一个CORBA-CMIP网关的设计与实现

范春晓　于中强　刘杰　(北京邮电大学电子工程学院　100876)

摘要：本文设计并实现了一个CORBA-CMIP网关，使开发人员摆脱CMIP复杂编程，解决了传统电信网管向标准TMN网络管理的平滑过度。

关键词：电信管理网(TMN)　CORBA　CMIP　PDU　GDMO　IDL

电信网管，尤其是国内的电信网管一直面临着一个多管理系统之间的互操作问题以及多厂商设备环境下的网管问题，尽管TMN提供了一系列支持网管系统互操作的标准接口，但由于其协议较复杂，并且没有支持协议的相应开发工具，因此国内的许多网管系统还是传统意义上的专业网管系统。要想短时间内将其转化为TMN标准模式是不太现实的，近几年出现的CORBA技术，具有分布式处理，异构通信能力，并带有强大的可操作工具，使我们有了一种新的网管互操作问题的解决方案，可以满足传统网管向TMN系统平滑过度的需求。

1　CMIP与CORBA

TMN定义的支持互连、互通、互操作的标准接口称为Q3接口，Q3接口是一个非对称接口，接口两端的实体分别为Manager和Agent，Q3接口由三部分组成：通信协议栈，网络管理协议和管理信息模型，其中网络管理协议有CMIP和FTAM。

管理信息协议（Common Management Information Protocol：CMIP）是Q3接口的应用层中很重要的协议，主要用来进行网络管理操作等数据量较小的网络管理服务，通过该协议实现管理者与代理的互联和互操作。CMIP在用于面向事物处理的管理应用中起到了很大的作用，通过它使各个管理者提交的事物，由代理来统筹安排，进行分布处理及计算，它与FTAM协议结合，很好的达到TMN要达到的分布计算目标。但是，CMIP管理系统对多个专业管理系统却有些不太适宜，它更象一个点对点的"对象通道"，并且CMIP自90年代初无太大进展，对于许多新技术没有更多的支持。

最新的基于TCP分布式处理方法是CORBA（Common Objects Request Broker Architecture），这是对象管理组OMG（跨工业行业的联合组织）定义的分布对象计算的技术，是面向对象的分布式系统建立所依据的标准。CORBA允许跨网络分布的应用间作面向对象通信，当然，TMN标准中的分布机制OSI CMIP也可以做到，但与CORBA不同的是，不能跨工业行业使用，所以不能作为主流计算技术，并且CORBA出现后新的编程语言都支持CORBA开发，如Java及有Web显示特点的语言，这些都使CORBA比CMIP更适于分布计算，表1总结了在分布计算和管理中CORBA与CMIP差别。

表1　CORBA与CMIP比较

	CORBA	CMIP
目的	是分布式面向对象应用通信的“总线”	是管理应用之间的点对点面向对象通信的“对象管道”
标准体	OMG。 CORBA也是OSI的对象分布处理（ODP）标准的基础。	80年代末90年代初在ISO。 ISO现在也移到ODP标准上了，是CORBA的外延。
标准API	OMG定义的(Java,C, C++,Smalltalk,COBOL)	NM论坛定义的（C++）
支持Web	Java支持即Applet支持，Netscape浏览器内嵌ORB	不支持。 Web技术之前出现的。
实时支持	ORB供应商提供可嵌入的CORBA核心，并嵌入未来的Java虚机	CMIP供应商有其代理和栈的实时操作系统接口
领域	与领域无关	只限于电信管理
TMN信息模型	需要用NM论坛的JIDM标准类似途径翻译成IDL	用CMIP数据定义语言GDMO/ASN.1

2 CORBA-CMIP网关设计

如果能将CORBA应用到TMN中，即用CORBA来代替Q3(CMIP)，那么TMN的开发近乎于纯CORBA开发，而支持CORBA开发和应用的工具有很多，开发起来就容易多了。

由表1的比较可以看到：CORBA没有电信接口的标准模型，而Q3的标准化就比较好，使用它的经验也比较多(主要指的是在网元管理层)，所以就目前的情况来看，在网元管理层，在对具体的网络设备进行管理时，很多时候还得用CMIP，这是一个比较棘手的不得不面对的问题。

