维普资讯 http://www.cqvip.com

基于设计模式的 Mogent 通讯服务器的设计与实现*)

任晓鑫　陶先平　卞昭娟　吕　建

(南京大学软件新技术国家重点实验室　计算机科学与技术系　南京210093)

摘　要　随着 Interent 的发展，移动 Agent 技术对人们的吸引力越来越大。对通讯的支持在移动 Agent 系统中起着非常重要的作用。为了控制和解决移动 Agent 系统 Mognent 的复杂性问题，本文讨论了用于设计和实现 Mognent 通讯服务的一种设计模式方法。

关键词　移动 Agent，Mognent，设计模式

Design and Implementation of a Communication Server in Mogent System Based on Design Patterns

REN Xiao-Xin　TAO Xian-Ping　BIAN Zhao-Jian　LU Jian

(State Key Laboratory of Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　With the development of Internet, mobile agent technology is becoming more and more attractive. The support of communication plays a very important role in mobile agent systems. In order to control and settle the complexity of the mobile agent system--Mogent, this paper discusses a design pattern approach to the design and implementation of the Mogent communication server.

Keywords　Mobile agent, Mogent, Design pattern

1　引言

移动 Agent 计算模式是随着 Internet 技术的发展而出现的一种新型计算模式。一般来讲，移动 Agent 是一段独立的计算机程序，它按照一定的规程，能够自主地在异构网络上移动，寻找合适的计算资源、信息资源或软件资源，利用与这些资源同处一台主机或一个局部网络的优势，处理或使用这些资源，代表用户完成特定的任务[2]。移动 Agent 具有自主性、流动性、协同性、安全性和智能性等特点。基于移动 Agent 模式的网络应用结构灵活，开放性好，支持移动计算、分布并行计算，可以有效降低网络负载，较好地适应异构环境，具有较高的坚定性和容错能力。因此，移动 Agent 技术在电子商务、分布信息检索和发布、个人助理、并行处理等方面具有广阔的应用前景。

Mogent[8]系统是由南京大学计算机软件新技术国家重点实验室自主设计和开发的移动 Agent 系统，它基于 Java 语言，以 Internet 为基础网络环境。Mogent 系统提供了基于任务级迁移的弱迁移方式，系统中提出并采用的结构化迁移机制，不仅把移动 Agent 的移动信息从功能体中分离开来，而且还实现了旅行计划和功能体的动态装配，这使得 Mogent 系统具备了对移动 Agent 强大而又灵活的表达能力。

Mogent 系统由三层结构组成，最低层是 Java 虚拟机，在其之上是 Mogent Server，也即站点层(Host)，最上层就是 mogent (Mogent 系统中我们称 mobile agent 为 mogent)，见图1。从概念上讲，Mogent 系统由两个实体：Host 和 mogent 构成。Host 是 Internet 上的某个节点的抽象，而 mogent 是一个移动 Agent 的概念模型。Mogent Server 提供了 mogent 运行时刻的全面支撑环境，包括：迁移支撑子系统、通信支撑子系统、安全支撑子系统。从 Mogent Server 的设计上讲，该 Server 应支持 mogent 的创建、执行、传输和终止，并为 mogent 提供通信和安全服务，满足其工作需求。此外，从环境支撑看，我们在 Mogent 环境中还提供了开发和使用子系统，支持 mogent 的开发、监控和远程操纵。

从功能上讲，通讯的支撑是 Mogent 系统中的一块十分重要的基础设施，是 mogent 互通有无，进行信息交流、分工合作的必要条件，同时也是其他子系统的基本通讯功能的重要保证，如迁移支撑子系统中对序列化的 mogent 数据进行网络传输[8]。对通讯设施设计的好坏，将直接影响到整个 Mogent 系统的性能(包括 mogent 迁移及通信的效率等)，以及系统的复杂程度，因此在 Mogent 系统中我们将使用一个通讯服务器来实现系统对通讯功能的需求。

本文在对 Mogent 系统通讯需求进行分析的基础上，针对现有系统模块及对象之间的复杂交互问题进行了综合分析，有针对性地提出了一个通讯服务器的整体解决方案来为系统提供一个有效的底层通讯服务支撑，并引入 Chain of Responsibility, Mediator, Decorator[1]等几个设计模式完成了通讯服务器的设计与实现。该服务器的设计屏蔽了 Mogent 系统关于通讯的内部细节，为整个系统提供了一个统一、通用的通讯视图，同时也降低了系统其他模块设计上的复杂性。

