一种基于中介的适应性软件协同环境的设计与实现*)

许 婷 俞 春 陶先平 吕 建
(南京大学 计算机软件新技术国家重点实验室 南京 210093)
(南京大学 计算机科学与技术系 南京 210093)

摘 要 Internet平台的发展为软件协同带来了巨大的挑战。Internet平台要求软件协同及中间件能够适应Internet之上软件的多样性,复杂性和多变性。而传统的软件协同存在协同模式单一的不足,不具有适应性和灵活性的特点,难以适应网络环境和用户需求的变化。本文在Internet软件协同研究背景下,基于共享空间协同技术,设计并实现了一个基于中介的适应性协同环境,为Internet平台提供了良好的中介协同支撑。

关键词 协同,中介,适应性,元组,匹配

Design and Realization of a Space Based Adaptive Software Coordination Environment

XU Ting YU Chun TAO Xian-Ping LU Jian
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)
(Department of Computer Science, Nanjing University, Nanjing 210093)

Abstract The development of Internet platform brings huge challenges to software coordination. Internet needs software coordination and related middleware canbe adapted to the diversity, complexity and variability of its environment. Traditional software coordination models seems no flexibility and adaptability ,so difficult to satisfy the changes of the environment and users' demands. This article proposes a space-based coordination architecture, in which space is a shared data store . This kind of architecture has outstanding flexibility and adaptability ,which gives a good supporting to software coordination on the Internet.

Keywords Adaptive ,Coordination, Match ,Space , Tuple

1 引言

"协同"一词是我们对"Coordinate"及其名词形式"Coordination"的汉译。从技术的角度来看,一种软件协同技术包括两个层次,一是其协同模型,二是该协同模型的软件实现。协同模型,作为"绑定一组分离的活动为一整体的粘合剂"[1],为主动独立的被协同实体之间的交互的表达提供一个框架。它通常涉及被协同实体的创建与撤销、实体间的通信、实体的空间分布等内容。Internet上的各种资源,因为应用的需求,需要恰当的协同和封装来完成相应的服务。由于Internet环境具有一些区别于传统分布计算环境的特点,对软件协同提出了一些新的要求。首先是网络连接的多样性和动态性:由于Internet中网络连接的带宽和质量差异很大,因此具有多样性的特点;其次是硬件设备和软件平台的多重异构性:Internet上的计算机在硬件和软件上有很大差别;最后是用户需求的个性化和多变性:Internet面向的用户具用多种不同需求,同一用户的需求也会不断变化。以上这些特点要求用于Internet资源协同的系统应具有灵活性和可适应性的特点,能够对用户需求的变化和环境的变化有一定的适应能力。

软件系统的可适应性是指软件系统具有对变化的适应能力[2]。这种变化主要来自于用户的需求和外部环境的变化。在软件协同中,协同环境应该提供一定的适应能力。首先是对用户需求的适应能力:协同环境应该能够适应用户多种要求;其次是对不同协同实体类型的适应性:能够适应不同类型的协同实体交互。最后是对协同中错误的适应能力,例如网络环境的错误或服务调用的错误:协同环境应该具有对错误的处理能力。

传统软件协同模型主要分为两种。一种采用以RPC为基础的交互方式,如RMI[3],CORBA[4]都是一种空间紧耦合的模型。这些模型依赖于Client/Server结构,需要保证Client端对Server端的单向引用,同时无法克服一次交互过程中可能产生的问题和失败。而基于共享空间的模式是一种时间松耦合、空间松耦合的协同模型。在这种模型中存在着一块公共区域:中介(Space),协同的双方并不直接进行交互,而是通过中介作为媒介来进行交互。在有中介的协同模型中,中介充当了请求方和服务方之间的桥梁。请求方和服务方之间的交互是匿名的,即事先可以不知道对方的名字标识、访问地址、服务的调用方式等。他们所知道的仅仅是中介的名字标识,地址和调用方式。请求方将自己的需求提供给中介,服务方把自己可以提供的服务注册到中介上,这是两个可以不同步的操作。中介为每个请求匹配到相应的服务,并组织调用,最后将结果返回给请求方。正因为中介的此种特性,我们可以在其中添加适应性的部分,使中介能够支持适应性的软件协同过程。这种适应性主要表现在以下两个方面:

