一种基于 CORBA 和 JNDI 技术的通用 MIB 访问接口的设计

王 宇, 李增智, 薛文革, 杨振江
(西安交通大学计算机系统结构与网络研究所, 陕西西安 710049)

摘 要: 本文提出了利用 JNDI(Java Naming and Directory Interface, Java 名录访问接口) 与 CORBA(Common Object Request Broker Architecture, 公共对象请求代理构架) 名字服务相结合实现 MIB(Management Information Base, 管理信息库) 与 DIT(Directory Information Tree, 目录信息树) 在 API(Application Programming Interface, 应用编程接口) 级的集成构架, 集成的关键是用 CORBA 名字服务建立一棵基于 DN(Distinguished Name, 绝对识别名) 的名字树. 本文对如何利用 CORBA 名字服务构建 MIT(Management Information Tree, 管理信息树) 名字树的问题进行了详细讨论.

关键词: 目录信息树; 管理信息库; 目录服务
中图分类号: TP391 文献标识码: A 文章编号: 0372-2112 (2001) 08-1117-04

Design of Generic MIB Interface Based on CORBA and JNDI

WANG Yu, LI Zeng-zhi, XUE Wen-ge, YANG Zhen-jiang
(*Inst. of Computer Architecture & Networks, Xi'an Jiaotong Univ., Xi'an, Shanxi 710049, China*)

Abstract: The paper presents a framework for integrating the MIB and DIT by using JNDI API and CORBA Naming Service. The key approach of integration is to build a naming tree based on Distinguished Name using CORBA Naming Service. In addition, more details on how to set up a naming tree of MIT by using CORBA Naming Service are stated and discussed.

Key words: directory information tree; management information base; directory service

1 引言

在 OSI(Open System Interconnection, 开放系统互连) 网络管理模型中, 需要维护的最重要的信息资源是管理信息库. 传统上, 对管理信息库的维护和访问是通过专用的管理协议, 如 SNMP(Simple Network Management Protocol, 简单网络管理协议), CMIP(Common Management Information Protocol, 公共管理信息协议) 等进行的. 采用专用的管理协议访问管理资源的缺点如下: ①对被管信息资源的访问受限于网络管理应用, 企业中的其它应用如果需要相同的信息资源, 则必须有该信息资源的不同形式的副本, 这可能造成企业网络上管理信息资源的不一致性; ②随着网络技术的发展, 网络上的所有资源, 包括用户, 网络, 交换机等资源都将由企业目录服务统一管理, 但对 SNMP, CMIP 而言, 网元的 MIB 是独立于企业目录而作为企业信息库中的一个信息孤岛而存在, 即 MIB 不支持公共的目录访问协议, 如 LDAP(Lightweight Directory Access Protocol, 轻量目录访问协议) 等, MIB 中的对象必须通过特殊的网络管理协议来访问. 上述缺点表明, 如果企业网络需要采用统一的方式维护网络上可管理的信息资源, 必须实现 MIB 与 DIT 的集成, 即 MIB 必须成为 DIT 树的一棵子树. 本文主要讨论 MIB 与 DIT 的集成技术.

2 MIB 与 DIT 集成的途径

实现 MIB 和 DIT 的集成主要有两种方法: ①迁移方法. 该方法是将 MIB 信息导入 DIT 中, 从而实现通过目录服务来访问 MIB 中的信息, 该方法的缺陷在于网络中存在两个内容相同的信息资源库, 两者的一致性维护将成为一个很重要的问题; ②目录访问接口方法. 该方法是使得 MIB 提供目录访问接口, 即将 MIB 作为企业信息库的一棵虚拟子树. 这种方法的好处是管理协议代理(SNMP 或 CMIP) 可以与其它应用采用相同的目录接口来访问 MIB 信息. 本文主要讨论第二种方法.

3 MIB 与 DIT 的集成构架

MIB 中维护的信息分为两类: ①永久性信息, 如配置信息; ②临时性信息, 如状态, 性能信息. 永久性信息必须存放在永久性存储器中, 如文件, 数据库等, 对于提供目录服务的网络, 配置信息存放在目录服务器中是一个好的选择, 因为目录服务器可以提供很好的按名访问特性和较好的安全特性.

