基于事件的发布-订阅系统模型

汪 洋[1,2] 谢 江[3] 王振宇[2]
(中国科学院软件研究所 北京 100080)[1] (武汉数字工程研究所 武汉 430074)[2]
(上海大学计算机学院 上海 200072)[3]

摘 要 Internet的广泛应用已经改变了分布式系统的规模,使得传统的基于请求/应答的点对点的同步通信已不能很好地满足大规模的动态分布式应用环境。为了加强大规模的分布式环境中实体之间的通信协作,系统要求更加灵活的通信模型,以反映应用的动态和非耦合特性。基于事件通信的中间件是建立大规模分布式系统的有效方式,发布者/订阅者(Publisher/Subscriber)是目前广泛使用的基于事件的通信模型,支持发布者和订阅者之间在时间、空间和同步方面的非耦合以及多对多的通信模式,提供大规模系统所要求的交互间的松散耦合。本文详细阐述了事件系统的组成以及基于事件系统的PUB/SUB模型,提出了基于PUB/SUB的事件系统中间件的核心及实现的关键,这些中间件为大规模的分布式应用提供更多和更有力的保障。

关键词 基于事件的系统,发布者/订阅者模型,基于事件的中间件,分布式系统

Event-Based Publish/Subscribe System Model

WANG Yang[1,2] XIE Jiang[3] WANG Zhen-Yu[2]
(Institute of Software, Chinese Academy of Sciences, Beijing 100080)[1]
(Wuhan Digital Engineering Institute, Wuhan 430074)[2]
(School of Computer Engineering and Science, Shanghai University, Shanghai 200072)[3]

Abstract The Internet application has changed the scales of distributed systems, which making the point-to-point and synchronous communication based on request/reply paradigm is not enough for the large-scales dynamically distributed application environment. In order to communicate and cooperate efficiently among the entities in a large-scale distributed system, it needs more flexible communication models to reflect the dynamic and decoupled nature of the applications. The event-based middleware can be well used to build such large systems. Publisher/Subscriber scheme based on the event model supports many-to-many communication and time, space, and synchronization decoupling of subscribers and publishers and provids the loose coupled form of interaction required in the large-scale system. The paper specifies the event model and the publisher/subscriber model, and presents the key factors of event-based middleware architecture.

Keywords Event-based system, Publisher/subscriber, Event-based middleware, Distributed system

1 引言

虽然CORBA和Java RMI等中间件系统可较好地解决复杂的分布式应用,但这些基于调用的中间件系统采用的是请求/应答模式,即客户端通过发送一个请求或者是执行一个远程方法调用(RMI),然后接受一个返回应答的方式向服务器端请求一个特别的服务[1]。该方式可在数量不太大的客户端和服务器的局域网分布式环境中得到较好的应用,但Internet的广泛应用已经改变了分布式系统的规模,使现在的分布式系统包括了分布于世界各地的成千上万个实体,这使得传统的基于请求/应答的点对点和同步通信已不能很好地满足大规模的动态分布式应用环境。大规模的分布式环境一般采用多对多的通信,这就要求有更加灵活的通信模型,以反映应用的动态和非耦合特性。为减少应用设计者的负担,大规模应用系统中实体之间的通信应该由基于高效通信机制的专用中间件基础设施来提供足够的保证。基于事件通信的中间件是建立大规模分布式系统的有效方式。在基于事件的系统中,PUB/SUB是目前广泛使用的通信模型,它采用事件作为基本的通信机制,提供大规模系统所要求的松散耦合的交互模式:订阅者(如客户端)以事件订阅的方式表达出它有兴趣接收的一个事件或一类事件;发布者(如服务器)可将订阅者感兴趣的事件随时通知相关订阅者。一个事件可异步地传播给所有对它感兴趣的订阅者。这种基于事件的交互模型中,支持在发布者和订阅者之间的非耦合、多对多的通信,在时间、空间和同步方面是完全非耦合的。本文对事件系统的组成与分类进行了详细的阐述,并分析了基于事件系统的PUB/SUB模型的特性和种类。在此基础上,提出了一种基于事件的中间件的体系结构,为大规模的分布式应用提供了更多、更有力的保障。

2 事件系统

基于事件的通信模型为分布式和异构环境中的应用构件

汪 洋 博士,主要研究领域为软件工程、分布式系统、中间件技术等;谢 江 博士研究生,主要研究领域为软件工程、分布式系统理论等;王振宇 研究员,博士生导师,主要研究领域为软件工程、软件理论与工具开发等。