1. 适应性的匹配算法:中介通过一定的匹配算法来为请求方寻找相应的服务方。在一般的基于中介的协同环境中,中介提供固定的匹配算法。这种匹配算法可能有多种,匹配过程也可能比较复杂,但所提供的灵活性是非常有限的。如果匹配算法能够具有适应性,不同的协同实体可以采取不同的匹配算法,将大大提高协同对个体需求的适应性。

2. 适应性的调用方式和调用过程:在基于中介的交互模型中,由于中介的存在,将以前的一条完整、统一的信道分割

*)本文受国家自然科学基金(No. 60273034),国家重点基础研究发展规划973(No. 2002CB312002),江苏省自然科学基金(No. BK2002409ZS991-A25-014-G)资助。许 婷 硕士研究生;陶先平 副教授;吕 建 教授,博士生导师。

成了两个部分:中介和请求方之间的信道及中介和服务方之间的信道。这种分割从客观上提供了中介对不同实体类型和信道进行适应的可能。同时由于请求方和服务方之间的匹配过程和调用过程也是分离的,因此中介可以在调用的过程中提供容错功能,以适应网络环境的变化。

而现有基于中介的系统,如 Javaspace[5],Mars[6] 并没有利用中介来提供协同过程中的适应性。本文提出了一种新的基于中介的软件协同环境。这种协同环境基于中介,但是扩充了中介的功能,提供了协同中的适应性。在本文所提出的协同环境中,中介不仅能够实现请求方和服务方之间的时间/空间松耦合,而且通过动态匹配机制以适应双方的协同要求。同时,中介不仅能够对请求方和服务方之间的协同提供最基本的支持,而且在请求方对服务方调用过程中,可以适应双方不同的实体类型和信道;在服务方出现错误的情况下进行相应处理,因此具有很强的适应能力。本文第 2 节将介绍基于中介的一般的协同模型,这是适应性协同环境的基础。第 3 节将详细介绍中介所提供的适应性功能特性,将具体从匹配和调用两个方面介绍。第 4 节通过一个具体的实例对中介的适应性功能展开说明。第 5 节介绍了系统的实现情况。最后对全文和目前的研究状况进行了总结。

2 基于中介的一般协同模型

在基于中介的协同环境中,中介是用于请求方和服务方协同的机构。中介所需要完成的功能包括:1)接受请求方的请求,并转换成中间形式;2)接受服务方的注册信息,并转换成中间形式;3)根据请求匹配相应的服务;4)为请求方调用服务方,并返回结果;在南京大学软件与新技术国家新技术重点实验室研发的"基于 Agent 多模式软件协同中间件"中,我们给出了中介的一般的协同模型及其实现[7]。

2.1 中介中信息的存放方式

在中介中,请求方的请求信息和服务方的服务信息都被抽象为元组的形式存放,中介为每一个请求元组匹配相应的服务元组。元组是匹配的基本依据,具有相应的数据表示形式。一个元组由若干分量所构成,可以具体表示为如下矢量形式:

(方法名,{备选方法名集合},{参数类型序列},{参数值序列},返回类型,其他信息)

在元组的内容中,第一个分量代表着方法名,在计算机中应该表示为一个字符串。方法名是服务的名字。对于请求方来说,它所提供的方法名代表了它所期望的服务的语义信息,比如提供方法名为"add",表示其希望得到加法的服务,"sub"表示其希望得到减法服务。不同的是,对于服务方来说,它所提供的方法名表示它所提供的服务的具体的方法名,而不仅仅是一种语义,而是实际可以调用的方法名。

备选方法名集合:在计算机中应该表示为一个字符串的集合,仅仅是对于请求方而言的,而且对于请求方来说也是可选的。代表了请求方在服务名称上的可能的多种近似的选择。

参数类型序列:代表了服务所需要参数的相应类型的序列;

参数值序列:参数相应的值序列;

返回类型:代表了服务返回的类型;