2　设计需求分析

2.1　Mogent 系统中的通讯

Mogent 通信支撑子系统是 Mogent 系统通讯的核心，它支持 mogent 间的点对点通信和组播通信。点对点通信主要

*)本项目受973项目(No. 2002CB312002)、863项目(No. 200AA1160101)、国家自然科学基金项目(No. 60273034, No. 60233010)、江苏省基础研究计划(No. BK2002203)资助。任晓鑫　硕士研究生，研究领域：移动 Agent 技术，web services 技术。陶先平　副教授，博士，研究领域：对象技术，移动 Agent 技术，软件协同技术。

维普资讯 http://www.cqvip.com

有同步发送，异步发送，同步接收和异步接收；而对于组播通信，Mogent 系统支持逻辑组播和物理组播的概念。

Mogent 系统中除了通信支撑子系统负责 mogent 通信之外，还有一些子系统如 Migration Manager，Other Modules（见图1）有通讯功能的需求，其中，Other Modules 里面的类文件服务器就有对 mogent 运行所需要的类文件的网络传输要求。

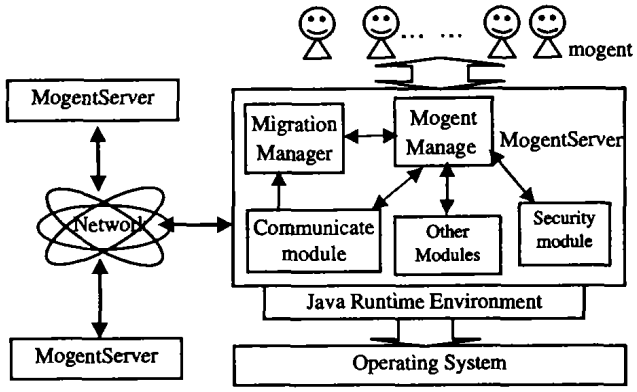

图1　Mogent 系统结构图

一般而言，Mogent 系统中的通讯主要有两大类：mogent 之间的通讯，以及 Host 之间的网络数据的通讯。其中第一类通讯分为 mogent 本地通信（图2中①）和 mogent 异地通信（图中②），这个是 Mogent 通信支撑子系统主要完成的工作，Mogent 系统通过 MailBox 等机制处理通讯失效问题，具体请参见文[3,6,7]等。第二类比较复杂，有 Host 间迁移支撑子系统所需的序列化后的 mogent 数据（图中③）的通讯，有代码服务器所要处理的类文件的传输（图中④），还有 mogent 因异地通讯而导致的消息通讯（图中②），其中包括一些系统消息，如寻址消息、同步应答消息等。

从图2中很明显可以看出迁移支撑子系统、通信支撑子系统以及代码服务器中都有一个通讯服务器，分别负责 Host 以及 mogent 之间的通讯。通常系统中对此的解决方案是为各个模块设计不同的通讯服务器实现它们各自不同的通讯需求，每个通讯服务器支持特定的通讯协议，并按通讯的特点实现特定的功能。Mogent2.0系统中采用的便是这种方案。

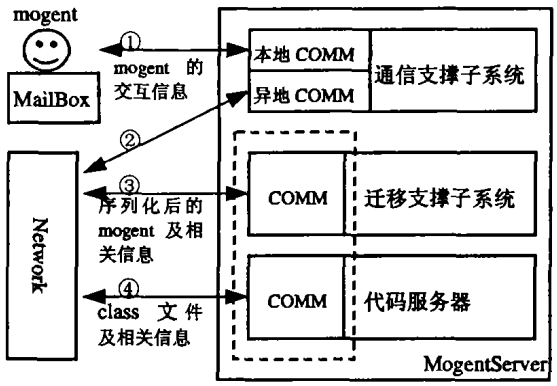

图2　通讯需求图

2.2　分析

我们在 Mogent2.0的系统设计和开发实践中发现，这些通讯功能模块的分散，虽然保持了各个子系统在逻辑意义上的功能的完备性，而实际却在大程度上对系统造成了负面影响，具体表现在以下几点：1)对一些基础通讯功能模块的重复性设计，造成资源的浪费。2)使得各个子系统之间的耦合性增加，子系统之间的分工界限不明。3)不必要地增加了系统中对象的数量，以及对象间错综复杂的引用关系，导致系统的复杂性剧增。