为了将 MIB 集成进企业级 DIT 中, 可以采用图 1 所示方法. MIB 中的配置信息存放在目录服务器中, 当网络设备初始化时, 网络设备通过 LDAP 协议访问目录服务器, 根据目录服

收稿日期: 2000-09-18; 修回日期: 2001-01-10
基金项目: 国家 863 重点项目(No. 863-511-946-008)

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

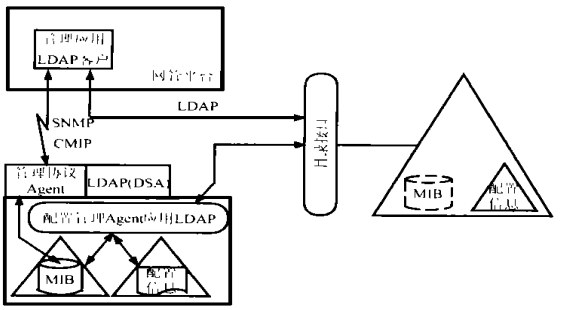

图 1 MIB 与 DIT 的集成构架

务器中的配置信息建立一个 MIB, 同时为了实现 MIB 与 DIT 的集成, 在目录服务器中建立 MIB 的一个映象, 作为 DIT 树的一棵子树. 当上层应用通过目录服务访问 MIB 树时, 所有对 MIB 树的访问都被重新定位到网络设备的实际的 MIB 上, 此时网络设备必须支持 DSA(Directory Service Agent, 目录服务代理), 以实现 DIT 中 MIB 子树的按名访问.

因为在网络设备中实现 DSA 相对比较复杂, 为了避免在网络设备上实现 DSA, 可以采用基于 JNDI 的折中方案如下. 假设 MIB 是在 CORBA 域中实现的, 可以利用 JNDI 和 CORBA 名字服务实现一个一致的 API 级目录访问接口. JNDI 方案的体系结构见图 2.

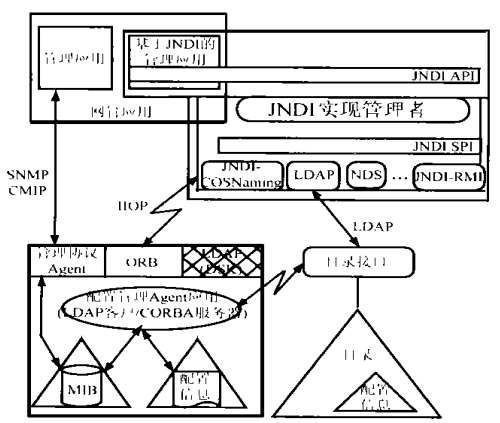

图 2 基于 JNDI 和 CORBA 名字服务的 MIB 目录访问构架

配置信息依然存放在目录服务器中, 当网络设备初始化时, 网络设备通过 LDAP 协议访问目录服务器, 根据目录服务器中的配置信息建立一个 MIB, 同时依据 MIB 建立一个基于 CORBA 名字服务的 MIB 名字树. 当上层的 Java 程序需要访问 MIB 中的信息时, 就通过调用 JNDI API 接口实现对基于 MIB 的 CORBA 命名树的访问. 这种方法实际上是在 JNDI API 级将 MIB 实现为一个虚拟 DIT 子树, 而不是在公共目录协议级将 MIB 映射为 DIT 的一棵子树. 采用 JNDI API 的好处是接口统一, 对各种目录访问协议, 上层用户所感受到的接口是一致的.

4 用 CORBA 名字服务实现具有目录功能的 MIB

在 JNDI API 级实现 MIB 与 DIT 的集成技术的关键是用 CORBA 名字服务建立一棵基于 MIB 的名字树. 下面描述如何将 OSI- SM(OSI System Management, 开放系统互连系统管理) 域内的名字映射成 CORBA 名字服务所需要的字符串格式的名字, 映射的结果是应用可以以统一的方式用 CORBA 名字服务接口和 JNDI API 访问 MIB 和其他目录服务中的信息.