其他信息:服务方和请求方的其他信息,例如地址。

对于服务方而言,用以匹配的元组是其可以提供的一个具体的服务。而对于请求方而言,用以匹配的元组是一个模板,代表了其所希望得到的服务。所以元组虽然具有统一的表现形式,又可具体分为请求元组和服务元组。请求元组是请求方提供信息的抽象,服务元组是服务方提供信息的抽象。从静态的角度看,我们可把中介看作若干元组的集合。从动态的角度看,中介频繁地进行着元组的写入、写出和匹配工作。同时,中介不仅要进行请求方和服务方信息的匹配工作,还要根据匹配到的信息进行相应的调用,最后把结果返回请求方。

2.2 元组的匹配机制

中介对协同的基本支持体现为请求方和服务方信息的匹配,具体表现为请求元组和服务元组之间的匹配。中介为此提供了多种匹配算法和相对复杂的匹配机制。基本的匹配算法包括:

基本匹配算法:是决定两个元组是否匹配的最一般的算法,也是最为严格的算法。在参数类型上,要求参数类型的顺序是一定的,并且不考虑模板中的备选方法名序列。

考虑备选方法名的基本匹配算法:在基本匹配算法的基础上,考虑了请求方提供的备选名字。这种算法比基本匹配算法要宽松,考虑了请求方的多种选择,具有更高的灵活性,但是根据这种算法找到的元组不一定是匹配程度最高的。

非严格匹配算法:这种算法主要用于服务型构中参数的匹配。在有些参数顺序无关的服务中,为了能够更大范围地查找匹配的服务,可采取非严格匹配机制。该机制先将服务型构中的参数按字典序重新排列,将重排过的参数列表与服务进行比较,若匹配则原服务请求与该服务匹配。

每写入一个请求元组时,就代表了一次匹配过程的开始。中介首先通过基本匹配算法轮询所有的服务元组,找到可以匹配的元组。如果找到,匹配过程就此结束,如果不能找到相匹配的服务,将通过考虑备选方法名的基本匹配算法继续搜索。如果同样找不到,会通过非严格匹配算法搜索。直到找到相应的服务元组,或者所有算法用完后匹配失败。

2.3 服务的调用过程

中介根据匹配到的服务的注册信息,为请求调用相应的服务,并把服务结果返回给相应的请求方。默认的调用方式是 Web Services 的,这要求在中介注册的服务都应该是 Web Service 类型的。

3 基于中介的适应性协同环境

在基于中介的一般的协同环境的基础上,我们可以为中介增加适应性的特性。

首先,在匹配算法方面,我们为中介增加了可编程功能,这种可编程功能体现了针对不同交互实体的个体的适应性。请求方和服务方可以将自己特殊的匹配要求放入中介中。中介提供元空间存放这些特殊的匹配要求。根据这些要求,中介可以动态地实现请求方和服务方之间的匹配。这种动态的匹配是针对某个具体的请求方和服务方而言的,并且在某次涉及该请求方或服务方的匹配过程中被触发。我们把这种匹配机制称为动态匹配机制,而中介原有的匹配机制称为静态匹配机制。

其次,在调用方式和调用过程方面,我们可以对中介所提供的接口进行封装和改造,以适应不同类型的请求方和服务方及其调用方式。具体来说,对于请求方来说,只需将所需的服务名和参数传递给中介,并把具体找到服务方和进行相应调用的工作交由中介完成。调用过程中的实体类型差异和信道差异都由中介屏蔽。同时,调用过程本身也被中介所屏蔽,如果调用不成功,中介可为请求方提供容错处理,以适应网络环境的变化,从而最大程度地对协同过程给予支持。

3.1 具有适应性的匹配机制

适应性的匹配机制主要包括两个方面：一是匿名匹配和动态匹配机制；二是匹配流程。前者是为匹配本身提供适应性，而后者是为适应性的调用过程做准备。

3.1.1 匿名匹配与动态匹配机制

匿名匹配：在上述的算法中，我们都增加匿名匹配。所谓匿名匹配是指在上述的元组中，请求方可以把任意一项或多项表示为匿名分量(NULL)进行匹配。中介对匿名分量不做处理，默认为匹配。这种机制主要作用有二：1)请求方可以通过这种方式扩大可能匹配到的服务的范围。这样，当中介发现一个服务调用不成功的时候，可以找到更多可能替代的服务；2)允许请求方保留一些自己不确定的因素，把一些处理的工作都交给中介来做，用以适应可能发生的多种现实情况。

