第7章 消息接发

在本书中，主要是从客户机激发远程方法的角度来讨论 CORBA系统。特别是，我们假定客户机和服务器组件按TCP/IP连接，而且服务器主动监听从客户机来的请求。当收到一个请求后，服务器处理这个请求，并把结果返回给发出请求的客户机。客户机在等待回应时是阻塞的，只有在它接收到回答后才能继续处理。即假定激发是同步的，而且是紧耦合的。更形式地说，隐含的用于远程方法激发的CORBA模型有以下特征：

- 同步方法激发　标准的CORBA远程方法激发是同步执行的 —— 即客户机在等待激发的回答时是阻塞的。
- 紧耦合的组件　客户机激发一个特定服务器中的特定对象，而该服务器又在特定的主机中。客户端的代理指向一个（而且只能一个）目标对象。如果目标主机或服务器不能接触，或是目标对象不存在，客户机就会接到一个异常。

在很多场合中，这种同步和紧耦合行为正好是用户所希望的。我们的逻辑经常指出客户机需要阻塞，直到接收到请求的回答后，才能继续处理。除非结果是立即需要，否则为什么要先发出请求呢？很多时候都需要客户机和服务器间的紧耦合。

7.1 CORBA和消息接发

在某些场合中，应用程序可从不同的通信模型中得到好处。许多系统都有基本的异步特征，并经常要求一非耦合的交互方式。例如， StockWatch实例研究中的股票价格更新部分就是这种系统的一个例子。

这种通信类型称为消息接发。本书中，我们不太严格地把消息接发定义为应用程序组件间的非耦合、异步通信。这些组件是非耦合的，是指它们之间没有直接的连接。它们的通信要使用无连接的多目网络协议或通过中间组件来进行。它们的通信是异步的，即是指调用者在向接收者发出消息后保持不阻塞。它们可继续处理，而不用等待接收者完全处理好消息。消息接发另外的特征还包括潜在的一对多和多对多通信。

前面已经提过，我们的 CORBA系统不使用消息接发来进行通信。但是用 CORBA来实现异步的非耦合通信也是可能的，下面探讨由 ORB提供的对此特征的支持。在本章后面会讨论相关的CORBA服务。

7.2 ORB对消息接发的支持

CORBA规范给出了一些对消息接发所需的支持。 ORB被要求在一定程度上支持异步通信，这是通过 IDL关键字“ 单向 ”的使用以及使用动态激发接口时的延迟同步激发来实现的。

7.2.1 异步单向激发

一个对象上的同步远程方法激发暗示着调用程序是阻塞的，直到服务器处理完该方法并返回结果。而事实则更复杂：除非这个调用在 IDL中被定义为单向的，否则 CORBA方法激发会导致调用线程阻塞，直到结果返回。回顾 IDL支持单向操作的概念，它是由客户机异步发送的，即客户机发出一单向请求，然后继续处理，而不用等待服务器处理该请求[⊖]。单向请求虽然必须遵守某些限制（没有返回值或输出参数），但也是很有用的。下面会讨论单向请求，但首先会讨论为什么同步激发有时候是不需要的。

考虑一个发出阻塞激发的单线程客户机。根据定义，客户机一直阻塞，直到服务器回应激发。这意味着客户机程序在等待回答时不能做任何其他的事情。如果这个程序不和用户交互，这通常就不是问题。但是，如果用户要等待这个回答返回，而且还要花不少的时间，那么在用户看来程序已经挂起了。通常，在可能要长时间运行的请求中提供某种反馈是更可取的，可能还可以给用户一个机会来取消操作[⊜]。

再考虑一个单线程的服务器，它有时候还作为客户机。当服务器正忙于为一个客户机处理请求时，它就不能为任何其他的客户机执行工作。这种行为可以预料到并被接受（这里只有一应用程序线程，所以这是不可避免的）。但是，当该服务器作为一个客户机并发出请求时，它就会阻塞，以等待回答。在此期间，它不执行任何有用的工作，也不能处理任何到来的请求。这种情况是不可接受的，它会减少系统的吞吐量，甚至会在服务器创建一循环调用图时导致死锁。当然，避免这些问题的一个方法是使组件变为多线程而不是单线程。但是有时我们不能（或不选择）这样做。下面讨论一些方法，以便可以使用 CORBA来改善系统的吞吐量，而用不着多线程。

解决这个问题的一个方法是把标准的阻塞调用分解为一对单向操作，即客户机发出它们的激发，然后继续处理，而用不着在发出一请求后阻塞，直到结果返回。稍后，当服务器完成对该请求的处理后，它通过向客户机发回一相应的单向激发把结果“返回”。下面简要地说明这种方法的IDL示例。

```
// *** Original IDL ***
interface Stock
{
    PriceInfoSeq getHistoricalPrices(in Date startDate,
    in Date endDate);}

// *** Modified IDL ***
interface PriceInfoCallback
{
```

⊖ 虽然这并不够充分准确地指出实际的行为，但却是一真实的声明。由于大多数 ORB使用TCP作为它们底层的传输，而TCP是一可靠的协议，所以即使用户认为是非阻塞的调用，实际上在 TCP层还是阻塞的调用。用户的客户机应用程序可能不等待服务器应用程序接收并开始处理请求，但是客户机的 TCP/IP栈却要等待，直到请求缓冲区被服务器的TCP/IP栈成功接收（和确认）。

⊜ 注意在很多商业可用的ORB中，不可能取消已经在进行的请求。但是，在应用程序层进行取消却是可能的，所以在请求被接收时，它只是简单地被忽略。

```
    oneway void historicalPriceResults(in PriceInfoSeq);};

interface Stock
{
    oneway void getHistoricalPrices(in Date startDate,
    in Date endDate,
    in PriceInfoCallback
    callbackObject);
};
```