我们初步认为图2虚框中的部分应该统一划归到 Mogent 通信子系统中来，进行系统的整合处理，以构建我们的 Mogent 通讯服务器。

首先我们希望通讯服务器的设计中尽可能多地重用原有通信子系统中的核心模块。通过对代码的分析，我们发现 MailBox，MCast，Message 等类稍加改动就可以复用，有效地实现 mogent 的本地/异地通信，点对点/组播通信，同时对通讯失效问题的处理也可以达到复用。

其次我们从各类通讯的数据格式的入手进行分析：迁移支撑子系统的通讯格式是序列化后的 mogent 对象数据，代码服务器的通讯格式是一系列的 class 文件字节流、文件名及相应的存储路径信息，通信子系统的通讯格式比较复杂，有 mogent 通讯的信息，以及为保障 mogent 通信而额外添加的一些系统消息，这些消息需要收发双方达成共同的理解。我们发现尽管这些模块之间的通讯格式不同，而且它们的功能特点也不尽相同，可是从逻辑上讲，上面的所有通讯都可以看成最基本的字节流的传输，以及在此基础上添加的特定应用所附加的协议，如图3。

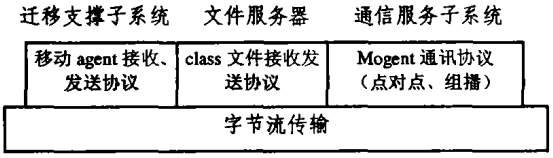

图3　通讯数据的分析

在上面分析的基础上，我们提出了新的 Mogent 通讯服务器的设计需求：

首先要提供全面的通讯支撑能力，包括基本的 socks 通讯，P2P 通讯，组播通讯，而且还需要在特定的情况下实现对通讯数据的加密、解密、验证等处理。我们在设计中考虑分层的设计，具体表现为一个共有的底层通讯设施，在此基础上集成的各个不同的通讯上层协议，以及其上的具体应用。这种分层的设计，使得我们在系统变动时，无需对基础的通讯设计进行改动，直接根据需求更改上层的应用模块就可以了，而且这种设计可以动态地增加其他关于通讯的上层支持设计。

其次我们希望通讯服务器提供对各种类型的通讯任务的统一、动态自适应的处理的能力。也就是说在用户视图上，通讯服务器的使用、理解应具有一致性，用户无需考虑太多的内部或者底层细节。例如，使用者只需要简单地调用 comm.send(mogent,host)，服务器就能把 mogent 的迁移数据正确地传递给 Host 上的迁移支撑子系统来处理；简单地调用 comm.send(mogentMessage)就可完成 mogent 通信消息的发送。

最后，我们希望通讯服务器的设计有很强的可扩展性，灵活性强，复用性好。

3　Mogent 通讯服务器的设计实现

Mogent 系统是一个有着相当复杂度的系统，由于 Mogent 通讯服务器提供整个系统的通讯支撑，因此会和很多的系统模块有着紧密的交互引用关系，所以它的复杂性也是相当高的，这给我们的设计带来了困难。在明确通讯服务器的设

维普资讯 http://www.cqvip.com

计目标的前提下，我们深入地剖析了 Mogent 系统中的复杂对象关系，发现借鉴设计模式的思想可以大大减轻我们的设计负担，而且运用那些成熟、灵活的模式，有利于实现我们上文中提到的第三点设计需求。

设计模式的概念最先来自于城市建筑专家 Christopher 对建筑模式的定义："每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复劳动"[1]。设计模式使得那些能在特定环境中良好工作的设计得以在相似的环境中被再一次应用或者借鉴，并且通过模式我们可以很方便地与他人交流这些经验，设计模式让我们摆脱了从失败中才能汲取经验教训的无奈。

下面将逐一对我们设计中考虑的要点进行解释。

3.1 基本通讯功能的保障

首先我们用到的是行为模式中的 Chain of Responsibility（责任链），该模式的思想是，给多个对象处理一个请求的机会，从而解耦发送者和接收者，该请求沿对象链传递至其中一个对象处理它[1]。

