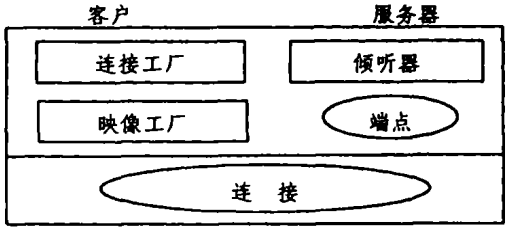

图4 协议槽的几种逻辑元素

以上所定义的几种逻辑元素中，倾听器在服务端的 ORB 核心协议槽中登记，每个倾听器对应一个端点，连接工厂和映像工厂在客户端的核心协议槽中登记，而连接只是由以上几种元素所产生。

```
TList<TOutgoingConnFactory *,TOutgoingConnFactory *>
    connFactoryList_;//协议槽所登记的连接工厂
TList<TCommListener *, TCommListener *>
    listeners_;//协议槽所登记的倾听器
TList<TProfileFactory *,TProfileFactory *>
    profileFactories_;//协议槽所登记的映像工厂
```

5.2 互操作协议（IOP）

对象互操作表达了一组对象在完成某一任务时的动态协作关系，对象互操作的行为描述与抽象是支持面向应用对象互操作的基础[7]。下面是 CORBA 规范所规定的标准标记映像格式及其它有关互操作定义：

```
typedef unsigned long    ProfileId;
const ProfileId          TAG_INTERNET_IOP=0;
const ProfileId          TAG_MULTIPLE_COMPONENTS=1;
struct TaggedProfile{
        ProfileId        tag;
        Sequence<octet>  profile_data;
};
```

对象引用至少含有一个标记映像。标记映像中可以包含传输协议及定位对象所需的信息。如 IIOP 的 IP 地址、端口号、对象键，封装为字节流形式的方式。任一个单一的标记映像容纳足够的信息来完成从客户端到服务端对象的一个完整引发。每一个标记映像有一个唯一的数字标记，对传输协议而言则指明该标记映像所支持的具体协议类型，进而可知道该按照何种方式对协议进行编码和解码。

针对消息传输的标记映像，标识独立的可通过消息传输层访问的对象。我们定义以下的协议体：

```
const ProfileId  TAG_MIDDLEWARE_IOP = 0x1000;
struct TMiddlewareProfileBody{
    struct hostaddress addr;
    Sequence<octet> object_key;
}
```

addr 封装了表示服务程序的地址信息。对不同的消息中间件来说，地址信息的表示都不相同。

objkey 在 OA 中唯一地标识了一个对象，描述为一个对

象的标记。服务端接收到请求以后，根据此标记找到对应的对象实例。

一个 TMiddlewareProfileBody 实例可以编码成一个字节流的封装。这个封装也就是对应于前面所定义的 TaggedProfile 的成员变量 profile_data，并且此 TaggedProfile 的成员变量 tag 为 TAG_TMIDDLEWARE_IOP，客户根据此标记区分服务程序的地址信息。

5.3 Wrapper(对消息中间件的封装)

消息中间件是基于 C 语言实现的，为实现从 GIOP 到具体传输接口的映射，核心如虚连接的建立、消息的发送和接收以及具体协议相关的地址定义等，我们用面向对象语言对消息中间件进行封装。这样可以向 GIOP 层和核心提供一个公共的调用接口，从而最终支持协议槽规定的语义。

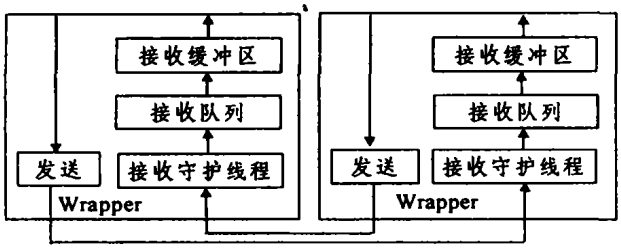

图5 Wrapper 的内部实现

(1)多线程封装 由于 ORB 是多线程的，因此对消息的发送和处理都应该提供对多线程的支持。在消息中间件的消息接收前，加入了一个接收队列，能够区分各自不同的线程号，最后再进行消息的接收和处理。

(2)端点和连接的封装 在这里主要将消息中间件的地址信息封装为被 ORB 中其它逻辑元素所能够识别的实体，向映像工厂、连接工厂、倾听器提供服务程序的地址信息。消息中间件需要定义一条虚连接，提供字节流消息的发送和接收。因此在连接的创建前，就应该完成和连接相关的消息队列的初始化，如消息中间件核心的启动等。