解决这一问题可以考虑在较高层次的系统中使用CORBA技术，而在直接对网元进行管理时仍使用CMIP协议，也就是说两者会共存，办法就是使用CORBA-CMIP的网关，这样CORBA管理者能操作原有的CMIP系统，而且具有透明性：即CORBA管理者不知道它操作的是CMIP系统；CMIP代理者也不知道那是CORBA管理者，这样便于系统更新和兼容，使用CORBA-CMIP的网关的网管系统示意见图1。

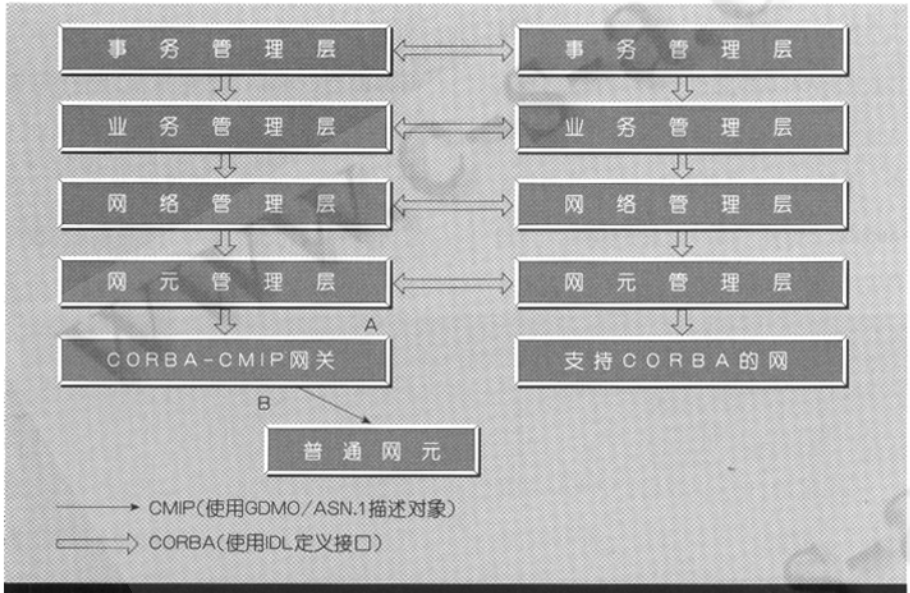


图1 新结构网管系统

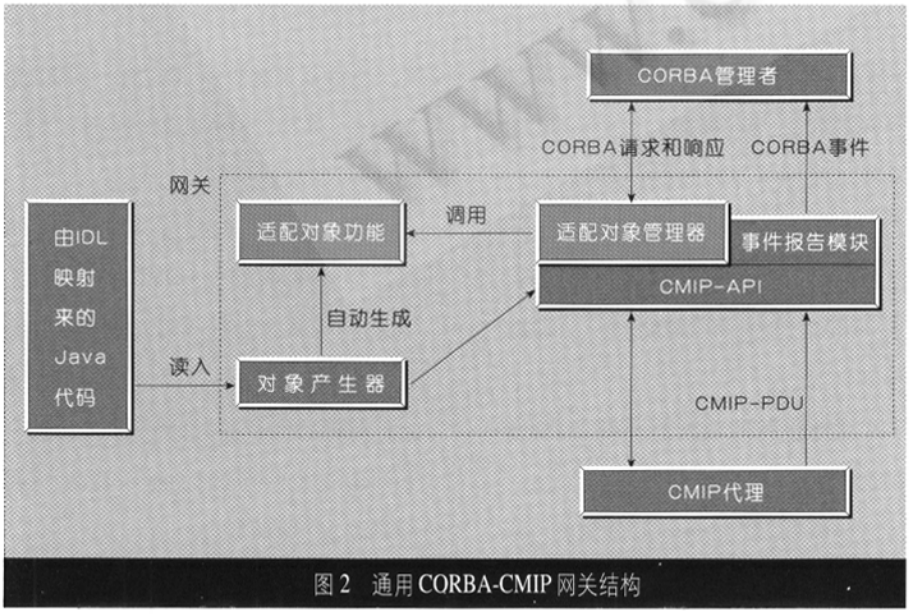


图2 通用CORBA-CMIP网关结构

CORBA-CMIP协议网关是协调基于OSI的TMN代理和基于CORBA的管理程序之间的通信。网关运行时，它的输入是由CORBA管理者发来的CORBA的请求，输出是将要发送给CMIP代理者CMIP-PDU形式的CMIP请求。而在创建网关时(主要是得到适配对象的代码)，需要读入由IDL定义映射来的Java代码(该IDL定义是由相应的GDMO-CORBA的翻译器生成的)，因为通用网关需要适应各种GDMO/ASN.1描述，所以需要对象生成器生成不同的代码，重新编译"适配对象"部分。

