移动 Agent 系统构建研究*)

贾志勇 谢 立
(南京大学计算机系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)

Research on Mobile Agent System Construction

JIA Zhi-Yong XIE Li
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract Mobile agent technology has strong adaptability to the status of network and executing environment, and provides a flexible and uniform framework for Web and distributed applications. Mobile agent system, as the foundation of agent computing, should provide the basic supports like executing support, agent migration, and communication, as well the extensive supports like naming service, security, fault tolerance, interoperation, application development and management tools. Based on the analysis of current mobile agent system, this article discusses these supports deeply, and also points out the main challenges to the development of mobile agent system.

Keywords Mobile agent, Agent system, System construction

1 引言

移动 agent 计算是近年来较活跃的一个研究领域,它本质上是一种"计算能力的自主和异步移动",对网络状况和执行环境具有很强的适应性,在移动计算、分布信息处理、服务定制、电子商务、软件集成以及网络监控和管理等领域都有着良好的应用前景[1,2]。

通常认为,移动 agent 计算需要有移动 agent 系统作为基本支撑。从通用性角度考虑,我们这里设定的 agent 系统模型与目前多数 agent 系统[3~16]和主要标准[18,19]是相容的,如下所述:(1)agent 系统由多个分布于不同网络节点上的 agent 平台组成。(2)每个 agent 平台都提供了 agent 的基本运行环境和对底层服务的抽象。(3)agent 能够在各个节点平台之间自主移动,并能在节点平台上执行、访问相关服务以及与其它 agent 发生本地和异地交互。在该模型下,对一个 agent 来说,其计算任务的完成需要以下三部分操作:(1)agent 受任务驱动在各个节点之间进行迁移。(2)agent 内部运算逻辑的执行以及对当前本地节点资源的访问。(3)与其它节点上 agent 之间的异地交互和协作。为了完成以上操作,agent 平台需要提供以下三种基本支持:(1)运行支持,即能够支持 agent 程序代码的执行,例如对于许多基于 Java 语言的 agent 来说,JVM 即可以提供基本的运行支持。(2)迁移支持,即首先暂停 agent 的执行并冻结其当前的执行状态,再将执行代码和状态传输到新的目的节点,最后在新节点恢复 agent 的状态并继续执行。(3)通信支持,即能够实现 agent 之间,各节点平台之间以及 agent 与节点平台之间的可靠通信。除了以上三部分基本支持外,根据实际应用环境的需要,移动 agent 平台还可能提供以下一些扩展支持:(1)安全支持,当移动 agent 平台处在 Internet 环境下时,需要满足 agent 系统和应用的多种安全需求,如保护节点主机不受恶意的 agent 攻击、防范和检测由主机节点以及其它 agent 发起的恶意攻击、保障 agent 和通信信息在传输过程中的安全等。(2)容错与恢复机制,提供错误检测和恢复机制,保障系统的稳定运行。(3)互操作支持,随着 agent 技术在开放网络环境中的应用,不同的 agent 系统之间以及 agent 应用之间常常会有相互通信、相互协作的需求,这通常需要 agent 平台给予相应的支持。(4)管理控制机制,良好的管理工具对 agent 系统的管理和应用开发都是很有帮助的,它也常常是商业公司所推出的系统(如 Aglets[3]、Concordia[4]及 Voyager[5])与研究机构所开发系统的重要区别。(5)方便、高效的应用开发工具,通常提供对 agent 应用开发、调试、部署等多方面的支持。

2 基本支持

2.1 运行支持

运行支持主要考虑对 agent 编程语言的支持和选择。由于 agent 可能会在多个异构的节点间迁移和执行,agent 代码的平台独立性就成为了编程语言选择时的首要考虑,因此,大多数 agent 系统都选择了能够提供虚拟执行环境的解释语言;此外,出于对安全性的考虑,提供类型检测和限制访问等功能的编程语言也较受欢迎。早期的 agent 系统大都采用经过改进的某类解释型语言,如 Agent TCL[6]采用了增强安全性的 TCL 脚本语言,MESSENGERS[7]采用了一种在书写形式上与 C 语言类似的脚本语言,April[8]中采用了名为 April 的脚本语言。在 Java 语言出现后,由于它所具有的平台无关性、内置的安全性以及比纯解释性语言更好的执行性能,使其很快成为首选的 agent 编程语言,JVM 也即成为了最为常见的运行支持环境。