(3)消息的封装 CORBA 客户的请求必须按照 CDR 数据流格式进行封装，才能在网上进行传输和在不同的操作系统和不同的 ORB 之间进行通信。对消息的封装是在 GIOP 层完成的，在消息传输层只是需要对消息包进行传递。但并不是说消息是不能够改变的，如果消息需要特殊处理，如安全加密，可以再对消息进行进一步的处理，并由对等通信实体来对消息进行逆操作。

结束语 常规的 ORB 缺乏强大的消息处理能力和消息的存储转发能力，而消息中间件在这方面有足够的支持，我们提出将消息中间件嵌入到 ORB 中，以弥补后者在这方面的不足。在这样的 ORB 中，基于 GIOP 协议的映射方式和动态协议槽，可以扩展 ORB 常规的通信能力，支持定制的协议构件。我们通过协议槽、消息中间件的 Wrapper 和互操作协议实现的消息传输协议构件，既可以支持面向对象的应用开发，又可以向前兼容传统中间件的应用，为相同平台不同 ORB 之间和不同平台 ORB 之间的消息传递提供一个良好的通信环境。所开发出的 S/W 产品，已经在省级范围的电力系统中得到成功的应用，效果良好。

参考文献

1 Object Management Group. The Common Request Broker: Architecture and Specification, Revision2.4, Oct. 2000

2 Vinoski S. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. IEEE Communications Magazine, 1997, 14(Feb.)

3 Gokhale B N A, Schmidt S Y D C. Applying Patterns to Improve the Performance of Fault Tolerant CORBA. In: The 7 Intl. Conf. on High Performance Computing, ACM/IEEE, Bangalore, India, Dec. 2000. 17～20

4 O'Ryan C, et al. The Design and Performance of a Pluggable Protocols Framework for Real-time Distributed Object Computing Middleware. In: Proc. of the Middleware 2000 Conf. ACM/IFIP, Apr. 2000

5 Schmidt D C, et al. Software Architectures for Reducing Priority In-version and Non-determinism in Real-time Object Request Brokers. Journal of Real-time Systems, special issue on Real-time Computing in the Age of the Web and the Internet, To appear 2000

6 Arulanthu A B, et al. The Design and Performance of a Scalable ORB Architecture for CORBA Asynchronous Messaging. In: Proc. of the Middleware 2000 Conf. ACM/IFIP, Apr. 2000

7 Li Gui, Yin Chao-Wan, Zheng Huai-Yuan. A Behavior Model of Object Interoperability. The Chinese Journal of Computers, 1999 (5)

(上接第107页)

需要指出的是，这里是按行处理优先来说明的，如果考虑到旋转、左右消失点变化等情形，则优先顺序是变化的。如图5，当旋转至(b)方位时，就应以列优先，而且列内部各方格的次序也与原来相反。

跟上一个算法一样，对此算法也在串行机 HP 工作站上进行了假定处理机台数为4的模拟试验，实现的加速 $S_{p=4}$ 约为3.8，效率 $E_{p=4}$ 约为0.95。

结论 本文初步探讨了并行算法在三维景观模型可视化方面的应用。并行算法是可视化系统的加速工具，通过它，可以加快可视化系统的速度，改善可视化效果。在设计并行算法的过程中，必须考虑 CPU 的个数、存储方式等具体情况以及各 CPU 之间的负载平衡和网络通讯的时间耗费。在网络通讯中采用合适的压缩和解压缩技术，不仅能够缩短通讯时间，还可以将解压缩过程与结果合并过程融合到一块，进而改善整个算法的效率。

参考文献

1 Watt A, Watt M. Advanced Animation and Rendering Techniques —Theory and Practice, 1993

2 沈志宇，等. 并行编译方法. 北京：国防工业出版社，2000

3 Wood J. Visualization Contour Interpolation Accuracy in DEMs. In: Hilary M, H, et al., eds, Visualization in GIS, 1994

4 章孝灿，等. 遥感数字图像处理. 杭州：浙江大学出版社，1997

5 李军，李德仁. 分布式遥感图像处理中的若干关键技术. 武汉测绘科技大学学报，1999，24(1)

6 沈志宇，廖湘科，胡子昂. 并行程序设计. 长沙：国防科技大学出版社，1997

7 全惠云，高汉平，康立山，陈毓屏. 并行计算机程序设计导论. 武汉：武汉大学出版社，1998

8 邓俊辉，等. 基于虚拟机的并行体绘制. 软件学报，2000，11(8)

9 李晓明. 数据并行计算概念、模型与系统. 计算机科学，2000，27(6)

10 胡友元，黄杏元. 计算机地图制图. 北京：测绘出版社，1987