这里主要解决如下问题: ①如何根据 GDMO(Guidelines for Definition of Managed Objects, 被管对象定义指南) 定义目录对象的模式, 将 GDMO/ ASN. 1(Abstract Syntax Notation One, 抽象语法记号 1) 管理协议描述规范映射到 IDL(Interface Definition language, 接口定义语言) 接口可以解决这一问题; ②如何将 MIB 中的对象名字映射到 CORBA 名字服务中字符串形式的名字; ③如何用 CORBA 名字服务的名子上下文接口建立名字树, 以及如何用名字上下文接口的操作来访问名字树中的对象.

4.1 GDMO 规范到 IDL 的映射

GDMO 到 IDL 的映射规范由 XoJIDM(X/ Open and NM Forum sponsored Joint Inter- Domain Management, 网络管理论坛, 域间管理委员会) 工作组制定, 可参阅相关文档[1].

4.2 MIB 向 CORBA 名字的映射

4.2.1 OSI 系统管理中的命名规则[2]

OSI-SM 中的对象名字遵循 X. 500 名字系统的层次结构(超- 属结构). 被管对象实例的 DN 是所包含对象实例的 RDN(Relative Distinguished Names, 相对识别名) 的有序序列. 由超类和属类对象所形成的整个名字树称为被管信息树(MIT) . 在超类 MO(Managed Object, 被管对象) 中, 所有的属类用 RDN 唯一的标识. 在 OSI- SM 域中, RDN 由单一的 AVA(Attribute Value Assertion, 属性值对) 构成. AVA 是一个名值对, 其中名字是对象的识别属性标识, 值是识别属性的值.

4.2.2 CORBA 命名规则[3]

CORBA 名字服务规范定义如何将一个名字和一个对象相关联(称为名字绑定) . 一个名字绑定总是相对于一个" 名字上下文" . 一个名字上下文是包含一系列名字绑定的对象, 其中每个名字都是唯一的. 一个名字是名字部件的有序序列. 一个名字部件由两个元素构成: id(标识) 和 kind(类型) . 由多个部件构成的名字称为复合名.

CORBA 和 OSI 命名规则对照表如下:

表 1 IDL 描述的 OSI 和 CORBA 命名规则对照表

OSI 名字规则	CORBA 名字规则
Module X711INF {	Module Naming {
struct AVAssertion {	Typedef string Istring;
ASN1_ ObjectIdentifier attributeType;	Struct NameComponent {
any attributeValue;	Istring id;
};	Istring kind;
typedef sequence< AVAssertion> RDN;	};
typedef sequence< RDN> RDNSequence;	typedef sequence < NameComponent >
typedef RDNSequence DistinguishedName;	Name;
union Name switch (long) {	……
case 1: RDNSequence rDNSequence; };	interface NamingContext {};
};	};

4.2.3 名字映射规则

通过比较 OSI 和 CORBA 的命名规则, 可以得到它们之间的映射规则如下: ①将 DN 映射到 Naming: : Name; ②将 RDN 的唯一的一个 AVA 元素映射到 Naming: : NameComponent; 如果在

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

RDN 中有多于一个 AVA 元素, 则在每个 AVA 元素之间使用"+"作为分隔符; ③每个 AVA 映射为类似"〈属性标识〉=〈值〉"的结构, 其中〈属性标识〉是 AVAssertion. attributeType 的字符串表示, 〈值〉是 AVAssertion. attributeValue 的字符串表示. 我们用转换的字符串来初始化 Naming::NameComponent. id, 而 Naming: NameComponent. kind 初始化为空串. 由于 Naming::NameComponent. id 的意义可以任意解释, 因而上述规则是合理的; ④AVAssertion. attributeType 中的 OID(Object Identifier, 对象标识) 被映射为相关属性模板 OID 的 IDL 范围名的字符串表示; ⑤AVAssertion. attributeValue 被映射为如下的字符串:

如果 attributeValue 是简单类型或 IDL 内建类型, 则用该值的字符串表示

如果 attributeValue 是 BIT STRING 或 OCTET 序列类型, 则我们将该值转换成十六进制字符序列

如果 attributeValue 是 SEQUENCE 或 SET 类型, 则用"{< 元素值> [,< 元素值>]* }"形式的字符串来表示, 其中< 元素值> 采用< 元素标识> < 空格> < 元素值> , 代表每个部件的字符串形式. 在< 元素标识> 和< 元素值> 之间只有一个空格.","用作两个元素之间的分隔符, 两个元素的字符串表示之间不允许空格.