2.2 迁移支持

*)本文研究得到国家 863 高技术项目基金(No. 2001AA113050)和江苏省高技术产业化项目基金资助。贾志勇 博士研究生,主要研究领域为软件 agent 技术,分布式计算。谢 立 教授,博士生导师,主要研究领域为分布式计算与先进操作系统等。

移动agent平台中，对agent的迁移支持主要有以下几方面需要考虑：

迁移粒度：通常有两种形式的迁移，一种是粗粒度的，称为弱迁移，即系统仅对由应用程序所定义的用来表示agent相关数据状态的特定数据结构加以捕捉和迁移，它通常是发生在由应用程序指定的程序的特定位置，且通常是在某一特定功能完成后发生；另一种即为强迁移，它是一种细粒度的迁移，在这种情况下，不仅agent的相关数据状态，而且其执行线程或进程的上下文、堆栈状态等也要加以捕获和迁移，因此agent在这种情况下能够在运行期间的任意时刻发生迁移并在新的节点恢复执行。虽然从功能上来讲，强迁移提供了更为灵活的处理机制，但它需要编程语言以及相应的程序运行环境的支持。一些标准运行环境如JVM等如果不加以改变就无法直接提供对强迁移的支持，此外，实践表明，弱迁移能够满足绝大多数的应用需求，因此，目前的移动agent系统中除了Agent TCL[6]和Ara[9]外，基本上都只提供对弱迁移的支持。

代码传输：由于同类型的甚至同一个agent可能会多次经过相同的节点，而它们的执行代码是不变的，因此在实现时常常会将agent数据状态和执行代码的迁移分开来考虑。对代码而言，一般有三种传输方式：(1)重量级方式，即执行代码(在Java中即为各个class文件)在agent的每次迁移中都随其一同传送到目的节点。(2)轻量级方式，即在迁移时只传输必要的数据状态，如果在执行时需要某些特定的代码和类则再从指定的代码基地实时下载。(3)预安装模式，要求在agent迁移之前其所需的代码都必须预安装到目的节点平台上。上述三类模式中，重量级方式的优点在于当agent抵达目的服务器后，在执行时就不再需要通过额外的通信来取得执行代码，缺点是可能会传输一些不必要的(如在目的节点已经存在的)代码，从而增大了传输量。轻量级方式可以减少传输量，但实时代码下载会影响执行速度，且不适合在执行时网络连接会发生断开的情形。预安装模式省去了在agent迁移和执行时的代码传输，但需要预先了解agent在各个节点的执行状况。

迁移控制：agent在各节点之间的迁移是有目的和受任务驱动的，其控制方式通常有如下两种，(1)单步迁移指令控制，即在满足迁移条件时，通过调用一条以下一节点地址为参数的迁移指令来完成从当前节点到下一节点的移动。Agent TCL[6]、Mole[10]、Voyager[5]和Grasshopper[11]中采用了这种方式。(2)旅行计划(Itinerary)控制，即将agent要经过的节点预先保存在agent自身携带的一个称为旅行计划的结构当中，旅行计划中的每一个旅行步骤对应agent任务执行过程中的一次迁移，为了适应不同的应用需求，agent也可能会在其任务执行过程中动态改变其旅行计划。Ajanta[12]、Aglets[3]、Concordia[4]和Mogent[13]均采用了这种迁移控制方式。