客户机程序实现了PriceInfoCallback类，并把一对象引用作为Stock::getHistoricalPrices激发的一部分传递到服务器。服务器处理该请求，并在结果准备好时通过发出对回调对象的historicalPriceResults激发来把结果传递到客户机。通过这个方法，调用的应用程序在服务器忙于查找股票历史价格并制表的期间不必阻塞，客户机在很长的调用期间内有空闲处理用户的输入。这两个方法的交互图如图 7-1和图7-2所示。当服务器作为客户机时，应用类似的方法的好处就更加明显 —— 服务器在此期间有空闲来处理其他客户机发来的请求，因此不需要多线程就能帮助系统增加吞吐量。但是，要注意这个改善也要付出代价：它增加了服务器的复杂性。具体地说，系统需要有打破应用程序逻辑的能力，以便能应付非连接的处理。在一个单线程的应用程序中，发出单向请求，要把当前工作放在一边一段时间，然后在接收到包含结果的相应单向请求后恢复工作，这种能力是很重要的。它取决于应用程序是如何组织的，可能很难实现。

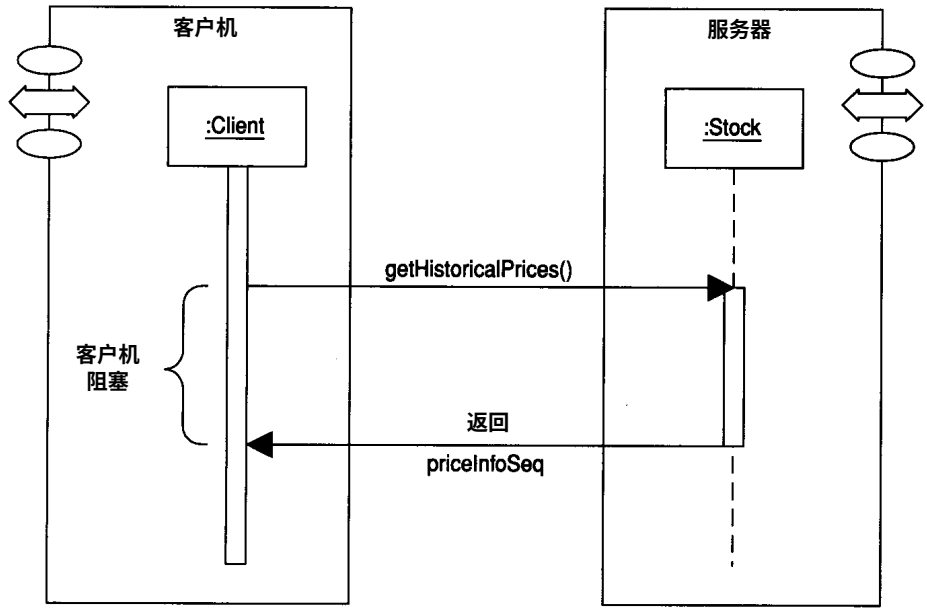

图7-1 同步激发示例

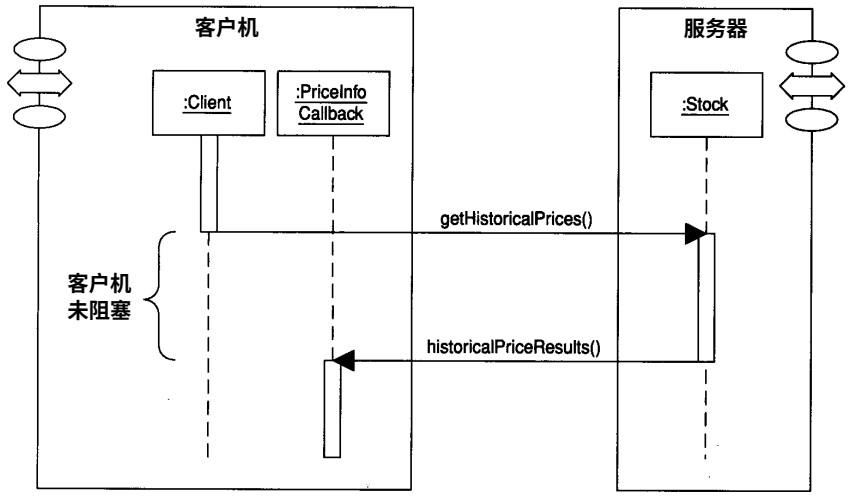

图7-2 异步激发示例

7.2.2 延迟同步激发

通过动态激发接口（dynamic invocation interface, DII）的使用，CORBA支持发出非阻塞调用的另一个方法。要记住DII可用来发出对接口的激发，而客户机程序在编译时并不知道这些接口。把这和正常的静态激发接口（SII）比较，在后者中，客户机程序员只是简单地发出对代理对象的方法激发。由IDL编译器生成的代理对象类负责打包参数、发出远程激发以及解包结果。用了DII，客户机程序员就必须执行这项工作。显然，这是一更复杂的方法，但对于某些应用程序（如网关或浏览器）类来说，独立于任何特定的IDL是很重要的。

DII还支持延迟同步激发的概念。这是无需单向方法施加任何的IDL限制就可发出非阻塞的远程激发的一个方法。即客户机即使在方法激发要返回某些数据（客户机负责在请求发送后的某段时间检查结果）时，也可发出非阻塞的激发，这是很有用的。最后一点在于这个方法只在使用DII方法时可用，因此它对于很多应用程序不是很实用。要注意的一点和单向请求时相似，在使用延迟同步激发时，客户机必须能够把它们的逻辑分解为独立的步骤，以便在不同的时段处理。同样，客户端延迟同步激发的使用对服务器是透明的。