如果 attributeValue 是 CHOICE 类型, 则字符串表示为如下形式"< 识别元素标识> :< 识别元素值> ", 其中< 识别元素标识> 是识别元素标识,< 识别元素值> 是识别元素值的字符串表示.

否则, 值为编码为 BER(Base Encoded Rule, 基本编码规则) 码的 ASN. 1 值

字串形式的复合名的名字部件之间的分隔符为"/". 如果名字部件间的分隔符或名字的标识部分被用作名字的值, 则用"\"作转义符.

4.3　用 CORBA 名字服务建立基于 DN 的 MIT

在 CORBA 名字服务下建立基于 DN 的 MIT 的基本方法是使每个节点支持名字上下文接口. 有两种建立 MIT 的方法: ①用超类对象作为属类对象的名字上下文. 在这种方法中, 所有的 MO 都作为 MIT 的一部分, 名字解析过程是对象行为的一部分; ②用单独的名字服务器来注册被管对象, 其中用名字上下文接口单独建立 MIT, 而 MO 不作为 MIT 的一部分. 在这种方法中, 名字解析不作为对象行为的一部分.

4.3.1　超类对象作为属类对象的名字上下文

因为被管对象总是维护一个超/属关系, 属类对象的名字是从超类对象推演而来. 所有被管对象都是其属类对象的名字上下文. 如果存在名字绑定模板将被管对象的类名作为超类, 则被管对象可以用作其属类的名字上下文.

这种方法中, MO 需要支持名字上下文接口的操作, 结果 MO 参与了名字解析过程. 图 3 描述这种情况. 这种方法对程序员十分方便. 给定一个 MO 的引用, 我们可以通过一次强制转换而得到名字上下文接口的引用. 然后利用名字上下文接口中的 resolve() 方法根据属类对象的名字解析得到属类对象的对象引用, 然后将得到的对象引用转换为正确的类型, 最后得到该属类对象的对象引用.

但是如果 MIT 树很深, 并跨越多个域和多个服务器, 则这种方法存在潜在的性能问题. 因为所有的 MO 都继承名字上下文接口, 名字解析过程是 MO 行为的一部分, 结果, 在名字解析过程中, 沿 DN 路径上的所有 MO 对象都要被激活, 因为并不是对 DN 路径上的所有 MO 对象感兴趣, 所以这种情况有时是不希望发生的. 例如, 为了解析被管对象 *M*31, 我们必须首先在域 *D*0 中得到 Root 的名字上下文, 然后用 resolve() 操作得到域 *D*1 中的名字上下文 *M*3, 然后在 *M*3 中得到 *M*31 的接口引用. 见图 3.

4.3.2　独立的被管对象名字服务器

如果将名字上下文接口从被管对象接口的继承结构中删去, 而用名字上下文接口建立一个独立的 MIT, 就可以避免在名字解析过程中跨越多个域. 基于 DN 的 MIT 用名字上下文来建立. 被管对象自身没有结构化的层次关系. MIT 中基于 DN 的层次关系是在一个独立的服务器中加以维护. MIT 的每个节点支持名字上下文接口, 它用 MO 的 RDN 来命名, 将 MO 的接口引用作为属性维护. 当 MO 对象创建时, 在 MIT 树上一个相应的节点也被创建, 并将名字上下文对象引用绑定到相应的超类对象的名字上下文. 因为被管对象的引用并不用于绑定名字, 被管对象引用是作为一个特殊的属性加以维护的, 名字叫"moRef". 之后, 在名字解析过程中, 可以用返回的名字上下文接口的引用来获取相关的 MO 对象的引用. 见图 4.