之间的异步互连提供了一种规范，特别适用于分布式应用环境。这些应用要求一个或者多个应用构件之间相互交互，以改变另一个应用构件的状态，该构件提供一对多或者是多对多的通信模式。基于事件的通信是异步方式，与传统的请求/应答通信模型比较，应用构件的通信关系相对松散一些；同时，由于其匿名特性，它能很好地适用于由大量的匿名交互的构件组成的应用系统，而不依赖于集中的控制[2]。

2.1 事件系统的组成

每个事件系统均有事件服务和事件模型，它们之间的相互关系如图1。

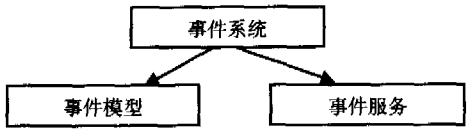

图1 事件系统的组成

一个事件系统是一个利用事件服务来执行基于事件通信的应用；

一个事件模型是由一组描述基于事件的通信模型的规则组成；

一个事件服务是实现一个事件模型的中间件，提供与另一个事件系统基于事件的通信。

在此区别事件服务和事件模型，是为了强调事件模型是定义事件服务的应用层视图，并定义了可以实现一个特殊的事件模型的一组事件服务；事件模型反映了预先定义好的不同的用法。如由 OMG 提出的作为 CORBA 一部分的 CORBA 通知服务就是一个事件模型，而由 SUN 公司提出的是 Java AWT 代理服务模型。这两个模型在其目标方面完全不一致，导致了它们提供的 API(即用法)完全不同。

CORBA 通知服务的事件模型支持包括 typed 和 untyped 很广泛的事件通信，还具有过滤和管理能力，而且可支持一些服务质量(QoS)特征(如事件的可靠性、连接的可靠性、事件的优先级和事件的提交顺序等)，以控制事件特征的传播，这些可由相对多而复杂的 API 反映出来。与此不同，Java AWT 代理事件模型则倾向于为小规模、集中应用(如 GUI 等)提供服务，因此它省略了 CORBA 事件模型中的许多特征，使得 Java 事件模型的 API 比 CORBA 事件模型要简单得多。

事件系统、事件服务和事件模型之间的关系从图2可以看出[2]。该图除了描述一个事件系统如何利用一个事件服务实现一个特定的事件模型外，还概括了一个事件系统和服务如何映射到一个传输机制，以及应用如何使用实体作为勾(hook)与事件服务联系起来。实体是一个包括生产者和消费者应用但不包括事件服务的构件。一个实体可同时扮演生产者或者是消费者事件的角色，或者是同时充当生产者和消费者事件的双重角色。

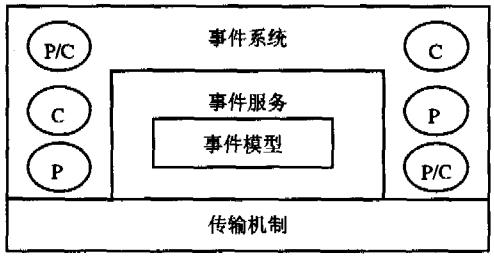

图2 事件系统的视图

P 表示生产者实体；C 表示消费者实体；P/C 表示既是生产者又是消费者实体。

2.2 事件模型

事件模型定义了一个事件服务的应用视图，定义了事件服务对应用程序员可见到的风格。它明确了应用程序员如何明确地使用一个事件服务的构件来订阅事件和传播事件。在应用中，可分为三种不同类别的事件模型：它们是对等的(peer to peer)、仲裁者(mediator)、隐式的(implicit)，如图3所示。

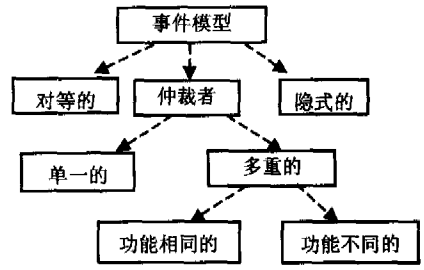

图3 事件模型

对等的事件模型允许消费者实体直接订阅已明确命名的生产者实体，生产者实体直接将事件提交给已明确命名的已订阅的实体。Java 分布式事件模型就是一个对等的事件模型。

事件模型使用一个仲裁者，允许消费者实体向指定的仲裁者订阅，而生产者实体将事件提交给仲裁者，然后由仲裁者将这些事件转发给那些已订阅的实体。

在事件模型中，根据仲裁者的数量和功能，将其分为单一仲裁者模型和多重仲裁者模型。CORBA 事件模型可以使用单一仲裁者，称为事件通道。多重仲裁者又可进一步分为功能相同的和功能不同的两种。前者表示所有仲裁者具有相同的功能，因此实体可以向其中任何一个订阅或者是提交事件。当仲裁者功能不同时，实体不得不向相应的正确的仲裁者去订阅或者是提交事件。CORBA 事件模型可以使用多个生成不同事件类型的事件通道。