7.2.3 消息接发的实现

ORB所支持的功能是向着消息接发进了一步，使用单向请求或延迟同步激发提供了某种程度的异步通信。但是，还有一些领域是当前的ORB支持不了的。组件仍然需要紧耦合在一起：在客户机和服务器之间必须有直接的通信连接，而且这些组件必须要清楚知道对方。用户在系统中想要的其他消息接发特征包括消息的持久存储（这样客户机和服务器不用同时运行）、一对多通信（这样通过一个激发可把消息传播到多个接收器）、或事务性的消息传递。

为了了解组成消息接发的功能范围，下面简要地探讨商业上可用的面向消息的中间件产品。

这些被广泛接受的系统提供了丰富的特征集合和多样的服务质量（它们的消息传递机制的能力），它们都是解决现实世界业务问题所需要的。稍后，我们会看到 CORBA服务如何提供类似的特征集合。

7.3 面向消息中间件的需求

通常，业务需求要求应用程序能以异步和非耦合的方式通信。这主要是通过归类为面向消息中间件的软件来实现的。这些中间件允许应用程序无需阻塞、无需发送者和接收者之间的直接通信连接就可发送消息，并具有某些传递特征。

7.4 当前面向消息中间件产品

今天，商业上可用的消息中间件产品包括 IBM的MQSeries、Microsoft Message Queue Server、TICBO Inc的ETX和PeerLogic Inc的PIPES。这些产品通常提供异步的消息队列和持久的消息存储，还有各种支持服务，例如队列的事务性接口。

异步消息接发允许应用程序不用阻塞就可把消息加入队列。持久消息存储提供了一种方法，使消息即使在发送者和接收者应用程序没有同时运行的情况下也可传递。可能还有对消息队列的事务性访问。这个工具向应用程序提供了消息会被传递而且是立即传递的保证。

7.5 相关的CORBA服务

CORBA规范的设计者认识到对这些多样化特征和服务质量的需要，于是针对该需求定义了一系列的服务。通过在CORBA框架中使用这些特征，用户能够以标准的方式构建功能丰富的系统。下面，我们会讨论一些当前的和即将出现的 CORBA标准。

7.5.1 CORBA事件服务

CORBA事件服务在前面已作过简短的介绍，它为基于推和拉的消息传递（也称为事件通知）提供一个通用的模型。这个 CORBA服务为非耦合组件间的消息传递定义框架，但没有明确说明任何服务质量方面的内容（这留着由实现供应商决定）。事实上，通信通常是基于标准的CORBA远程方法调用。虽然 CORBA激发是同步和紧耦合的，但 CORBA事件服务的总体结构是非耦合且异步的。这是通过逻辑事件信道的使用来实现的，如图 7-3所示。事件信道，如在CORBA事件服务中的定义，是一个对象，它的工作是把事件从生产商传递到所有感兴趣的顾客。

组件交互通过正常的同步激发发生，但事件信道取消了 StockWatch客户机到StockWatch服务器的耦合。考虑在 StockWatch服务器送IBM股票价格更新时会发生什么。送一个股票价格的更新会阻塞 StockWatch服务器，但这只是暂时的。事件信道几乎是立即返回控制，这样StockWatch服务器就能继续其工作。只有在这以后，事件信道才可开始把股票价格送到所有对接收IBM价格更新感兴趣的 StockWatch客户机（用户）。通过这个方法，事件服务即使在利用同步调用时也可提供异步行为。

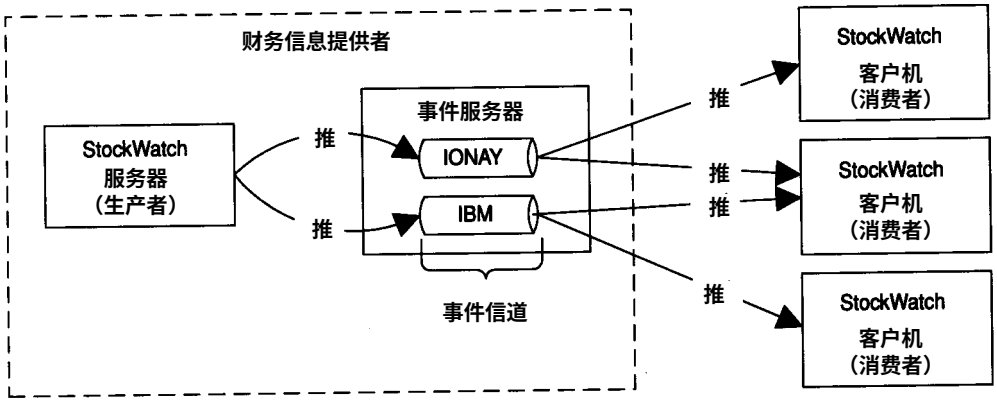

图7-3 事件信道

使用事件服务的关键是事件信道的选择。每个信道把事件从所有的供应者传递到所有的用户，应用程序根据自己的兴趣把自身注册为一个特定信道的供应者或用户，仔细选择信道是很重要的。发送到特定信道的所有事件再由事件服务转向所有订购了该信道的用户。每个信道要为逻辑上相关的事情集合携带事件。在上面的例子中，每个股票代号有一个事件信道，允许客户机只接收它们感兴趣的股票的更新。

另一个可选的配置是只有一个事件信道，通过它来发送所有股票的价格更新。显然，这是一个蹩脚的设计。StockWatch客户机可能在任一时间只对一小部分股票感兴趣。这是一极端的例子，但它说明了重要的一点，即事件信道要有足够细的粒度，这样订购者才会对所有或大部分通过该信道发送的事件感兴趣。

事件信道被实现为CORBA对象，它在事件服务器进程中消耗资源。如果一个设计要调用大量的事件信道，则要保证事件服务实现可支持这一点。

1. 联邦事件服务

假定可能有数以千计的客户要访问 StockWatch系统。在这种情况下，显然从性能、资源使用和容错性的角度来看，一个单独的事件服务器进程不可能为所有这些使用者服务。幸运的是，通过使用一种称为事件信道联邦的技术，事件服务可轻易地被扩展。把一个事件信道注册为另一个生产者的事件信道的消费者，这样的一些事件信道就能够结成联邦。通过这个方法可以生成一个联邦的分布式处理在多个事件服务器中的事件信道树。