我们可以将各种通讯的内容看成一个统一表示的 TransData，该 TransData 可以被看作通讯信息字节流的抽象，BasicComm 是底层通讯设施实现类的公共接口，为所有上层模块所共用。CommProtocol 则是上层通讯协议实现的公共接口。这种设计的好处在于，不但分离了通讯底层和上层的逻辑视图，达到底层通讯设计的复用，而且还提供了底层对象与上层对象之间的请求手段。

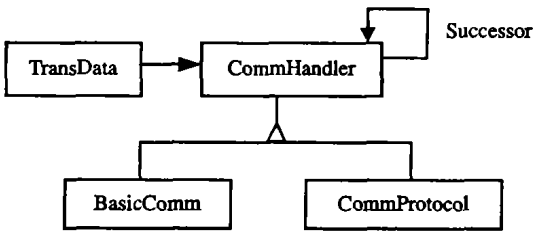

图4 数据传输功能分层

具体来说，当 TransData 发给通讯服务器的时候，首先由继承 BasicComm 接口的子类处理底层的通讯，获得通讯传输的字节流之后转发给继承 CommProtocol 接口的子类，完成各种不同类型的、与应用相关的上层解析处理。实际应用中一个典型的触发是这样的，BasicComm 收到一定数目的字节流之后触发上层模块如 CodeProtocol（文件服务器中的通讯协议），解析出文件、文件名、文件的存储路径等信息。

3.2 不同通讯方式的统一处理

其次我们用到的是 Mediator（中介者）模式。该模式用来封装一系列对象的交互，中介者使得消息对象和消息处理对象之间的交互不需要显式地相互引用，从而使其耦合松散，在 Mediator 里我们可以做一些工作，使得对消息对象和处理对象的匹配对外界透明化。

图5中 InternalMsg 是 TransData 经过 BasicComm 处理后的一个通用的内部消息接口，CommProtocol 接口有很多实现，如文件服务器的上层通讯协议 CodeProtocol，迁移子系统的上层通讯协议 MigProtocol 等，我们在图中仅以 Prol示意；中介 MessageDirector 负责对接受到的内部消息和具体的通讯上层协议进行匹配，这样对不同类型消息的接受、发送可以实现统一的处理，这是我们的核心模块。

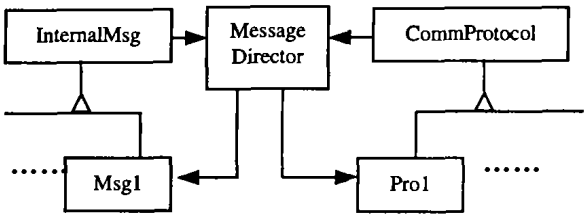

图5 中介对消息和协议的匹配

此外，该设计中还需要用到面向对象语言 java 的多态特性，如函数的重载，才能使用户在使用通讯服务器时真正感觉到通讯的自适应处理视图，例如 Mogent 系统中只需统一调用 comm.send(mogent,host)就可以为迁移子系统提供发送 mogent 的服务，统一调用 comm.send (mogentMessage)就可以为 mogent 间的交互通信提供支持。

3.3 其他的一些设计考虑

Decorator（装饰）模式旨在使你能够不需要生成子类即可给对象添加职责。这就避免了静态实现所有功能组合，从而导致子类急剧增加[1]。为了可以动态添加功能，我们使用了该模式。图6中 ConcreteComm 是通讯底层的基本功能的具体实现，而 SecDecorator 和 ZipDecorator 则是基本数据传输的一些安全性处理。该设计的优点在于：1）复用了安全子系统中的一些数据处理功能。2）底层的通讯中相同的功能被得以复用。3）动态实现通讯附加功能的添删。

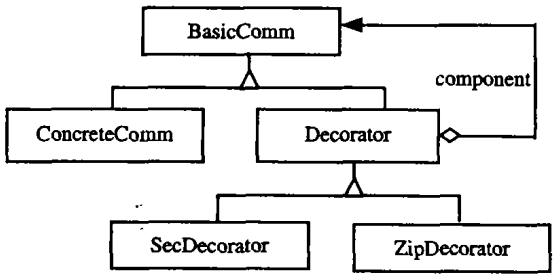

图6 动态添加通讯功能

此外，由于通讯服务器在一个物理节点（Host）上有且只有一个实例，并且要提供该节点上 MogentServer 访问它的一个全局访问点，很显然我们可以使用对象创建型模式中的 Singleton（单件）模式[1]来实现这一要求。