动态匹配机制：动态匹配体现了中介对协同实体需求的适应性。具体来说，请求方在提供请求信息的同时，可以把自己对服务方的具体要求写成一段程序，然后把这段程序提供给中介。中介把这段程序存放在元空间中。中介在涉及这个请求的相关匹配工作时，会在元空间中检索与匹配元组相关的程序，并触发执行这段程序。对于服务方来说，也是如此。在中介中，这样一个可触发的动态匹配程序可表示为一个三元组：

(Id , T , reaction)

Id 是这段程序提供者的标识符。T 代表一个元组，这个元组既可以是请求元组也可以是服务元组。reaction 代表 T 在和别的元组匹配时所要求触发的匹配操作，表示为一段程序：

```
boolean match (Tuple requestTuple, Tuple ServiceTuple)
{....}
```

这个元组代表了请求方或服务方所期望的适应性逻辑。这样的三元组由协同的双方提供，中介将这些适应性逻辑存储于元空间中等待触发。触发的过程可以表示为对 *match*(T,X)(或 *match*(X,T))方法的执行。X 代表将与 T 进行匹配的对应元组。动态匹配的过程如图 1 所示。

适应性逻辑的触发需要一定的适应性规约：

1)首先，对于特定的请求方或服务方来说，他们只能对自己写入的请求或服务元组写动态匹配程序，无权对别人写入的元组操作。也就是说任何一个有权写入(Id,T,reaction)的 Id，都必须是 T 的写入者。

2)其次，触发的程序中不能对中介中已有的其他元组进行修改。

这些规则的执行，需要中介采用一定的安全策略。一个可行的策略是：使得每个访问中介的请求者或服务者都需要持有一个 Id。同时每个元组都有一个所有者，这个所有者就是写入它的 Id。只有元组的所有者才能对其进行修改。对于(Id,T,reaction)，也只有在此 Id 是 T 的所有者的情况下，才能被写入。在这种策略之下，可以保证元组只被其写入者修改。

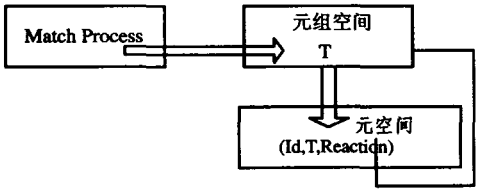

图 1 动态匹配机制的应用

3.1.2 具有适应性的匹配过程

上文中提到，在一般的基于中介的模型中，匹配的目的只是为了寻找一个可以调用的服务。而适应性的调用过程，需要多个备选的服务。同时这些服务因为匹配程度不同，应该具有不同的调用次序。因此我们应该改变中介的匹配过程：每写入一个请求元组时，就代表了一次匹配过程的开始。中介通过轮询所有的服务元组，找到相应的三个集合。这三个集合代表了与请求元组匹配程度不同的服务元组。这三个集合之间是没有交集的。集合 1 代表了基本匹配算法认为匹配的服务元组。集合 2 代表了考虑备选方法名的基本匹配算法认为匹配的服务元组。集合 3 代表了通过打乱参数次序的非严格匹配算法认为匹配的服务元组。

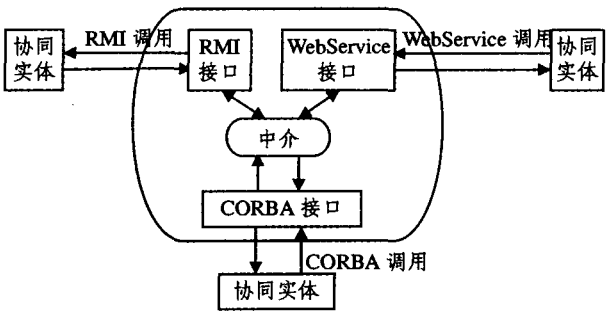

图 2 中介对不同信道和实体类型的适应性

3.2 具有适应性的协同过程

3.2.1 中介对协同实体类型和信道的适应

由于中介的存在，造成了发送方和接受方之间信道的分割。在客观上为发送方和接受方在信道和访问方式上的适应性提供了可能。在传统的 RPC 方式中发送方和接受方的信道要求是统一的。虽然 CORBA[7] 中通过 ORB 的互操作对这一缺陷做出了改变，但是迄今为止，能够支持的也是为数不多的互操作协议。而有了中介，支持发送方和接受方之间不同的信道和交互方式就变得相对容易。如图 2 所示，中介支持以下几种适应性变化：