CORBA事件服务非常灵活，而且事件信道能以多种方式使用。当确定了如何使用事件信道后，就要确定如何最佳地组织事件信道的联邦。图 7-4给出了两种可能的方法。第一种方法称为完全联邦，每个事件服务器进程包含所有事件信道的实例。在根事件服务器中的每个信道隶属于每个子事件服务器进程中的相应信道。第二种方法是信道联邦，每个根事件信道被提交给一个单独的事件服务器进程。这样就会导致有多少个信道就有多少个子事件服务器进程，每个子事件服务器含有一个单独的事件信道。

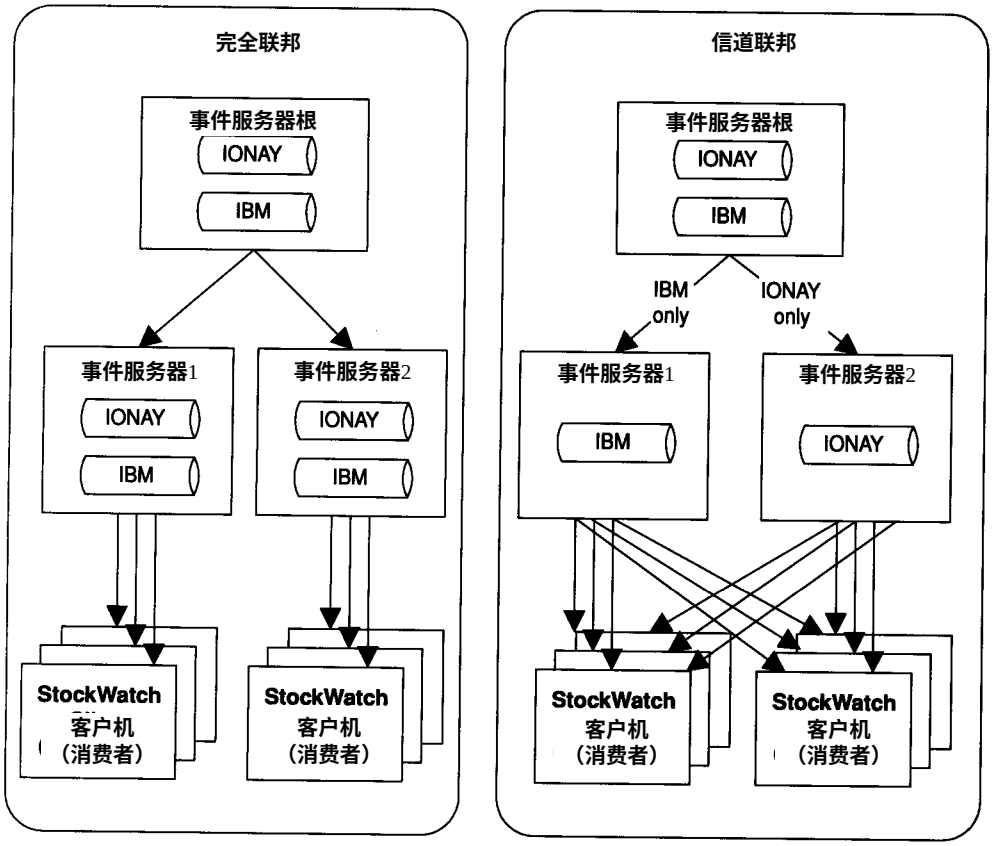

图7-4 事件信道联邦

完全联邦通常是较为可取的方法，这主要是因为每个客户机只和一个单独的事件服务器交互。同时，事件信道被复制到所有的事件服务器上，这样提供了较好的负载平衡。另一方面，信道联邦可能会导致客户机和多个事件服务器连接，并没有消除那些促使我们用联邦要解决的问题。但是，如果客户机只对一个事件信道感兴趣的话，这个方法就很好。

2. 找出事件信道

一个有趣的话题是用户如何找出事件信道。因为事件信道实现为正常的 CORBA对象，所以可以使用前面讨论过的任一解决方案 —— 例如交易服务或命名服务。我们会很容易地想到使用一个名字层次来获取事件信道。但是从系统设计者的角度来看，要小心如何组织层次，所以我们要遵循在联邦图中隐含的规则。例如，如果要使用完全联邦，那么必须要保证 StockWatch客户机只和一单独的事件服务器连接，来取得其中所有的事件信道。如果客户机错误地在多个事件服务器中取得事件信道的引用，那么想要通过完全联邦实现的目标就不会达到。

所以我们需要一些方法来保证客户机总是和一个单独的事件服务器连接。实现此目标的一个方法是把每个事件服务器显示为名字层次中的唯一对象。每个客户机被配置为在它整个生命周期内使用（或随机选择）一个单独的事件服务器。这是一个简单的方法，但却受静态性的约

束，不能对运行期间的变化条件作出反应。一个更透明、而且能让它自己更灵活、更好地动态管理系统的方法，是让客户机通过对某个其他对象的激发来得到一个事件信道的引用。例如可让客户机应用程序激发 Stock对象上的方法，这个方法会返回相关联的事件信道的引用。应用程序服务器在内部要记录哪个客户机使用哪个事件服务器，以保证客户机在每次请求时都能得到相同事件服务器进程中信道的引用。

这种方法也可允许应用程序服务器只使用处理当前负载所需的那么多的事件服务器。如果突然间有更多的客户机要预约股票价格的更新消息，服务器就会生成新的事件服务器，并加到动态增长的池中。这些领域的更深入讨论可在第 15章“负载平衡”和第16章“容错性”中找到。