一个隐式的事件模型让消费者实体向特定的事件类型订阅事件，而不是向另外的实体或者是仲裁者订阅。生产者实体生产某一类型的事件，然后提交给已订阅的实体。

总之，一个事件系统使用对等的或者是基于仲裁者的事件模型，允许其实体通过相互之间直接调用远程方法或者是一个或多个独立的仲裁者来交互，而隐式事件模型事件系统的实体则通过本地的事件类型订阅和提交来交互。

对等的和基于仲裁者的事件模型需要应用程序员分别获得显式命名的生产者和仲裁者的标识符，采用一个查找表或者是命名服务来维护。在基于对等事件模型的事件系统中，每个消费者要求获得它所感兴趣的每个生产者的标识符，应用程序员必须保证一个消费者向正确的生产者订阅，并在它们的生命周期中维护这些标识符。类似地，一个基于仲裁者的事件模型的事件系统的实体，需要获得所包含的仲裁者的标识符等，应用程序员必须跟踪一个特定实体需要连接的仲裁员的标识符。在一个事件系统中，程序员如果采用隐式的事件模型，则不需要显式地标识一个消费者需要与之通信的生产者，因为消费者已透明地使用事件类型向生产者进行了订阅。这就要求一个更长久的事件服务来负责定位、维护相应的标识符，并将事件类型映射到标识符。

由一个事件系统采用的事件模型影响基于事件通信的是系统中实体之间的匿名。通过由消费者实体向它感兴趣的事件进行订阅事件，并且事件的传播和提交的方式影响着匿名的程度。对等的方法允许明确命名的实体相互之间直接交互，因此实体不是相互匿名的；基于仲裁者的事件模型，实体用一个或者是多个仲裁者注册，提供一定程度的匿名，即实体相互之间是匿名的，但是其它仲裁者是可知的。隐式的方法允许实体透明地利用事件类型来交互，因此实体是相互匿名的，但是实现由事件类型到实体之间的映射的事件服务是可知的。

2.3 事件服务

事件服务提供了一种松散的、间接的、松耦合的"生产者/消费者"的通讯模式，这种通讯模式允许通讯双方在互不知道对方的情况下交换数据。事件服务允许对象在某一特定事件里动态登录或退出。一般情况下，事件的生产者不必知道对它感兴趣的对象，事件的消费者也不必知道产生事件的对象，这全部由事件服务处理。事件服务在互相透明的生产者与消费者之间产生一个松耦合通信通道。这种松耦合的通讯模式对于分布式管理系统具有非常重要的意义。

事件服务定义了三种成员角色：事件生产者、事件消费者和事件通道。

(1)事件生产者产生事件数据。

(2)事件消费者处理事件数据。

(3)事件通道是一个"代理"，它将事件生产者产生的事件数据透明地广播给事件的消费者。从事件生产者的观点看，它是事件消费者；从事件消费者的观点看，它是事件生产者。它是事件通讯的核心枢纽。

最基本的事件服务有四种实体：生产者、消费者、事件和事件服务；并有三种基本功能：为服务供应一个事件、注册感兴趣的一类事件和注销感兴趣的一类事件。生产者用来为服务提供事件，消费者注册它感兴趣的事件的规格说明，该说明描述注册的事件模式及在找到与该模式相匹配的事件时所执行的操作，最通常的操作是 notify。执行该操作，消费者用匹配的事件提交一个通知。

事件服务涉及到对一个事件服务的特征的分类。一个事件服务的特征可分为三个不同的类型，如图 4 所示。其中，构架主要关注实体的分布和一个事件系统的中间件，并以此方式组成一个事件系统构件之间的协作。交互模型定义生产者和消费实体之间相互通信的通道。特征则表述了一个事件系统所具有的其它功能和非功能特征。

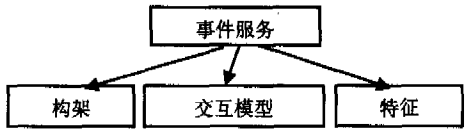

图 4 事件服务

2.3.1 构架

构架按照事件系统的实体所在的位置将一个事件服务划分成集中的和分布的两类。一个事件系统的实体如果驻留在同一物理机器的同一地址空间，则称为集中的。否则，如果一个事件系统的实体是分布的，则它们可能位于不同机器的不同地址空间。