1)为请求方请求中介提供了多种可能的信道。我们用包装的方法，为中介提供多种形式的接口。例如 RMI 形式的接口，Web Service 形式的接口。特别是，以 Web Service 方式提供的接口可以和其他服务一样地发布和调用。这就提供了一种 Web Service 可能的应用场景：客户可以到网上去查找自己所需要的服务，如果没有发现注册的合适的服务，也可以直接调用中介所提供的服务，间接地去寻找和调用自己所需要的服务。同时，我们为服务方的注册也提供了多种形式的接口。从这个角度来看，请求方或服务方与中介之间的通信是一种基于 C/S 模式的通信。作为服务器端的中介提供了多种供客户端调用的方式。

2)支持服务方不同的调用方式：在调用的过程中，中介可以根据匹配到的服务方的调用方式动态地组织参数，进行调用。

3.2.2 中介对调用中错误的处理

上文中已经提到，不同匹配程度的服务存放在不同的集合中。中介在匹配的过程中可以先选取匹配程度最高的服务进行调用。中介如果发现一个服务的调用发生异常，可以采用如下策略：1. 如果同一集合中还有其他服务，则调用同一集合中后面的服务；2. 如果没有同一集合中的服务，但是还有较低层次集合中的服务可以调用，则调用较低层次集合中的服务；3. 如果暂时没有其他服务可调用，可以让请求方在中介中等待，或者返回错误信息。从以上过程中可以看出中介通过这种调用方式可以具

有错误的处理和适应能力。

4 实例分析

上文中，我们对中介的基本工作机制和中介对协同过程的支持进行了介绍。下文将通过一个具体的实例来体现中介对协同过程的适应性支持，如图 6、图 7 所示：用户需要对关键词进行检索，得到一些相关联的网页；首先用户需要一个能够提供检索功能的服务，因此他把自己的需求提供给中介，期望得到相应的检索服务。同时，网上有三个可以提供检索功能的服务。这三个服务都可以提供检索，但是由不同供应商提供，在检索范围和质量上都可能存在差异，而且服务 SearchService1 提供服务的方式是 Web Serrice，而服务 SearchService2 提供服务的方式是 java RMI Server，SearchService3 提供服务的方式是 CORBA Server。这三个服务都将自己的服务信息注册到中介上。其中 SearchService1 还有特殊的要求，由于 SearchService1 是一个快速而小型的检索服务，因此它要求用户提供的关键词必须是与某一领域相关的，如果不相关，就不能提供服务。它把这一特殊需求也通过程序的形式写入到中介中。

```
Class B_Reaction extends Reaction
{
    Boolean match(RequstTuple T1,ServiceTuple T2)
    {
        Vector Para = T2. getPara();
        Int I = T2. getParaCount();
        Boolean b =true;
        for ( int j=0;j<I;I++)
            if (Para. elementAt(J). outofrandom())
                b=false;
        return b;
    }
}
```

中介为用户的一次调用进行相应的匹配，首先发现的服务是 SearchService1，但是通过自定义 match 方法的触发，发现由于用户提供的关键词落在了 SearchService1 的词库之外，因此不能选择 SearchService1。接着，用户选择了其他匹配的服务 SearchServie2 和 SearchService3 并分别放入集合 1 和集合 2 中（SearchService2 的优先程度更高，所以优先调用）。SearchService2 调用的过程中出现了异常，中介捕获了异常，为用户重新调用了 SearchService3，最终成功地返回了结果。