3. CORBA事件服务的局限

CORBA事件服务向应用程序提供了以非耦合、异步方式通信的方法。事件服务的商业实现多数使用IIOP作为它们的网络协议，这样把这个服务集成到 CORBA系统是很简单的。但是，这个服务仍然有一些局限。

首先，事件服务没有明确说明服务质量。例如，规范中并没有论述事件的持久存储、事件的传递保证或特定时间段内的传递。每个事件服务的实现要支持自身特征的集合。理想情况下，每个实现会以标准的方式支持相同的服务质量。

其次，区别事件的唯一方式是在事件信道层次上。作为应用程序设计者，我们要决定在一特定的事件信道中发送的事件的数量和类型。但是，这并没有提供对事件粒度的区分。信道把所有的事件传递到所有的用户，而不管它们是否真的对这些事件感兴趣。例如，股票价格客户机可能只对监视 IBM的价格变动感兴趣。但是客户机应用程序对所有小的价格变动不感兴趣。在应用程序层次上，用户只对股票价格的显著变动（大于 2%）感兴趣。有了事件服务，这种类型的过滤只能在客户机应用程序中发生，这将导致网络和客户机使用的增加。

第三，服务并不维护关于事件信道的更高级数据。例如，事件供应者想知道是否有顾客对特定类型的事件感兴趣。如果当前没有预约者，供应者可通过发送不需要的事件，来避免浪费网络资源。同样地，事件的用户可能想找出供应者能提供什么类型的服务。开发 CORBA通知服务就是用于说明这些局限。

7.5.2 CORBA通知服务

CORBA通知服务提供了过滤事件、维护关于事件信道的更高级信息和规定服务质量的标准方法，它也是事件服务的自然扩展。在通知服务中定义的每个接口继承了事件服务中的对应接口，并提供了与使用事件服务的应用程序的向后兼容性。

有了通知服务，每个用户都可把一个限制列表和一个事件信道相关联。限制是用 OMG特定的限制语言所写的布尔表达式。如果一个用户注册到一个事件信道，并指定了一个或多个限制，则事件信道会把这些限制应用到每个来到的事件中，以允许事件信道来决定是否把事件转送到每个用户。如果事件已被接收，这就防止了对监听应用程序本该丢弃的事件的传递。

通知服务的另一个特点是管理服务质量方面的能力。服务质量可在不同的粒度层次上规定：每个事件、每个使用者 /生产者或每个事件信道。服务质量的例子是传递保证、超时和优先权。传递保证的范围可从根本没有保证到保证事件在它成功地传递到所有注册使用者之前不会在信

道中被丢弃。超时规定了在事件被丢弃之前允许使用多长时间，而不管事件是否成功传递到所有的使用者。优先权规定了一个事件相对于其他事件的优先权。通知服务还向事件信道提供了事务性的接口，这样事件生产者和使用者可依赖于刚好一次传递的事件。最后，通知服务提供了对服务质量的管理。应用程序可把服务质量需求应用到服务元素，例如事件信道、代理、甚至是单个事件中。

我们预料通知服务的商业实现将被广泛接受，因为它与事件服务的兼容性提供了一个易于迁移的途径，而另外的特征也大大激励了一些机构去引入通知服务。

7.5.3 即将实现CORBA服务

在编写本书时，OMG正致力于开发CORBA消息接发规范。这个标准通过扩展在核心 ORB中对异步通信的支持来说明对消息接发的需要。像异步方法激发（ asynchronous method invocation, AMI）、时间独立激发（time independent invocation, TII）和消息接发服务质量这些特征都包括在内。

AMI提供了两种标准方法来发出返回值的异步激发。客户机既可传递一回调对象（服务器可在请求完成后激发这个回调对象），也可请求一个轮询对象（客户机可通过查询它来得到结果）。这个回调方法和本章前面讨论的单向回调方法类似。

TII提供了一个标准的机制，有了这个机制，客户机和服务器就无需同时运行，它提供了一个标准的存储和转发机制来实现这个目标。

消息接发服务质量（它可应用到同步和异步消息）提供了对带有目标 ORB的同步程度以及服务器端消息处理控制的标准化支持。我们预料只要这些特征被商业上的 ORB支持，开发人员就会很快开始使用它们。

7.6 多目消息接发

到目前为止，客户机和服务器之间的通信是使用 IIOP。这个协议运行在 TCP/IP之上，而TCP/IP则向用户提供了组件间的可靠和基于面向连接的通信连接。但是，还有其他的网络协议可用，它们都适合应用到CORBA环境。在本节中，主要讨论使用 IP多目传送。对网络术语作解释后，我们将检验IONA Technologies的OrbixTalk产品，它使用多目传送来提供组件间的有效、无连接通信。

7.6.1 网络术语

网络今天普遍使用 TCP/IP协议集合来通信。网络协议的基本视图在图 7-5中显示，下面分别介绍每一种协议。

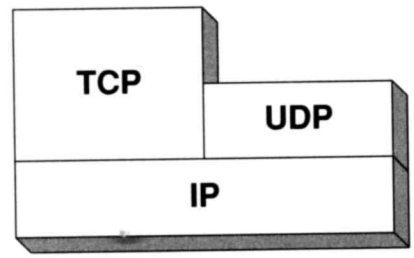

图7-5 基本网络协议

7.6.2 网际协议IP

如我们现在所知道的，网际协议是互联网的基础。它用来把包从一个主机发送到另一个主机。IP是一非连接协议，并不能提供可靠性。这是指发送者不能保证任意给定的包都可被目标