2.3.2 交互模型

交互模型定义了在生产者和消费者实体之间发生的事件通信的通信路径，它定义了中间件的数量以及中间件相互协作从生产者到消费者发送事件的方式。与构架模型集中于描述一个事件服务的静态视图的实体和中间件相比，交互模型描述了一个事件系统中的信息流，因此它描述了一个事件系统的动态方面。

生产者和消费者实体可以使用由应用提供的实体的地址，以点对点的方式直接相互通信。中间件在将事件从生产者发送到消费者时使用确定的实体地址和一个单向强换的通信模式。Java 分布式事件模型允许生产者使用由应用提供的消费者的地址将事件发送到消费者。

生产者和消费者实体可使用一个命名服务进行相互之间的直接通信，命名服务实现将事件描述(如事件类型)映射到应用提供的实体的描述。中间件使用单向或者是多向的强转通信模式来将事件从生产者发送到感兴趣的消费者。

生产者和消费者实体可以使用隐式方法将事件的描述映射到由应用提供的实体的地址来进行相互之间的直接通信。中间件在将事件从生产者发送到消费者时使用一个多向强换的通信模式。

2.3.3 特征

特征描述了一个事件服务的其它功能和非功能特征。包括：

可扩展性。对 Internet 应用很重要，只有当所有的构件可扩展时，该系统才可扩展。

互操作性。基于事件的中间件系统允许集成多种构件，事件模型和订阅语言与编程语言和操作平台无关。中间件不依赖低层网络特殊性。

可靠性。在传送事件时，客户端或服务器可能会提出服务质量(QoS)方面的需求。所以，中间件必须支持一定范围(从最努力的到有保证的和及时的)的 QoS 保证。容错机制(如存储在数据库中的持久事件)允许中间件根据客户端和服务器的失败来操作。

可表示性。在描述事件和订阅时是一个很重要的需求，订阅允许根据事件的内容来过滤(基于内容的过滤)。复合事件的表示通过直观的、高层的抽象来帮助事件的订阅者去表示他们所需要的信息。

可用性。中间件应该容易操作，很方便地与应用编程语言进行集成。

3 PUB/SUB 系统

PUB/SUB 是基于事件系统之上的一种通信模型。相对于传统的消息传递、远程调用、通知、共享空间和消息队列等通信方式而言，虽然它们位于不同的抽象层次，不易进行比较，但它们在耦合通信能力方面存在很大差别。传统的通信方式与 PUB/SUB 相比只是有限地支持时间、空间和同步方面的非耦合。

3.1 PUB/SUB 系统模型

在分布式事件系统中，生产者在一条软件总线(事件管理器)上发布消息，消费者从总线上订阅它们感兴趣的信息。这些信息通常称为事件(event)，通过通知(notify)来提交[7]。

基本的 PUB/SUB 系统模型依赖于为订阅提供存储和管理以及有效交付事件的事件通知服务，订阅者对感兴趣某事件或某类事件进行订阅，发布者将订阅者感兴趣的事件随时通知给订阅者(见图 5)。

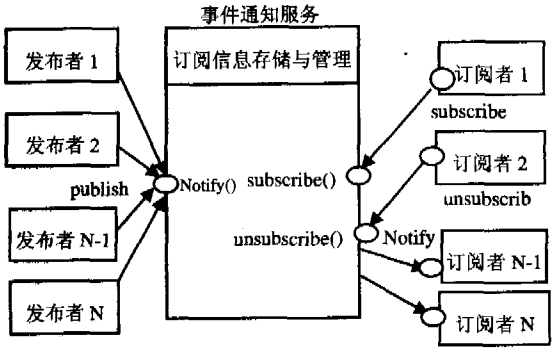

图 5　基于事件的 PUB/SUB 系统

事件通知服务充当生产者事件的发布者和消费者事件的订阅者之间的中间媒介(也可称之为代理)。订阅者通过调用 subscribe()操作向事件服务注册它们感兴趣的事件,而不需要知道这些事件的来源。这些订阅信息保存在事件服务中,不必转发给发布者。Unsubscirbe()则是终止一个订阅。

发布者调用 publish()操作来创建一个事件后,由事件服务向所有订阅者传播事件。所以,事件服务也可看作是订阅者的代理。当事件与每个订阅者感兴趣的事件相匹配时,将会得到相应的通知。所以,发布者和订阅者之间的事件服务是松散的、非耦合的。

