移动 Agent 互操作性研究*)

The Research of the Interoperability of Mobile Agent

张云勇　刘锦德

（电子科技大学计算机学院　成都610054）

Abstract　Mobile Agent has taken a very momentous influence on some fields such as collaborative system and workflow management system because of its unique technical advantages. Until today, a large number of mobile Agent platforms have been developed but it is very difficult to collaborate those different Agent systems since the difference among their design structure and the technology which they take. Solving the interoperability of mobile Agent becomes the primary task. In this paper, the specification of MASIF is studied in detail, a clue of solving the interoperability is put forward and an addon for the interoperability of mobile Agent system called MASIF addon is developed by ourselves based on the specification.

Keywords　Mobile Agent, Interoperability, CORBA services, MASIF addon, FIPA

1. 引言

Agent 的研究起源于人工智能领域，是指模拟人类行为和关系、具有一定智能并能够自主运行和提供相应服务的程序。随着网络技术的发展，可以让 Agent 在网络中移动并执行，完成某些功能，这就是移动 Agent 的思想。移动 Agent 由于具有任务异步执行、减轻网络负载、健壮性、并行处理、智能路由等特性[1]，自90年代初期以来，对许多领域产生了非常重大的影响。国外一些主要的厂商分别推出 Agent 系统[2]，然而由于它们采用的技术及设计框架的差异，使得它们难于很好地协调工作。正是在这种背景下，OMG 组织制定了 MAF（Mobile Agent Facility）规范，随后改为 MASIF（Mobile Agent System Interoperability Facility），并于98年三月正式推出[3]，它为解决不同厂商间 Agent 系统的互操作性提出了一些基本建议。

本文首先研究 MASIF 及其相关规范，并提出了解决 Agent 系统互操作性的思路。随后，给出一个 Agent 系统互操作插件 MASIF addon 的具体框架及其实现，并对此插件的 FIPA 扩展提出了解决方案。

2. MASIF 及其相关规范

2.1　互操作性的概念及 MASIF 术语

互操作性是不同 Agent 系统协调工作的基本特性，也是开放系统的必具特征。关于 Agent 互操作性，我们参照了文[4]，可以这样理解：当一 Agent 应用在网上某节点运行时，它应该可以动用网上其它节点的数据、处理能力和类似资源；如可将自己的部分任务委托其它节点上的 Agent 来帮忙完成，这时即使各节点上的 Agent 是异种的。

上述定义主要强调了 Agent 在异种环境中的协调工作，MASIF 规范正是为此而推出的。为了更好地理解 MASIF 规范，下面先阐述一些术语[5]。

·Agent 状态：由 Agent 执行状态（程序计数器和堆栈）和决定 Agent 到达目的地重新恢复执行时动作的 Agent 属性值的集合。

·Agent 名字：由权限、标识符和 Agent 系统类型组成。

·Agent Place：也称 context，是 Agent 执行环境。

·Agent Location：是 Agent place 的地址。

·Agent 区域（Region）：指具有相同权限但不一定属于同一个 Agent 系统类型的集合。

2.2　互操作性的要求[5]

为了达到 Agent 系统的互操作性，显然需要对移动 Agent 中的主要结构和概念加以规范化、标准化。MASIF 规范建议必须对 Agent 管理、Agent 迁移、Agent 以及 Agent 系统名称、Agent 系统类型以及位置语法标准化。

2.2.1　Agent 管理　必须提供创建、挂起、重新执行以及中止 Agent 的标准化方法，使得管理员可以

*)本课题得到国防预研基金的资助。张云勇　博士生，主要研究方向：开放系统与中间件技术、Agent 技术。刘锦德　教授，博导，主要研究方向：开放系统与中间件技术、网络多媒体技术、实时分布处理系统技术。