主机接收。虽然，大多数IP包都会到达它们的目的地，但总有可能其中一些不能到达。因此，在IP之上的大多数协议都提供了一个更高级的特征来保证所有消息的传递。

网络上的每个主机都有一个唯一的IP地址（如192.190.230.2）。目标主机的IP地址包含在每个IP包中。在发送包时，这个地址在每个路由中被检查，接着或是直接传送到目标主机，或是转送到另一个路由，这个路由能把包再转送到距离目标更近的点。不同IP地址类的定义决定了在给定网络段内包是传送到一个主机、一些主机还是所有主机：

- 单目传送　发送到一单独的主机。
- 多目传送　发送到一组主机，称为多目组。
- 广播　发送到网络区内的所有主机。

以太网的物理地址指定在一给定网络段内，在任一时间点只能发送一个消息。显然，当系统开始扩大时，这就会有影响。如果用户希望向多个主机发送消息，则单目传送要求用户必须重复发送同样的消息，每次把消息送到一个目标主机。这是用户在基于IIOP的系统中看到的行为。但是，如果用户利用多目传送，就可简单地一次性发送消息，并依赖IP的多目传送特征来把消息传递到每个主机。如果用户要发送很多消息到多个主机，则多目传送方法肯定可以提供比单目传送更好的性能。在对TCP/IP和UDP进行简短讨论之后，将会回到多目传送的讨论。

1. 传输控制协议（TCP）

传输控制协议（TCP）是基于IP的可靠的、面向连接的协议。两个进程之间（可能在不同的主机上）建立了逻辑连接，使得这些进程能以可靠的方式发送和接收数据流。由于TCP是基于不可靠的IP协议，所以它必须提供自己对消息传递的可靠性保证。它通过包计数、确认和超时等技术的结合来实现这一目标。

2. 用户数据报协议（UDP）

由于TCP/IP是面向连接的，所以它只支持两个专用进程间的通信（单目）。要使用广播或多目通信，就需要类似于用户数据报协议的非连接协议。UDP向应用程序层提供了比TCP/IP更简单的服务，而它实际上也是不可靠的，就像底层的IP协议一样。因此，它不提供对消息包会抵达它们预想目的地的保证。

多目请求在同一时间内发送到多目组的所有成员。类似于以太网消息，每个包由子网上的每个网卡接收。但是，对于单目消息，只有目标主机的网络软件会实际执行对消息的处理。其他的所有主机检验该包，并丢弃它，因为消息并不是指向它们的。但是，使用多目传送，就可令多目主机接收并处理同一个网络包。这是通过让进程加入一称为多目组的集合来实现的。这就通知了网络软件除了要监听指向主机的单目地址的包外，还要监听指向某些多目地址的包。随着进程加入和离开该组，在这些多目组中的成员是动态的。因此，多目传送对于大规模系统是一强大而有意义的机制。

7.6.3 单目和多目

如何比较基于IIOP事件服务的单目通信和基于IP的多目产品呢？基于多目的产品的主要优点是高吞吐量和高伸缩性。在基于单目的事件服务中，事件信道必须向每个事件接收者发送分离的消息。对于大量的事件用户，这就会明显地减慢事件的传递。另外，虽然组件间没有直接的

连接，但事件的每个生产者和使用者之间要求有 TCP连接。

多目传送的一个潜在问题是它要求所有加入者由一个网络来连接，而这个网络要能够在子网间为IP多目包寻径。并不是所有的路由器都能在物理上做到这点，如果它们有这个能力，就必须配置为这样做。如果要在用户所控制的领域之外通信（例如在 Internet上），使用基于TCP/IP的通信机制就有意义，因为尝试多目传送会不能抵达它们的目的地。由于多目传送提供了较高的吞吐量，所以对于被控制环境中的事件分布，使用多目传送就有意义，例如 Intranet中，在这里路由策略可以应用到整个网络。

7.6.4 多目传送实例研究：OrbixTalk产品

到目前为止，已经讨论过基于 IIOP的消息接发解决方案，而 IIOP当然又是基于TCP/IP协议。TCP/IP是一面向连接的网络协议，用于单目通信，即消息直接从一个单独的客户机进程发送到一单独的服务器进程（并从单独的服务器回到单独的客户机）。这对于基于IIOP的消息接发有重要的影响，因为它要求客户机和服务器间的直接 TCP连接（这是前面讨论的紧耦合属性）。

在下面一节中，我们将研究一个实际的产品，它使用 IP多目传送来提供CORBA组件间的真正的非耦合通信：IONA Technologies的OrbixTalk。我们会解释这个产品如何通过使用建立在不可靠的UDP之上的可靠协议来克服多目传送的局限。

1. OrbixTalk的可靠多目传送协议

UDP的多目能力使它成为建立高级的消息接发解决方案时值得关注而又有用的协议。当然，UDP和多目传送的缺点之一就在于它是一个不可靠协议。为解决这个问题， IONA Technologies开发了可靠多目传送协议（ reliable multicast protocol, RMP），它位于UDP之上，并应用到OrbixTalk产品。

RMP如何工作？TCP/IP的可靠传输实现的一个重要部分是主动确认。在 TCP层，接收者通过向发送者发送确认，指出消息已正确接收来对每个消息做出反应。在多目连接中使用确认并不是一个好主意，这是由于以下明显的原因：如果一个消息有 N个接收者，就会导致有 N个确认返回给发送者，这就会很快削弱使用多目传送的优点。相反， IONA的可靠多目传送协议使用消极确认、心跳（heartbeats，即持续的消息发送）和消息序列数的结合来保证可靠性。如果接收者确定它遗漏了一个消息，它就会要求发送者重新发送由其序列号标识的遗漏消息。

2. OrbixTalk体系结构

OrbixTalk使用可靠多目传送协议（RMP）作为基础来构建ORB通信协议。以同样的方式，ORB使用TCP/IP之上的IIOP来发送从客户机到一单独伺服对象的消息，OrbixTalk使用RMP来发送从一个客户机到多个伺服对象的多目消息。网络和应用程序协议层次如图 7-6所示。