在发布者和订阅者之间提供事件服务的非耦合通常可分为三种:

(1)时间非耦合。发布者和订阅者不必同时在线,它们不必同时参与交互。

(2)空间非耦合。发布者和订阅者不必相互知道对方所在的位置。发布者通过事件服务发布事件,订阅者通过事件服务间接获得事件。发布者和订阅者不需要拥有直接到对方的引用,也不必知道有多少个订阅者或者是发布者参与交互。

(3)同步非耦合。发布者/订阅者是异步模式。发布者可不断地生产事件,而订阅者(通过一个回调)则可异步地得到产生事件的通知。

信息的发布和订阅之间的非耦合,消除了交互双方的依赖性,极大地增加了系统的可扩展性。事实上,由于除去了这些依赖性而减低了不同实体之间的协调和同步,使得通信机制更适合于分布式环境。

3.2　PUB/SUB 的分类

订阅者通常只对特殊的事件或者是事件标本感兴趣,而不是对所有事件感兴趣。对感兴趣事件的不同描述方法导致了不同的订阅方式[3]。最常用的方式有基于主题的(topic-based)PUB/SUB 和基于内容的(content-based)PUB/SUB,以及最近出现的基于类型的(type-based)PUB/SUB。

3.2.1　基于主题的 PUB/SUB

最早的 PUB/SUB 模式是基于主题的,该模式在许多工业方案中得到了实现。在引入通道概念后,通道可绑定通信的双方,并可分类描述事件的内容。参与者可以发布事件,或者按关键字标识的主题来订阅事件。主题相类似的组,可用来定义组通信。订阅一个主题 T 可以被认为是变成了组 T 的一个成员,而在主题 T 上发布一个事件也相应地转换成在组 T 的成员中广播该事件。尽管组和主题是类似的概念,但它们通常与不同的应用领域有关:组用来维持在局域网中的关键构件备份之间的一致性,而主题用于为大规模的分布式交互建模。

基于主题的 PUB/SUB 系统引入了将主题映射到不同通信通道的程序设计抽象,它们表示类似于事件服务的接口,主题名字通常描述为初始化参数。几乎所有基于主题的 PUB/SUB 系统都提供层次化的组织模式,允许程序员按照相应策略去组织主题。在层次系统中,对某一节点的订阅暗含了该节点所有子主题的订阅。每个主题被视为自己拥有的一个事件服务,它由唯一名字来标识,并具有提供 publish() 和 subscribe()操作的接口。

3.2.2　基于内容的 PUB/SUB

尽管有了层次化编址并对通配符进行了相关改进,但基于主题的 PUB/SUB 表示的是仅能提供有限表达的一种静态模式。基于内容的 PUB/SUB 通过引进基于相关事件的实际内容的订阅模式,对主题进行了改进。即事件不是依据预定义的外部标准来分类,而是根据事件本身的特性,这些特性可以是承载事件的数据结构的内部属性,也可以是关联于事件的元数据,如 JMS 服务。

消费者通过使用订阅语言描述的过滤器(filter)来订阅可选的事件。过滤器通常以具有一定属性名字-值对和基本的比较操作(=、<、≤、>、≥)的形式定义限制条件。这些条件可以通过逻辑组合(and ,or)来组成复杂的订阅模式,订阅模式用来标识给定的订阅者感兴趣的事件,并传播相应的事件。事件服务为订阅提供一组包含订阅模式的 Subscribe()操作。

3.2.3　基于类型的 PUB/SUB

基于主题的事件分组代表了事件不仅在内容上而且在结构上具有共性。因而,人们主张用根据类型来过滤事件替代基于名字的主题分类。换句话说,事件种类的概念直接与事件类型相匹配,这使语言与中间件实现较紧密的集成。而且,类型安全可在编译时由相应事件类型的参数化抽象接口来保证,在结果代码中却不需类型的转换。基于类型的 PUB/SUB 可以通过事件类型的公共成员自然地描述基于内容的过滤,同时也保证了这些事件的封装性。

总之,存在几种不同的 PUB/SUB 系统的设计方法,它们提供了不同的表达力及性能开销。基于主题的 PUB/SUB 相对静态和初级,但可以比较方便地实现;基于内容的 PUB/SUB 有很好的表达性能,但需要有更高运行时间开销的复杂的协议。