管理不同类型的 Agent 系统。

2.2.2 Agent 迁移 包括初始化 Agent 迁移、接收 Agent 迁移和类迁移。

1)初始化 Agent 迁移。当目的地接收到 Agent 迁移请求时，必须初始化以下一些动作：挂起正在执行的 Agent；标识将要迁移的 Agent 状态；序列化要迁移 Agent 类与状态的实例；为特定 Agent 迁移协议将上面序列化的 Agent 进行编码；认证客户端；迁移 Agent。

2)接收 Agent 迁移。在一个 Agent 被目的系统正式接收以前，该目的 Agent 系统必须判断它能否解释这个 Agent，如果能够解释，就接收这个 Agent，并且：认证客户端；对 Agent 进行解码；对 Agent 类及状态进行反序列化；实例化 Agent；恢复 Agent 的状态；恢复 Agent 的执行。

3)Agent 类的传输(Transportation)。主要用在面向对象的 Agent 系统，有如下四种情况：自动迁移所有的类；只自动迁移 Agent 类，其它的类则是按需迁移；迁移 Agent 的时候，只自动迁移 Agent 类，其它的类则是按需迁移；而当远地创建 Agent 的时候，自动迁移所有的类；接收到创建 Agent 或者迁移请求时，迁移特定名字的类。

2.2.3 Agent 以及 Agent 系统的名称 必须提供产生唯一标识符(类似于 COM 中的 GUID 和 CORBA 中的 IOR 地址)的机制，让 Agent 间可以互相标识。

2.2.4 Agent 系统类型及位置语法 标准化 Agent 系统类型，主要是提供 Agent 系统的一些信息(如实现语言、产品名称)，以使目的 Agent 系统判定能否支持此系统。位置语法的标准化则是为了能够让不同 Agent 系统有效地定位对方。我们认为还需提供 Agent 跟踪机制的标准化，如注册与去消、查询等，当然安全机制的标准化也是必需的。

3. MASIF-addon 插件的框架结构

在仔细研究了 MASIF 规范后，我们实际开发出了一个移动 Agent 互操作性的插件，此插件符合 MASIF 规范，并且还得到一定的扩展。

如图1，整个框架由 DAE(分布式 Agent 环境，即图中的 Region)和 DPE(分布式处理环境，为图中通信通道及下面的部分)组成，其中DAE由注册组件和Agency 所组成。

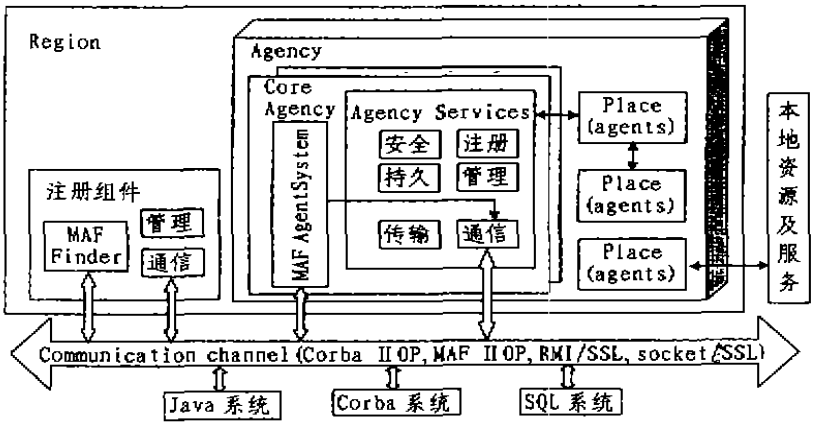

图1 MASIF 插件设计框架

3.1 Agency

我们这里的 Agency 相当于 MASIF 中的 Agent 系统，每个 Agency 包括两个部分：一个核心的 Agency 以及一个或者多个 place。核心 Agentcy 提供了执行 Agent 的最小功能集合，主要包括以下一些组件或服务：