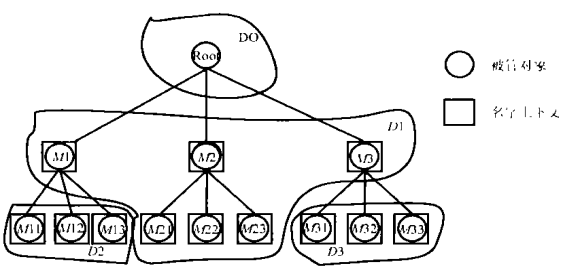

图 3　超类被管对象作为属类被管对象名字上下文

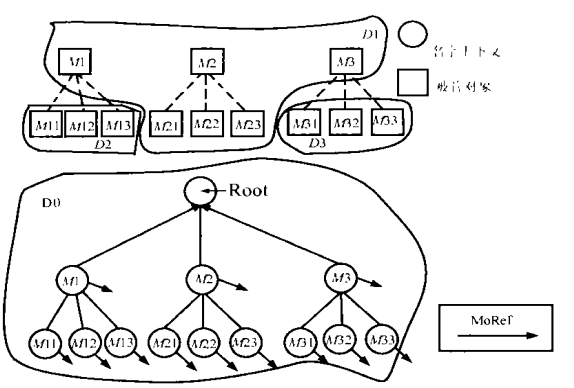

图 4　独立名字服务器

这种方法的好处是名字树可以位于逻辑上独立的名字服务器进程中. 对名字树的浏览不会迫使我们激活 DN 路径上的所有被管对象. 为了解析 *M*31 的 DN 来获得 *M*31 对象的对象引用, 首先从域 *D*0 的名字服务器中取得 Root 的名字上下文的对象引用. 然后从 Root 出发, 在域 *D*0 内逐步解析将 *M*31

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

的DN转换为一个名字上下文,然后用“moref”属性获得与*M*31的名字上下文相对应的对象引用.注意,在这种情况下,被管对象不需要象名字一样组织成一个树型结构.

这种方法与第一种方法相比没有性能问题,它将MIT封装在一个独立的服务器中.这种方法的缺点是用resolve()操作后为了得到实际MO的对象引用,必须调用一个额外的操作getmoref().

5 结论

企业计算的发展趋势是以统一的方式管理所有资源.由于当前的网络管理协议SNMP,CMIP等都不支持目录访问,因而网络管理信息资源成了企业信息资源中的一块信息孤岛.为了以统一方式更安全更方便的实现对管理信息资源的访问,必须实现MIB与DIT的集成.本文就是在MIB与DIT的集成技术方面进行了有益的尝试,实践证明上述方法是可行的.

参考文献:

[1] X/Open. Inter-Domain Management Specifications: Specification Translation [S]. X/Open Preliminary Specification, Draft, August 9, 1995.

[2] ISO. Information technology - Open System Interconnection - Definition of Management Information [S]. ISO, ISO/IEC 10165-2, 1991.

[3] OMG. The Common Object Request Broker: Architecture and Specification [S]. Object Management Group, Revision 2.0, July 1995.

[4] Network and Service Management Research Department Bell Laboratories. Mapping of common management information services to CORBA object service specification [R]. Bell Laboratories, Oct 1997.

[5] Johannsen T, Mansfield G, Kosters M, Sataluri S. Representing IP information in the X.500 Directory [M]. IETF RFC 1698 (Experimental), March 1994.

[6] 周彩章,吴宇红,常义林.TMN-MIB编译器的实现研究[J].电子学报,2000,28(4):24-27.

[7] 李木金,王光兴.基于Browser/Server模式的网络管理模型[J].通信学报,1999,20(3).

作者简介:

王 宇 男.在读博士,1974年出生于河北临漳,1996年毕业于西安交大计算机科学与工程系,取得学士学位,1999年在该专业取得硕士学位,现为西安交大计算机系统结构与网络研究所在读博士.主要研究方向为网络管理,分布式技术.

李增智 男.教授,博士生导师,1962年毕业于西安交大计算机专业,主要研究方向:计算机网络及应用,分布式系统,电子数据交换EDI.国家863项目“基于三层体系结构的企业CIMS综合智能网管技术研究”第一负责人.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net