4　基于事件的中间件

事件系统和基于事件的 PUB/SUB 模型为大规模的分布式系统应用提供了有效的组织形式和通信模型,中间件则是应用系统中十分重要的部分,可以让开发者快速构建系统并为应用提供相应的服务[4]。目前,大多数已有的应用于分布式环境的中间件系统(如 CORBA 和 Java RMI)是基于调用和采用请求/应答模式的。这些中间件可以运行在异构的操作系统之上,对整个分布式系统提供一个类似的、抽象的视图,它们可以在网络范围内为一定数量的客户端和服务器提供良好的通信,但不能胜任大规模的分布式系统[5]。

4.1　基于 PUB/SUB 中间件的关键因素

目前,虽然已开发了一些基于事件的 PUB/SUB 系统,但这些系统在应用时受到一定的限制,缺乏传统中间件在功能性方面如,调用的类型检查、可靠性、访问控制、事务等方面的支持。因此,应该将传统的中间件技术和 PUB/SUB 通信模式融合起来,以提供既有丰富功能又方便通信的基于事件的中间件[6]。

可扩展性：对 Internet 应用是很重要的要求，只有当所有的构件可扩展时，该系统才可扩展。这意味着中间件自身的实现必须是分布的，而且在所有中间件中不包含全局状态，同时网络带宽和内存等资源能被有效地使用。

互操作性：基于事件的中间件系统允许集成多种构件，事件模型和订阅语言是与语言和平台无关的。中间件不依赖低层网络的特殊支持。

可靠性：在传送事件时，客户端或服务器可能会提出服务质量(QoS)方面的需求。所以，中间件必须支持一定范围的 QoS 保证。同时，还要考虑容错机制，使在网络或者是构件失效时不影响整个系统，持久性事件和备份复制技术可实现更健壮的中间件。

可表示性：在描述事件和订阅时是一个很重要的需求，订阅允许根据事件的内容来过滤(基于内容的过滤)。合成的事件表示通过直观的、高层的抽象来帮助事件的订阅者去表示他们所需要的信息。

可用性：中间件应该容易操作，很方便地与应用编程语言进行集成。在语言上的支持包括在事件与编程对象之间的直接映射、支持订阅和发布的静态和动态的类型检查以及隐藏中间件的实现细节(如内部的事件格式等)。

4.2 PUB/SUB 中间件的实现

PUB/SUB 中间件的实现主要涉及到事件、代理和 QoS 三个主要方面。

4.2.1 事件

事件有消息和调用两种形式。事件通过操作(notify())将消息交付给订阅者，调用则是由事件触发订阅者执行特定的操作[7]。

消息：在最低层，网络上传递的数据就是一种消息。在多数系统中，由应用显式创建的事件通知就是消息的一种形式。消息通常由消息头和包含用户特殊信息的数据组成。典型的头字段中包括消息标识、发起者、优先级别或期限时间，这些能被系统解释或者纯粹是为消费者提供信息服务。

调用：在较高层次上，我们通常会区分调用和消息。一个调用指向一个特殊类型的对象，并有定义好的语义。系统确保所有的消费者具有一个与处理调用相匹配的接口，接口在调用者与被调用者之间担当一个绑定的协议。提供调用风格的交互、有不同语义和变化寻址模式的系统称作消息系统。它们在 PUB/SUB 或者是消息队列系统之上引入附加逻辑，将低层的消息转换成订阅者方法的调用，这些订阅者必须是相同类型的。

4.2.2 代理

中间件代理是生产者与消费者之间的数据传递的桥梁。代理可按照体系结构特征或者是为数据提供的保证(如持久性和可靠性)来分类。

体系结构：PUB/SUB 系统允许生产者和消费者之间以异步方式来交换事件。异步可以通过生产者将消息传递给存储它们的实体，并按要求转发给消费者来实现。由于通过中央的实体来存储并转发消息，这种方法被称为集中式体系结构。基于这种系统的应用在可靠性、数据一致性或者对事务处理的支持方面有很高的要求，但并不要求很高的数吞吐量，如电子商务和银行业务应用就是这种类型。

异步也可以利用在生产者和消费者之间实现存储转发机制的操作原语来实现。因此，对应用而言，通信是异步和匿名的，不需要中间实体。这种方式由于在系统中没有中央的实体而被称为分布式体系结构。该体系结构可很好地适用于快速有效的暂时数据传递，像证券交易和多媒体广播就是这种类型的应用。相对于完全分散的系统，这种方法使用专门的服务器来执行复杂的协议，保障持久性、可靠性和高可用性，以及基于内容的过滤和路由。