·安全：主要用 SSL 支持外部安全机制，以提供保密性、完整性和认证功能。用访问控制支持内部安全机制[6]。

·注册：注册一个 Agency 内部所有 Agent 的信息，以使得 Agent 之间能够有效地找到对方、与对方交互信息以及达到监控 Agent 的目的。

·持久性：本机制提供 Agent 在 Agency 之间迁移时由于某些原因迁移失败、Agent 所驻留的机器意外崩溃等情况下，有效保存 Agent 相关数据的功能。

·管理：负责创建、删除、挂起、恢复执行 Agent 以及拷贝 Agent 的管理。

·传输：负责 Agent 状态的串行化、通过 Agency 通信服务迁移 Agent 到目的 Agency 并在目的地重新启动 Agent。

·通信：负责不同 Agency 之间 Agent 的迁移，主要由上述传输服务激发，提供同步、异步、动态及多播模式。

·MAFAgentSystem：是 Agency 提供的主要接口，也是 MASIF 规范建议必须支持的接口，具体负责接收 Agent、列出 Agents/places、获得 MAFFinder 接口、获得 Agent 系统类型、获得 Agent 状态等。

3.2 注册组件

它提供了 Agency 注册的另一种有效的机制，负责管理所有已经注册的 Agency 的信息。为达到此目的，必须提供以下的服务：

·管理：主要提供区域内 Agency 的定位。

·通信：与 Agency 内部的通信机制类似，只不过它提供区域间以及本地 Agency 与远地 Agency 间的交互。

·MAFFinder：是本组件里主要的接口，也是 MASIF 规范建议必须提供的接口。提供注册、注消、查询 Agency 等服务，它实际上为一名字服务。

4. MASIF 插件的具体实现

这里只提供最主要的实现，图2(图1的简略图)中主要包括 MAFAgent 系统接口和 MAFFinder 接口，从客户端发出请求，到执行完成，主要应有以下步骤：

·初始化，主要由 MAFFinder 接口利用 register_Agent()完成一些相关信息的注册。

·由 MAFAgent 系统接口利用 receive_Agent()方法接收到客户端 Agent 的请求。

随后利用 get_Agent_info()得到 Agent 的类型，并利用 get_authinfo()进行认证。

·如果认证成功，MAF Agent 系统接口利用 get_MAFFinder()获得 MAFFinder 接口的引用。

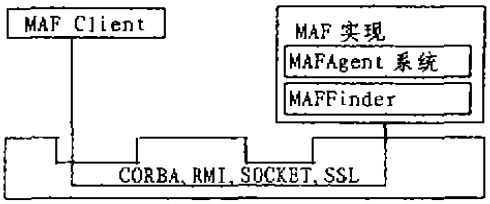

图2 MASIF 插件的简略图

·MAFFinder 接口利用注册库中的相关信息，并利用 lookup_place ()，lookup_Agent_system ()，lookup_Agent ()方法得到客户端请求中所迁移目的地的信息。