3.4 通讯服务器的实现情况

Mogent 通讯服务器的设计基本上已经在 Mogent 系统中实现，该服务器为整个 Mogent 系统的通讯提供了一个统一、通用的处理方式，该服务器的设计简化了 Mogent 系统中复杂而又容易混淆的通讯视图，减小了 Mogent 系统的复杂程度，同时也减轻了其他子系统的设计与开发中的负担。

在实现中我们使用一个统一的监听端口来监听通讯，避免了原来设计中对多个监听端口的管理。我们采用 Daemon 创建子线程来处理消息的方式，有着比较高的执行效率。此外通讯服务器的系统扩充性很好，很方便功能模块的添加、修改以及删除。

总结 Mogent 通讯服务器的设计与实现，为整个 Mogent 系统提供了一个通用的底层通讯支撑平台，使得我们在对 Mogent 系统及其子系统设计的时候，免去了很多通讯方面考虑的细节。从用户以及开发者使用的角度看，Mogent 通讯服务器自适应的对各种类型的通讯信息进行识别处理，非常方便。

（下转第207页）

维普资讯 http://www.cqvip.com

令 $\alpha=\lambda\|w\|^2$，由梯度下降算法以及 $w^Hw=\delta^{-2}$和算法(1)，有：

$$w(t+1)=w_c+\tilde{P}_c w(t)-\eta\tilde{P}_c(\Phi w(t)-\frac{w^H(t)\Phi w(t)}{\|w(t)\|^2}w(t)$$

$$+\alpha(\delta^{-1}-\frac{1}{\|w(t)\|})w(t)) \qquad (2)$$

由定理3，算法(2)是全局收敛的。算法(2)满足所有约束条件而无须再加限制条件 $w^Hw=\delta^{-2}$。

3 实验结果

本文算法是迄今为止唯一的全局收敛学习算法[10]，所以本节将主要验证上节的理论结果以及本文算法的实用性。实验假设三个麦克风在同一平面上[2]，三个麦克风成正三角形，扬声器在三角形的中央。如图1所示。

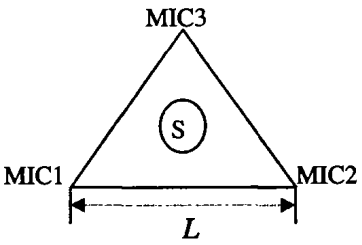

图1 麦克风阵列图

该麦克风阵列的方向向量是：

$$d(\theta)^H=[e^{-\frac{i\pi r}{\sqrt{3}}(\sqrt{3}\cos(\theta)+\sin(\theta))},e^{\frac{i\pi r}{\sqrt{3}}(\sqrt{3}\cos(\theta)-\sin(\theta))},e^{\frac{2i\pi r}{\sqrt{3}}\sin(\theta)}]$$

其中 $r=\frac{\omega L}{2\pi c}$。$\omega$ 表示频率，L 表示麦克风之间的距离，c 表示声速。参考坐标系的原点位于三角形的中央，θ 表示音波方向与 x 轴的夹角。

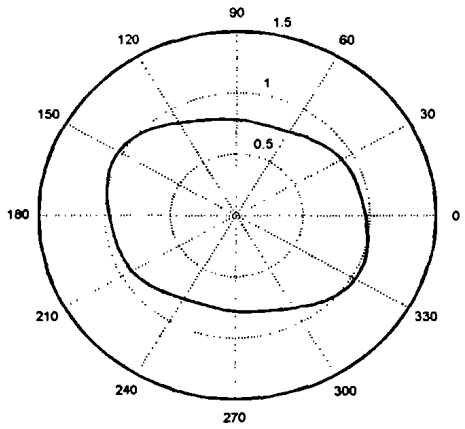

图2 未经处理的波束模式(dB)