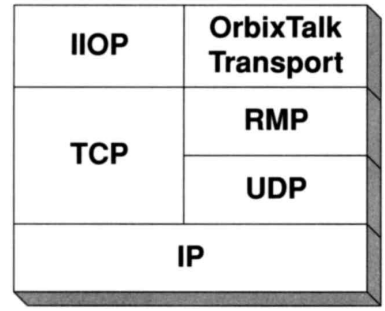

图7-6 OrbixTalk传输和IIOP

以前对CORBA远程方法调用的讨论都集中在发出对客户机中介的激发，这个中介把消息从客户机发送到一特定服务器进程中的一个单独伺服对象。客户机的中介和某些信息相关联，这些信息可识别目标对象、运行对象的服务

器进程的主机名字（或IP地址）以及服务器进程监听的端口号。

假定要向三个对象发送股票价格的变动。对于正常的基于 IIOP的通信，这要求三个远程方法调用，每个调用导致一个IIOP请求发送到特定的目标对象，如图7-7所示。

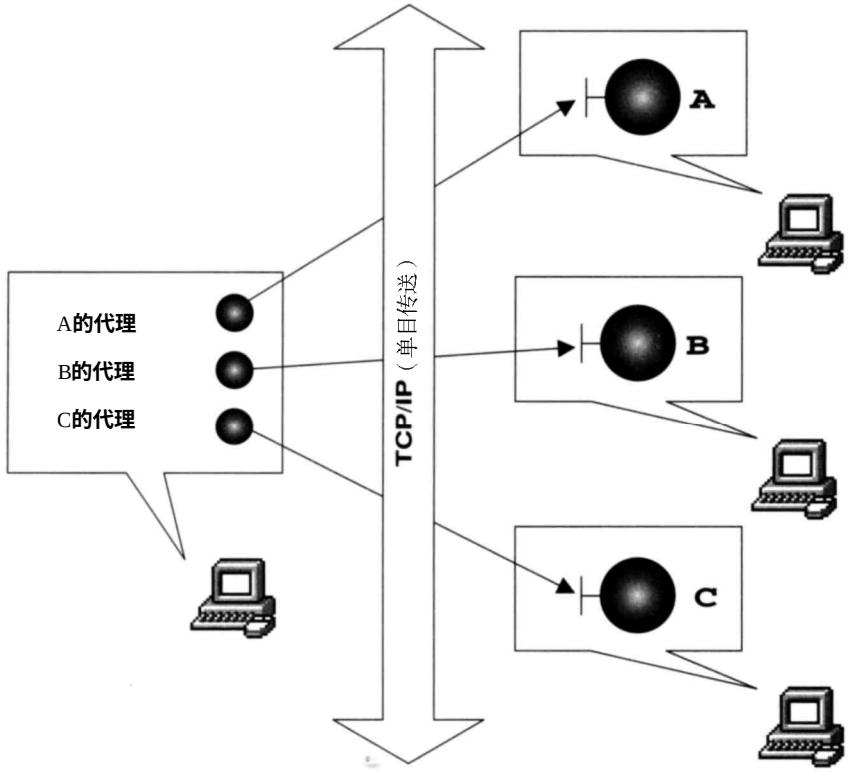

图7-7 通过TCP/IP的IIOP请求

用OrbixTalk和多目传送实现同样的结果更为有效，如图 7-8所示，向三个对象发送股票价格的更新用一个单独的多目消息实现。

OrbixTalk提供了如同基于IIOP通信中的中介对象和实现对象的相同概念，但有两个主要的差别。第一，所有的IDL接口必须只含有单向操作，因为多目传送是指通信是单方向的 —— 即发送者不希望收到回答。第二，中介对象不再和特定的伺服对象关联，而是和一标题名关联，该标题在内部映射到一IP多目地址。在图7-8中，标题名是“ /STOCK/IONAY ”。客户机中介称为谈话者，伺服对象称为接听者。每个谈话者和接听者与一个标题关联。除了一个给定的标题有多个接听者外，也可以有多个谈话者。

发出对谈话者的激发并不是对单个接听者的远程激发，而是对所有为特定标题（如图 7-8所示）而注册的接听者的远程激发。对普通中介的激发会导致消息由单目发送到特定 IP地址中的特定应用程序，而对 OrbixTalk谈话者的激发会导致使用特定的 IP多目地址来发送消息。从应用程序员的角度来看，除了获取对象引用中所包括的步骤外，编程方法是一样的。