·客户端 Agent 迁移到目的地，交互完成工作，然后返回。

由上面的分析知道，为了使得 MASIF 插件达到移动 Agent 的互操作性，我们必须提供步骤中所列出的接口及其方法，我们利用 IDL 语言定义了如下的接口：

```
//System.idl，一些系统定义
struct Name{
OctetString authority;OctetString Identify;
short Agent_system_type,//0表示非 Agent 系统;1表示 Aglet;2表示 MOA;3表示 AgentTCL
};
typedef string Location,
struct AgentSystemInfo{Name system_name;short system_type;//系统类型同上
PropertyList properties;};
struct AgentProfile{
LanguageID language_id;//0表示没有定义的语言;1表示 Java;2表示 TCL;3表示 Scheme;4表示 PERL,本插件暂时只针对 Java 语言，但是留有扩展余地
short Agent_system_type;//系统类型同上、
SerializationID serialization;//0表示没有定义的串行化;1表示 Java 对象串行化方法。
PropertyList properties;};
//MAFAgentSystem.idl
interface MAFAgentSystem{
Name create_Agent (//…) raises (ClassUnknown, ArgumentInvalid, DeserializationFailed, MAFExtendedException);
OctetStrings fetch_class(in ClassNameList class_name_list, in string code_base,in AgentProfile Agent_profile) raises (ClassUnknown,MAFExtendedException);
Location find_nearby_Agent_system_of_profile (in AgentProfile profile)
  raises(EntryNotFound);
AgentSystemInfo get_Agent_system_info();//得到诸如 Agent 系统类型，验证等信息
MAFFinder get_MAFFinder() raises (FinderNotFound);
NameList list_all_Agents();
Locations list_all_places();
void receive_Agent(//…)raises(ClassUnknown,DeserializationFailed,MAFExtendedException);
void resume_Agent (inNameAgent_name)raises (AgentNotFound,ResumeFailed,AgentIsRunning);
void suspend_Agent(inNameAgent_name)raises(AgentNotFound,ResumeFailed,AgentIsRunning);
void termiante_Agent_system() raises(TermunationFailed);
  };
//MAFFinder.idl
interface MAFFinder{
void register_Agent (in NameAgent_name, inLocation Agent_location, in AgentProfile Agent_profile ) raises (NameInvalid);
void register_place(in string place_name,in Location place_location) raises (NameInvalid); Locations lookup_Agent (in Name Agent_name, in AgentProfile Agent_profile) raises (EntryNotFound);//有四种方法可以定位 Agent，强制搜索、跟踪日志、Agent 注册以及 Agent 发布，本插件提供注册机制
void unregister_Agent (in Name Agent_name ) raises (EntryNotFound);
  };
```

其它的如异常信息等内容，由于篇幅所限，这里不再列出。

5. MASIF addon 的 FIPA 扩展

在 Agent 领域，除了 OMG 组织，FIPA(Foundation for Intelligent Physical Agents)也是一个非常重要的组织，FIPA 相应的规范中很好地体现了 Agent 的智能性[3]。

FIPA 的参考模型可以用图3描述。目录服务设施(Directory Facilitator，DF)、Agent 管理系统(AMS)和 Agent 通信通道(ACC)提供了管理 Agent 的一些功能。其中 AMS 提供平台内的 Agent 通信，ACC 提供了

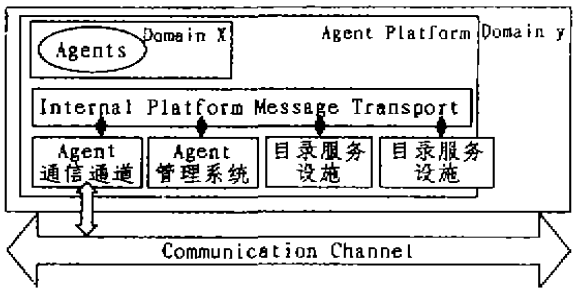

图3 FIPA 的模型图

平台间的 Agent 通信。内部平台信息传输(InternalPlatform Message Transport,IPMT)提供了消息路由服务。整个 ACC,AMS,DF 和 IPMT 构成了 Agent 平台。

因为 ACC 是整个系统的核心,而 ACL(Agent 通信语言)及 ontology(共享语汇)又是 ACC 的核心,所以为了提供 MASIF 插件和符合 FIPA 规范的 Agent 系统间的互操作性,我们提出了以一个 IDL/ACL 网关为核心的解决方案,将 MASIF 插件提供的一些组件服务映射成 FIPA 的规范格式,其映射主要如下:

·MASIF 的区域到 FIPA 域的映射。

·MASIF Agent System 到 FIPA Agent 平台的映射。

·MASIF MAFFinder 到 FIPA 目录服务设施(FIPA DF)的映射。

·MASIF Agent System 到 FIPA AMS 的映射。

·MASIF 资源接口到 FIPA Agent Wrapper 的映射。

有了这些基本的映射机制,利用 IDL/ACL 网关,便可以实现与 FIPA 规范相符的智能 Agent 系统间的互操作。

6. 实例与总结

我们利用 Orbix Web 3.0[8]为 KOBALT[9]开发了一个移动 Agent 的符合 MASIF 规范的插件 MASIF addon,并且进行了测试。由于采用纯 Java 开发,我们的插件具有良好跨平台性。测试平台为 NT(Orbix Web3.0, Aglet[10]) 和 Solaris (Orbix Web3.0, KOBALT,MASIF addon)。测试模型如图4所示。