在实验中，假设有两个信号，其中一个信号是一个非常弱的声音信号，来自方向 $\pi/2$，$\sigma_s^2=0.1$dB，另外一个信号是白噪声信号，$\sigma_n^2=0.2$dB。中央的扬声器已经关闭，并且不考虑来自其它方向的信号。令参数 $C=d(\theta_i)$，$b=1$，$r=0.4$，$\eta=0.0005$，$\alpha=tr(\Phi)$，$\delta=1$。图2为未做波束形成时，麦克风阵列的波束模式。图3为经过波束形成神经元处理后的波束模式。由实验可以看出，本文算法可以准确地加强来自 $\pi/2$方向的声音信号。

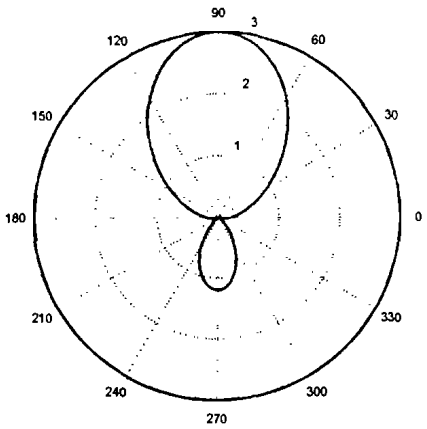

图3 处理后的波束模式(dB)

结束语 本文说明已有的应用于波束形成的 MCA 神经网络并不收敛。并提出了一种新的应用于自适应波束形成的全局收敛 MCA 神经网络学习算法。实验结果验证了我们的理论，同时也说明了本文算法的实用性。

参 考 文 献

1 Oja E. Principal components, Minor components, and linear neural networks, Neural Networks, 1992, 5: 927～935
2 Fiori S, Piazza F. Neural MCA for robust beamforming, Proc. of International Symposium on circuits and Systmes (ISCAS'2000), 2000, 3: 614～617
3 Xu L, Oja E, Suen C. Modified Hebbian learning for curve and surface fitting, Neural Networks, 1992, 5: 441～457
4 Mathew G, Reddy V. Developement and analysis of a neural network approach to Pisarenko's harmonic retrieval method, IEEE Trans. Signal Processing, 1994, 42: 663～667
5 Luo F, Unbehauen R, Cichocki A. A minor component analysis algorithm, Neural Networks, 1997, 10(2): 291～297
6 Feng D Z, Bao Z, Jiao L C. Total least mean squares algorithm, IEEE Trans. Signal Processing, 1998, 46: 212～2130
7 Pedro J Z. On the discrete time dynamics of the basic hebbian nerual network node, IEEE Trans. Neural Network, 2002, 13(6): 1342～1352
8 Anisse T, Gianalvo C. Against the convergence of the minor component analysis neurons, IEEE Trans. Neural Network, 1999, 10 (1): 207～210
9 Cirrincione C M, Herault J, Huffel S V. The MCA EXIN Neuron for the minor component anlysis, IEEE Trans. Neural Network, 2002, 13(1): 160～187
10 Ye M, Yi Z. A globally convergent MCA learning algorithm, Submitted to IEEE Trans. Neural Network

(上接第204页)

Mogent 通讯服务器是我们对系统开发中实际遇到的一些问题进行了深入分析后，运用设计模式的思想设计出来的，整个设计在效率、可扩展性等方面均有不错的表现，该设计对其他移动 agent 系统亦有借鉴价值。

参 考 文 献

1 Gamma E, Helm R, et al. Design Pattern: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995
2 张冠群，等. Mogent 系统迁移机制的设计和实现. 计算机研究与发展，2001，38(9)
3 冯新宇，陶先平，等. 一种改进的移动 Agent 通信算法. 计算机学报，2002，25(4)
4 Aridor Y, Lange D B. Agent Design Patterns: Elements of Agent application design. In: Proc. Autonomous Agents'98, Minneapolis, USA, 1998. 108～115
5 Tahara Y, et al. Agent System development method based on agent patterns. In: Proc. the 21th Intl. Conf. on software Engineering, Los Angeles, California, USA, 1999. 356～367
6 陶先平，冯新宇，李新，等. Mogent 系统的通信机制. 软件学报，2000，11(8): 1060～1065
7 冯新宇，吕建，曹建农. 通用的移动 Agent 通信框架设计. 软件学报，2003，5(14): 984～990
8 陶先平. 基于 Internet 的移动 Agent 技术和应用研究：[南京大学计算机软件新技术国家重点实验室博士论文]