2.3 通信支持

移动agent系统中的通信主要包括平台之间的通信、agent与平台间的通信以及移动agent相互间的通信。平台之间的通信与传统的分布式环境中两个固定网络节点之间的通信没有差别，因此我们在这里只讨论后两种情形，即参与通信的双方中至少有一方是具有自主迁移能力的agent。通常，为了充分利用agent的异步执行特性，agent系统通常都以异步消息传送作为主要的通信模式，而面向连接的同步通信手段在移动agent系统中通常只作为补充机制，用来完成少数实时信息的传输。当前移动agent系统中，agent间的异步通信模式主要有以下两种：(1)代理模式，分单点代理和多点代理两种。单点代理模式中，消息被首先发往一个特定的代理节点(如agent的出生节点)，再由其转发至目标agent。Aglets[3]和Ajanta[12]采用这种模式，其优点是实现简单，但对单个节点的依赖使得通信效率、容错性和异步执行都比较差。多点代理模式中，agent在经过的每一节点都产生一个代理，用来记录下一代理节点的地址，消息将沿代理路径被发往目标agent。Voyager[5]，April[8]和Mole[10]均采用这种模式，它对agent的迁移限制较少，且通信可靠性有所增强，缺点是通信成本高，且任一代理节点的失效都会导致通信失败。(2)邮件模式，即由专门的邮箱结构或邮件服务来完成消息转发。Messenger[7]采用这种模式，与单点代理模式类似，它在通信效率、容错性和异步执行等方面有所欠缺。

以上通信机制的共同缺陷是没有解决agent迁移所引起的消息丢失，因此就有一些专门针对这一问题而提出的通信算法，它们根据保障可靠性所采取的手段不同可分为三类：(1)对agent的迁移和消息接收进行同步。Mogent系统[13]通过引入“迁移状态”和“在途信件数目”两个信号量来对通信和移动所共享的位置信息进行同步，但它在保证可靠性的同时过多限制了agent的迁移，且由于每发一封信都需进行寻址，通信效率低。ARP[14]算法中，每个agent有一个用来接收消息的邮箱，通过对邮箱移动和消息转发进行同步来保证通信的可靠，且通过agent与邮箱的分离来减少对agent迁移的限制，其缺点在于通信量会随邮箱的移动而急剧增加，且容错性较差。(2)获取agent收信状况的全局知识。A. Murphy等[15]提出一个基于广播的可靠传输算法，它实质上是由分布式全局快照算法演化而来的。它不限制agent的迁移，且只需少量改动就可以支持针对agent群组的通信，其缺点在于全局广播造成了通信成本高和扩展性差。(3)借助下层的可靠通信支撑。M. Ranganathan等[16]通过在UDP上构造一个类似TCP的协议来实现agent间消息的可靠传输。它利用一个位置管理器来跟踪和广播参与通信的各移动端点的位置，并采用类似TCP中滑动窗口协议的办法对通信双方的收发操作进行协调，且通过重发来保证可靠通信。其缺陷主要是算法实现困难，可用性差。

3 其它支持

3.1 名字服务和目录服务

在agent的迁移和通信过程中，以及在实际的应用开发中，通常都会用实体的名字来代替其实际的物理地址，因此就需要系统能够提供名字到地址之间的映射。这一功能通常是由名字服务实现的，它能够对agent系统中的相关实体(如agent，agent平台或其它资源)统一进行注册，并能够把各实体的名字映射为其实际的物理地址。名字服务的基本实现通常可以用名字策略来描述，名字策略定义了一个名字空间，它给出了跨网络节点的整个系统范围内各种实体的命名规范。在移动agent系统中，一个良好的命名策略应该能够满足如下条件：(1)全局唯一性，即实体在可能的应用范围内(如Internet)具有全局唯一的标识，这样就使得相互独立开发的应用之间不会发生名字冲突，且能够方便地共享相同的实体。(2)位置透明性，即实体的名字与实体所处节点的物理位置无关，即使实体的物理位置发生了改变(如agent的迁移)，实体

的名字也可以保持不变。这就使得开发者在进行应用开发时，不必考虑资源的具体位置。目录服务可以看作是名字服务的扩展，即它除了提供实体的存在性注册和向物理地址的映射外，还可以包括实体的一些相关分类和属性，它可以通过在名字服务上附加分类功能来提供针对功能特性的黄页查询。