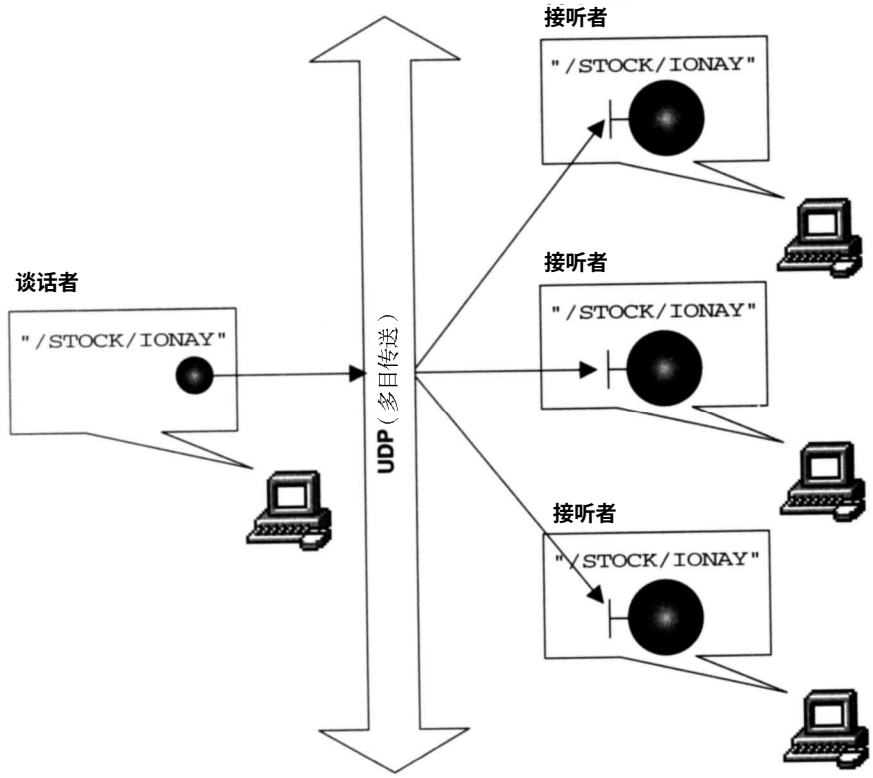

图7-8 通过多目传送的OrbixTalk

虽然OrbixTalk使用了专有的传输协议，但它仍然使用标准的 CORBA IDL来定义谈话者和接听者间的接口。 OrbixTalk还提供了CORBA事件服务的实现。虽然这可以用来构建使用标准事件服务接口的应用程序，却以基于多目传送的 RMP来作为底层传输机制。虽然，通信协议是专有的，但应用程序本身使用遵循CORBA的代码，允许程序员利用多目传送提供的优点。

3. 存储转发

到目前为止，可靠性只在传输层级别上保证。只要所有接听者应用程序都在运行， RMP就可保证它们能以正确的顺序接收到所有的消息。但是，很多应用程序有更严格的要求，它们可能会要求所有的消息都应该被传递，即使接听者在消息发送期间没有运行。在这种情况下，就需要一个持久存储消息的机制，它在更高级的层次上提供了可靠性。应用程序可在某段时间内不可用，但在重新启动后仍可接收所有相关的消息。为提供持久存储的可靠的消息接发，OrbixTalk提供了对RMP的扩展，称为存储转发协议（store-and-forward protocol, SFP），见图7-9中示例。基本思想是用一个专用服务器（存储并转发新进程）来接收从所有谈话者处送来的请求，把它们持久存储起来，并把请求转发到相关的接听者。如果一个接听者目前不可用，它就会在稍后要求重新发送先前的请求。这样的请求很容易应用到这个方法，因为所有的消息都已经持久存储起来。配置工具可允许程序员定义管理性的设置，例如请求在数据库中存储多久以及请求数据库多久备份一次。

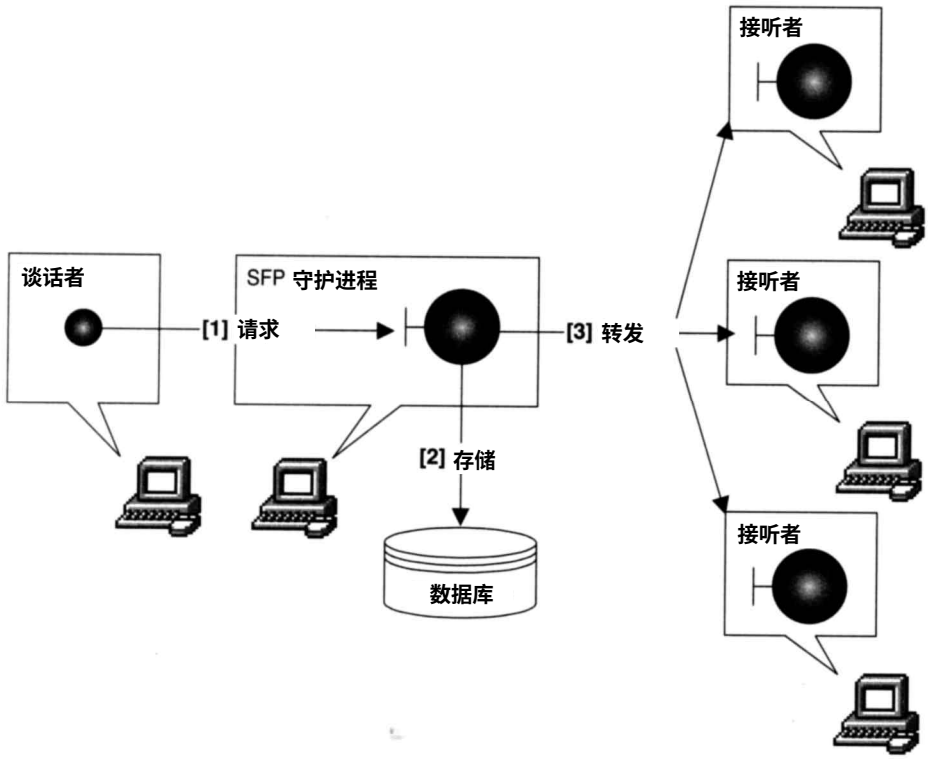

图7-9 OrbixTalk存储转发协议

7.7 小结

本章论述了消息接发，它被不严格地定义为组件间的非耦合、异步通信。我们看到基本的ORB提供了一定程度的异步性和非耦合特征。但是，在简要讨论商业上的面向消息的中间件产品时可以看到，在单向和延迟同步请求中，丢失了很多服务质量。其中一些特征由CORBA事件和通知服务支持，而其他则作为CORBA消息接发服务的一部分来论述。

对于大规模的系统，多目消息接发通常是用于通信的一个有效机制。像OrbixTalk这样的产品提供了标准的CORBA接口，但却使用IP多目传送作为通信机制。

当CORBA用来建造大系统时，经常需要其中的元素以异步、非耦合的方式通信。CORBA提供了大量的机制来完成这一任务，这些机制在复杂性、健壮性和性能方面都各不相同。多目通信同样也是构建需要消息接发的系统的有效方法，而且它在将来实现的CORBA规范中会被标准化。