说明:该模型可视为连锁商店模型,设 Agent1 为某连锁商店成都分店的库房主管,Agent2为北京分店的库房主管,现在成都对某商品缺货,需查找北京有无此货。首先它利用 MAFFinder1找到北京主管的联系方式,然后创建产生一个新的 Agent 到北京分店的库房里查找,并把货物的信息发送给成都的主管。这里特别强调两个地方系统不一样,通过这个实验,我们验证了 MASIF 插件的有效性。

注意:平台2上实验的顺序是先启动 Orbix Web CORBA Services,然后启动 Agent 服务进程,再启动 MASIF 插件,最后启动实验程序。

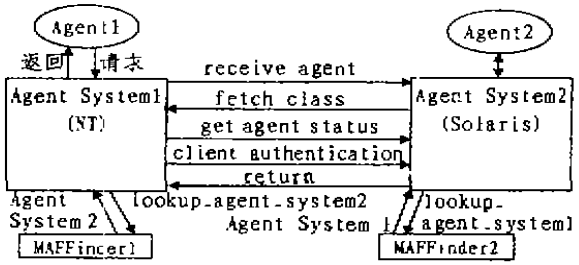

图4 利用不同 Agent 系统的互操作性,完成查找功能的连锁店模型

应该说,本插件的开发,为解决移动 Agent 间以及移动 Agent 和智能 Agent 间的互操作性迈出了重要的一步。进一步的工作是需要在插件中有机地融合 FIPA 的特性,具体实现 IDL/ACL 网关;另外 MASIF 规范中没有提出区域间 Agent 的互操作,我们建议用接口 ExtendMAFFinder 来解决此问题,当然多区域间安全机制也需加以考虑。

移动 Agent 不同于基于过程的 RPC(如 OSF/DCE),也不同于面向对象的分布式应用(如 OMG/CORBA,OLE/DCOM 和 Java/RMI),其独特的对象传递思想和优秀的特性给分布式计算乃至开放系统带来了巨大的革新。相信有效地解决移动 Agent 系统的互操作性,对于移动 Agent 系统的进一步应用会有一个极大的促进作用,并在移动计算、信息服务等新兴应用领域里找到用武之地。

参 考 文 献

1 张云勇 面向 Agent 的软件工程:[电子科技大学计算机学院博士生研究报告]. 2000
2 Agent Construction Tools. Available at:http://www.Agentbuilder com/AgentTools/index.html
3 Mobile Agent Standard. Available at:http://www.stud.ifi.uio.no/~sverreh/oppgave/node56.html,Nov. 1999
4 刘锦德.对于开放系统内涵的澄清 计算机应用,1997,17(6):1~2
5 OMG Joint Submission. Mobile Agent System Interoperability Facility. Nov. 1997. Available at: ftp://ftp.omg.org/pub/docs/orbos/97-10-05.pdf
6 OMG CORBAservices: Common Object Services Specification,Nov 1997
7 FIPA. FIPA98DraftSpecification: PartII: AgentManagementSupport for Mobility,FIPA8415,Version 0.3,Foundation for Intelligent Physical Agents,April 1998
8 OrbixWeb 3.0 Beta Progammer's Guide September, 1997. Available at,http://www.iona.com/
9 KOBALT,available via http://www.inrt.fr/,1999
10 Oshima M, Karjoth G, Ono K. Aglet Specification 1.1 Draft. Available at, http://www.trl.ibm.co.jp/aglet/specII.html,1998