分发：数据的传输有几种不同的方式。数据可使用点到点的通信原语来传送，或者利用硬件的广播机制，如 IP 广播。通信机制的选择依赖于像目标环境和系统的特点等方面的因素。

像消息队列系统这样的集中式方法，在生产者/消费者与中央代理之间常使用点到点的通信原语。这些系统更多地关注强保障性，而不是系统的吞吐量和规模可扩展性。基于主题的 PUB/SUB 系统可直接利用基于组的通信方式和通信协议来为订阅者分发事件。为了保证高的吞吐量，通过使用 IP 多路传输协议或者是一个更广范围的可靠多路传输协议 。

4.2.3 服务质量(QoS)

由于基于事件的 PUB/SUB 系统中的订阅者与发布者是非直接通信的，因此对 QoS 的支持与保证相对直接通信的模式要复杂。传统的支持 PUB/SUB 通信的分布式应用(如 CORBA 的事件服务、CORBA 的通知服务、Java 消息服务 JMS)在支持 QoS 方面也是有限的[8]。因此，随着应用要求的不断提高，在 PUB/SUB 系统中如何保证 QoS 也就非常重要。

PUB/SUB 通常考虑到的服务质量主要有持久性、事务性保障和优先级。

持久性：在 PUB/SUB 系统中，消息可在没有生成应答时被发送，并可在发出数小时后才被处理，通信各方不用控制消息是如何被传送的以及何时被处理的。因此，消息系统必须在可靠性及信息的持续方面提供相应的保证。只知道一个消息已到达位于生产者和消费者的消息系统是不够的，还必须保证在消息系统失败时消息不丢失。

分布式 PUB/SUB 系统通常不提供持久性，因为生产者直接将消息发送到所有的订阅者。除非生产者保留每个消息的备份，否则一个有故障的订阅者在恢复时不可能获得丢失的消息。

优先级：与持久性一样，消息的优先级也是一些消息系统提供的一种服务质量。事实上，它要求将消息分类，以待消费者按照优先级顺序来处理。如一个实时事件，可要求在其它消息之前立即得到响应和处理。

优先级影响在传输中的消息，在运行时执行的优先级由应用的调度者来处理，并不由消息系统来管理。这意味着在侦听同一主题的两个订阅者处理这些消息时可以有不同的顺序，因为它们处理这些消息的速度不一样，即使是 FIFO 的通信通道。优先级应该被看成是一种最重要的服务质量。

事务：事务通常用于对原子块中的多种操作进行分组，原子块要么完全执行要么不执行。在消息系统中，事务用来将消息分成原子单位，可能是一个完整的消息序列被发送(或者是接收)，或者是消息序列中没有一个消息被发送(或者是接收)。

可靠性：是分布式信息系统的一个十分重要的特征，它是将信息可靠地传递给一个或者是多个分布性实体的重要保证。由于信息系统中的生产者和消费者之间松散的同步，实现事件传播的可靠性就变得十分重要。

(下转第 123 页)

移实例负责一个特定业务流程的执行，其工作性质决定了其域内停留相对时间较长，跨域迁移率较小，通信分散度较小，因此，由上面的分析和实验结果可知，算法对迁移实例间频繁通信和域内频繁迁移有较好的适应能力，能够保证迁移工作流系统对通信可靠性以及时间和开销等方面的要求。

结论及进一步的工作 本文在分析迁移实例通信需要解决的问题及现有方法不足的基础上，提出了一种针对迁移工作流系统特定组织模式的通信模型及算法。同现有的方法相比，它具有以下优点：(1)迁移实例的双信箱机制，实现了迁移实例之间可靠异步的通信，保证信件的顺序"exactly-once"提交，并且减小了三角路由的带宽占用和迁移实例的注销注册开销；(2)高效透明的寻址机制，一方面有效降低了寻址时间，减轻迁移实例对出生地的依赖，另一方面能够使算法应对规模扩大有较强的适应性，增强了系统健壮性和效率。初步实验证明算法能够较好地满足迁移工作流对可靠性、适应性、快速响应性与效率的特定要求。进一步的工作将在通信安全性方面做深入的研究。

参 考 文 献

1 Cichocki A, Rusinkiewicz M. Providing Transactional Properties for Migrating Workflows. In: IEEE 10th Workshop on database &Expert Systems Applications, Florence, Italy, Sep. 1999. 90～94