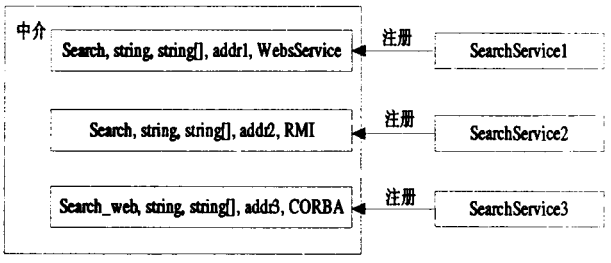

图 3 SearcgService 服务在中介中的注册

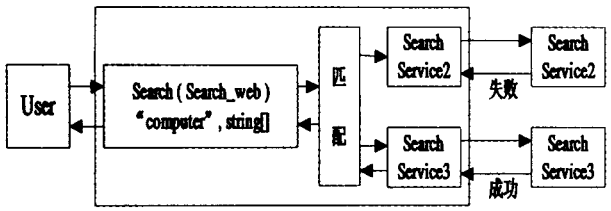

图 4 中介为 Search 请求的适应性匹配和调用过程

5 实现系统

本文的具体实现依托于南京大学软件与新技术国家重点实验室研发的“基于 Agent 的多模式协同中间件 ARTeMIS-M3C”系统的初步实现和改进版本。

在初步实现中，实现了一般的基于中介的协同模型。我们使用 Java 语言，并在 RMI、Web Service 等技术的支持下实现了中介的功能。系统实现分为以下几个部分：1）元组空间：系统通过元组库的形式提供了一个公共的空间，用以存放元组形式的信息。此外这个公共空间还提供元组存取的管理方法；2）元组的对外接口部分：系统提供了服务方注册的接口与请求方请求中介的接口；由于匹配所用的元组是一种中间形式，系统实现了信息和统一中间形式之间的转化工作；3）元组的匹配程序部分：实现匹配算法，提供请求元组和服务元组之间的匹配工作；4）持久化和安全部分：考虑到中介中的元组和其他信息需要持久的存储，系统利用 java 语言中提供的持久化存储机制，把中介中的元组和其他信息同时存入文件中。同时利用 java 语言中提供的身份认证和权限机制，实现了简单的安全管理。

初步实现已经能够实现中介的基本功能，作为整个系统的一部分，运行情况良好；而在改进版本中，增加了适应性的功能。所主要增加和修改的系统实现如下：1）可编程设施：增加元空间存放动态匹配的内容；增加用户接口用以提供元空间中的特殊元组；2）改变中介匹配流程：将各个匹配算法所获取的结果都存放到三个集合中，便于调用；3）支持多种实体类型和信道：对中介进行多种封装，支持多种对中介的调用方式；在中介对服务的调用中，增加多种调用方式；4）增加安全功能：为可编程设施增加新的安全设施。

改进版本运行状态良好，可以提高协同过程的适应性，完成协同过程中的非功能需求。

结论 综上所述，本文提出了一种具有适应性的软件协同环境。这种软件协同环境的基础是基于中介的软件协同模型。在其基础上实现了软件协同的适应性。这种软件协同环境旨在对协同进行灵活而有效的支撑。在支撑的过程中所提供的适应性主要表现在以下几个方面：1）协同实体需求的适应性；2）协同实体类型的适应性；3）协同中的容错能力。在以后的研究工作中，我们将进一步提高系统所提供的适应性，并避免这种适应性所带来的通信效率上的问题。

参 考 文 献

1 Carriero N, Gelernter D. Coordination languages and their significance [J]. Communication of the ACM, 1992, 35(2): 97～107
2 Laddaga R. Active Software [h]. In: First Intl. Workshop on Self-Adaptive Software, (IWSAS2000). 2000
3 Java™ Remote Method Invocation (RMI). http://java. sun. com/j2se/1. 3/docs/guide/rmi/
4 Object Management Group. Common Object Services Specification. Volume1, March 1994. http://www. omg. com
5 The JavaSpace Specification. Sun Microsystems. http://chatsubo. javasoft. com, June 1997
6 Cabri G, leonardi L, Zambonelli F. Mars: a Programmable Coordination Architecture for Mobile Agents [J]. EE Internet Computing, 2000, 4(4): 26～25
7 许婷，俞春，陶先平，吕建．软件协同中基于中介的协同模型应用研究．电子学报，2004，12A：64～68
8 Cabri G, Leonardi L, Zambonelli F. Mobile-Agent Coordination Models for Internet Application [J]. IEEE Computer, 2000, 33(2): 82～89
9 Gelernter D, Carriero N. Coordination Languages and Their Significance [J]. Communications of the ACM, 1992, 35(2): 96～107