当前的 agent 系统实现中，相当一部分系统没有提供名字和目录服务，需要硬编码实体的物理地址，也有一些系统[5,13]由各个 agent 的出生节点承担名字服务的功能，agent 的名字由出生地节点地址和在出生地具有唯一性的一个字符串组成，名字到当前物理地址的映射由各个出生地节点完成。个别系统[8,11]参照 FIPA[18]和 MAF[19]中对名字和目录服务的建议构造了便于和其它系统互通的名字和目录服务。

3.2 安全支持

当移动 agent 系统处于一个开放的网络环境中时，就必须提供相关的安全服务支持。根据受保护对象的不同，可将其安全问题分为如下三类[8,13]：(1)保护主机节点及主机上其它 agent 的安全，主要是防范恶意的或者是有错误的 agent 利用主机上安全设施的不足发起针对主机和在主机上运行的其它 agent 的攻击，攻击的主要形式有伪装、拒绝服务、未授权访问以及抵赖。在 Agent TCL[6]和 Aglets[3]等系统中针对这类攻击采用了运行环境隔离、访问控制、加密、签名及操作记录等防范措施。(2)保护一组节点资源，agent 可以在单个节点上只消耗少量的资源，且行为也完全符合站点的安全策略，但从整个网络来看，其消耗的资源却是非常巨大和难以容忍的。这类安全问题必须从网络整体来考虑，如果各节点从属于一个统一的管理者时，比较容易在整体上对资源的使用进行控制，但当在 Internet 这样的开放网络中时，则需要通过限制 agent 的迁移次数和子 agent 的产生个数，或者像 Mogent[13]系统一样，采用电子货币的形式对 agent 的资源消耗进行限制。(3)保护 agent，即防范恶意主机对 agent 进行破坏。由于对主机来说，运行于其上的 agent 的代码、数据以及运行时刻的通信都是暴露的，所以这类安全问题是最难解决的。目前采用的手段包括部分结果封装和加密、采用多个 agent 沿不同路径执行相同任务以及路径记录等，但这些手段都只能保证一定程度的安全。

3.3 容错与恢复

为了增强系统的稳定性和可用性，一部分移动 agent 系统提供了容错与恢复机制，这里的容错与恢复主要是指针对 agent 或节点可能发生的崩溃而采取的一些预防和恢复措施。当前较为常见的是检查点恢复机制，即在 agent 执行过程中的某些关键点将 agent 的执行状态进行冻结并保存至永久存储设备当中(在 Java 中可以利用对象的序列化来实现)，当应用或系统出现故障时，能够将 agent 和应用的状态恢复到某一关键点。Concordia[4]和 Voyager[5]均采用了这种容错机制。

3.4 互操作支持

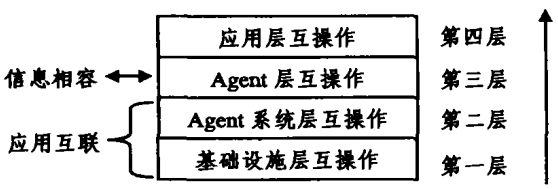

图 1 agent 环境下的互操作研究框架

由于应用环境的相似性，在移动 agent 环境下同样要对传统分布式应用下所碰到的互操作问题加以考虑，此外，由于 agent 特有的移动性、自主性和智能性等特点，它对互操作也提出了一些新的要求。简单地讲，移动 agent 系统中的互操作就是用来解决整个移动 agent 计算环境下各个组成部分(不同的 agent 平台、不同的自主 agent、不同应用)之间的相互联系和协作问题的。