2 曾广周，党妍. 基于移动计算范型的迁移工作流研究. 计算机学报，2003,26(10):1343～1349

3 陶先平，冯新宇，李新，张冠群，吕建. Mogent 系统的通信机制. 软件学报，2000,11(8):1060～1065

4 杨博，刘大有，杨鲲，张朝辉. 移动 Agent 系统的主动通信机制. 软件学报，2003,14(7):1338～1344

5 Finin T, Labrou Y, Peng Y. Mobile Agents can Benefit from Standards Efforts on Interagent Communication. IEEE Communication Magazine, July, 1998. 50～56

6 王红，曾广周，林守勋. 可移动 agent 系统位置透明通信的一种实现. 计算机学报，2001,24(4):442～446

7 Mishra S, Xie P. Interagent Communication and Syschronization Support in the DaAgent Mobile Agent-Based Computing System. IEEE Transactions on parallel and distributed systems, 2003, 14(3):290～306

8 Feng XY, Cao JN, Lü J, Chan H. An Efficient Mailbox-Based Algorithm for Message Delivery in Mobile Agent Systems, LNCS2240, Springer-Verlag, 2001. 135～151

9 Cao Jiannong, Feng Xinyu, Lu Jian, et al. Reliable Message Delivery for Mobile Agents: Push or Pull?. IEEE Transactions on Systems, Main and Cybernetics-Part A: Systems and Humans, 2004,34(5):577～587

10 StraBer M, Baumannn J, Hohl F. Mole—A Java based mobile agent system. In: Proc. Workshop Reader ECOOP'96. dpunkt. Verlag, 1996. 327～334

11 Mishra S, Xie P. Interagent Communication and Synchronization Support in the DaAgent Mobile Agent-Based Computing System. IEEE Transactions on Parallel and Distributed Systems, 2003, 14(3)

(上接第 115 页)

集中式 PUB/SUB 系统通常用可靠的点到点通道来实现发布者与订阅者之间的通信，并在稳定的存储系统中保留事件备份。因此，即使集中式事件代理可能由于延时传送而失败，但事件仍然可以可靠地提交给所有的订阅者。基于一个分布式事件代理的网络系统，通常使用可靠的协议向所有的或者是部分代理传输事件。基于组通信的协议和可靠的应用层多路传输是很好的选择。发布者和订阅者通常使用点对点的通信通道来与最靠近它们的代理进行通信。

结束语 PUB/SUB 是可适用于可扩展要求高、松散耦合系统的分布式交互模型。在此总结了基于三个方面的分类：时间非耦合、空间非耦合和同步非耦合。非耦合是很重要的特征，因为它强调了在抽象层的可扩展性，允许参与者不依赖于另一个而独立操作。然而，在实现层，可扩展性仍存在一些问题，因为 PUB/SUB 交互可建立在各种通信机制的基础上，容易被不恰当的体系结构所牵制。可扩展性通常与其它特性相冲突，如灵活的订阅要求复杂且开销大的过滤和路由算法。类似地，由于要保留事件日志且丢失的事件要求能被侦听到并可以重传，高可靠性保证就要求很高的开销。特别为广域网开发的协议也不能很好地扩展到订阅者数量大的系统，这是由于消息认可产生的网络流量很大，因而系统资源的开销也就大了。因此，基于事件的 PUB/SUB 中间件的开发与利用在一定程度上可以提高系统的效率。

参 考 文 献

1 Carzaniga A, Rosenblum D S, Wolf A L. Achieving scalability and expressiveness in an internet-scale event notification service, 1999

2 Eugster P Th, Felber P, Guerraoui R, et al. The Many Faces of Publish/Subscribe. ACM Computing Surveys, 2003, 35(2):114～131

3 Banavar G, Chandra T, Mukherjee B, et al. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In: Proceedings of the 19th International Conference on Distributed Computing Systems(ICDCS'99), Austin, 1999

4 Starovic G, Cahill V, Tangney B. An Event Based Object Model for Distributed Programming. In: Proc. of the Intl. Conf. on Object Oriented Information System, London, 1995. 72～86

5 Pietzuch P R, Bacon J M. Hermes: A Distributed Event-Based Middleware Architecture. Submitted to the Workshop on Distributed Event-Based Systems (DEBS), 2002

6 Cugola G, Nitto E D, Fuggetta A. Exploiting an Event-Based Infrastructure to Develop Complex Distributed Systems. In: Proceedings of the 20th International Conference on Software Engineering (ICSE'98), Kyoto, 1998

7 汪洋. 分布事件通知服务的关键技术及其应用研究:[博士论文]. 中国科学院软件研究所，2004

8 汪洋，王振宇. 一个支持 QoS 的实时 CORBA 中间件 ORB 的结构模型. 计算机科学，2002,29(4)