如图 1 所示，移动 agent 计算环境下的互操作可分为四个层次[17]，最底层的基础设施层互操作是其它各层的基础，它主要提供由不同开发者完成的 agent 系统之间在传输服务(包括消息的传输和 agent 自身的传输)、命名和目录服务、安全服务以及 agent 状态捕捉与恢复机制方面的协调和互通。作为第二层的 agent 系统层互操作用来对 agent 系统中一些基本操作的执行语义和执行步骤加以规范，主要面向跨多 agent 系统的管理。agent 层互操作主要是面向消息语义的，它包括以下两方面逐次递进的内容：(1)各系统的 agent 间具有明确语义的单个消息的传递和理解。(2)会话语义的理解，即在单个消息语义理解的基础上，对一组具有特定关联的消息集合的语义理解，这样一组消息通常具有共同的语义背景和执行目标，能够表达单个消息难以表达的复杂语义。应用层互操作包括两方面：(1)不同 agent 应用之间(其中每个 agent 应用可以包含多个 agent 或 agent 群组)的相互关联和互操作。解决的基本思想是把它们看作是需要进行简单合作的两个智能群体，并通过通信语义上的共享来实现这种合作。(2) agent 应用与非 agent 应用(如数据库系统、Web 浏览器及各种行业应用等)之间的相互作用。它应该既包括 agent 应用对传统应用的调用也包括传统应用对 agent 应用的调用，通常可以通过包裹(Wrapping)机制来对传统应用的调用接口进行包装来实现。

当前关于 agent 系统互操作性的研究主要集中于 OMG 的 MAF[19]以及 FIPA[18]的一组规范。MAF 通过定义一组 agent 标准化的接口来规范不同 agent 系统在基础设施层和 agent 系统层的互操作，FIPA 规范族以 agent 层的互操作为核心，定义了一组规范来提供对上面各个层次互操作的支持。在当前实际的移动 agent 系统实现中，April[8]提供对 FIPA[18]规范的部分支持，Grasshopper[11]提供对 MAF[19]的支持。

3.5 开发与管理工具支持

由于移动 agent 系统的用户需求千变万化，就需要系统能够为用户提供灵活方便的开发工具，来帮助用户快速地完成 agent 应用的开发和部署。现有系统中比较典型的有 Aglets 中的 Java Aglet 应用编程接口(J-AAPI)[3]和 ZEUS[20]工具包，J-AAPI 能够提供基本的 agent 创建、初始化、部署、回收、复制以及 agent 间的消息传递。ZEUS 工具包的核心是一组基于 ZEUS agent 系统架构的 Java agent 组件，这些组件从功用上可以划分为三类。一类为基本 agent 组件，用来提供通信、计划与调度、内部数据结构表示以及行为设定的能力；第二类为应用开发辅助工具，包括可视化工具、代码生成以及遗产系统接口生成等；还有一类用来提供一些公用的设施，如名字服务、统计分析、运行状态报告等。在基本的开发工具基础上，agent 系统通常还会提供一些辅助工具用来管理整个网络范围内的 agent 系统和监控 agent 运行状况。

总结和展望 本文分析了移动 agent 系统的平台支撑，指出了在系统构建时需要着重考虑的几个问题，并对照现有的 agent 系统实现对这些问题作了深入讨论。

从应用的角度来看，移动 agent 技术仍然处于起步阶段。在系统平台支持方面，当前移动 agent 技术所面临的挑战主要有以下几方面：(1)安全和容错仍然是移动 agent 系统平台构建中最具挑战性的问题。安全方面主要的难点在于很难保证不断变换执行环境的运行主体—agent 的安全，目前的各种安全措施也只能是在 agent 受到攻击后检测出攻击行为和攻击者。在容错方面，绝大多数系统都只提供基于检查点的静态恢复机制，即通常都需要暂停 agent 应用的执行，再通过一定程度的人工干预来重新恢复 agent 的运行，而对运行期间的动态恢复支持很少。(2)缺乏支持大规模 agent 应用的开发和调试工具。当前基于各个移动 agent 系统构建的应用几乎都是局限于很少数量 agent 的小规模示范应用，因此迫切需要系统能够提供面向大规模企业或科学计算应用的开发和调试工具，同时，agent 数量的增加和应用规模的扩大也对系统在 agent 监控和管理、agent 群组通信、系统安全等方面提出了更多的要求。(3)提供全面的互操作支持。随着移动 agent 技术的发展和越来越多 agent 系统的构建，不同系统之间、不同 agent 之间以及不同应用之间的互操作需求也更加迫切，但现有的多数 agent 系统或者根本不提供对互操作的支持或者只是提供图 1 中个别层次的实现，为了支持基于 Internet 环境的应用，就有必要提供覆盖图 1 中各个层次的全面的互操作支持。

除了前面各部分提到的一些主要的系统支持外，随着 Web Services 的兴起，移动 agent 系统也有必要提供面向 Web Services 的应用开发和运行支持。文[21]和文[22]中给出了 agent 通信语言(ACL)和 SOAP 消息之间的映射，给出了如何将 agent 以 Web Services 的形式进行封装、部署和查找以及如何封装 Web Services 以便让 agent 进行调用的一些建议，这些建议为实现 agent 系统级的 Web Services 支持提供了一些可行的参考。

参 考 文 献

1 David C, Colin H, Aaron K. Mobile Agents: Are They a Good Idea? IBM Research Division. Technical Report: RC 19887, 1995

2 Lange D B. Mobile Objects and Mobile Agents: The Future of Distributed Computing? Lecture Notes in Computer Science, 1998,1445:1～12

3 Lange D B, Oshima M. The Aglet Cookbook. http://www.trl.ibm.co.jp/aglets

4 Wong D, Paciorek N, et al. Concordia: An Infrastructure for Collaborating Mobile Agents. Mobile Agents - First International Workshop, MA '97

5 ObjectSpace. ObjectSpace Voyager Core Package Technical Overview:[Technical Report]. ObjectSpace, Inc. , July, 1997. http://www.objectspace.com/

6 Robert S, Gray. Agent Tcl: A flexible and secure mobile-agent system. Fourth Annual Tcl/Tk Workshop (TCL 96), Jul. 1996

7 Fukuda M, Bic L, et al. Distributed Coordination with MESSENGERS. Science of Computer Programming Journal, 1998,31(2)

8 Fujitsu Laboratories of America. April Project. Fujitsu Laboratories of America, Inc. http://www.nar.fujitsulabs.com/april/

9 Peine H, Stoplmann T. The Architecture of the Ara Platform for Mobile Agents. Mobile Agents - First International Workshop, MA '97.

10 Baumann J, Hohl F, et al. Mole - Concepts of a Mobile Agent System:[Technical report]. University of Stuttgart, Aug. 1997

11 IKV++ GmbH. Grasshopper Basics And Concepts, 1998. http://www.grasshopper.de/

12 Tripathi A, Karnik N, et al, Mobile agent programming in Ajanta. In: Proc. of the 19th Intl. Conf. on Distributed Computing Systems (ICDCS'99), May 1999. 190～197

13 陶先平. 基于 Internet 的移动 agent 技术和应用研究:[南京大学博士学位论文]. 2001

14 Feng X, Cao J, et al. An Efficient Mailbox-Based Algorithm for Message Delivery in Mobile Agent Systems. In:Proc. of the 5th IEEE Intl. Conf. on Mobile Agents (MA2001), 2001

15 Murphy A , Picco G P. Reliable Communication for Highly Mobile Agents. In: Proc. of the Agent Systems and Architectures/Mobile Agents (ASA/MA)'99, 1999. 141～150

16 Ranganathan M, Bednarek M, et al. A Reliable Message Delivery Protocol for Mobile Agents. In:Proc. of Agent Systems and Architectures/Mobile Agents (ASA/MA)'2000, 2000. 141～150

17 贾志勇，景广军，谢立. Agent 互操作性研究. 计算机科学，2003(3)

18 The Foundation for Intelligent Physical Agents. FIPA Specifications. 2001, 8. http://www.fipa.org/specifications/index.html

19 Object Management Group. Mobile Agent Facility Specification. 2000, 1. http://www.omg.org/technology/documents/formal/mobileagentfacility.htm

20 Hyacinth N, Divine N, et al. ZEUS: A Tookit and Approach for Building Distributed Multi-agent Systems. In:Proceedings of the Third International Conference on Autonomous Agents (Autonomous Agents'99), 1-5

21 Avila-Rosas A, Moreau L, et al. Agents for the Grid: A Comparison with Web Services (part I and II). Workshop on Challenges in Open Agent Systems. July 2002

22 Ankolekar A, Burstein M, et al. DAML-S: Semantic Markup for Web Services. In: Proc. of Intl. Semantic Web Working Symposium (SWWS), 2001