根据参考文献[1]中的"TMN与CORBA集成的结构"，作者具体化了一个通用网关，结构如图2所示。

其中各模块功能如下：

(1)"CMIP-API"：完成CMIP-PDU的编码和解码，发送和接收，提供给事件报告对象和适配对象管理器调用；

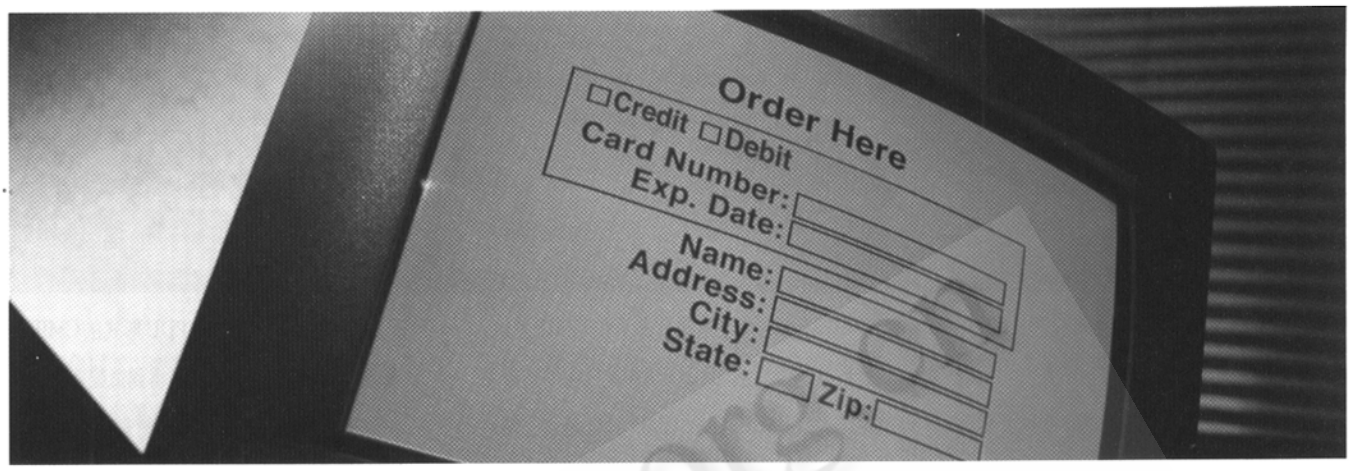
(2)"事件报告模块"：从CMIP代理中接收事件，并发送到CORBA管理者；

(3)"对象生成器"是这个网关的核心模块，它根据CORBA到CMIP的信息模型的映射规则来生成"适配对象功能"模块和"适配对象管理器"模块，它的输出是"适配对象功能"模块和"适配对象管理器"模块的Java源代码；

(4)"适配对象功能"和"适配对象管理器"：由"对象生成器"生成，当IDL定义改变后(通常是因为相应的GDMO/ASN.1描述变化的结果)，这两个模块需要重新编译，创建适配对象。

3 CORBA-CMIP网关实现

针对网关中的几个主要模块功能，分析



JAVA中CORBA开发工具特点，采用相关技术，实现了这个网关。

3.1 利用CORBA的事件服务或回调技术实现“事件报告”

“事件报告”模块的输入是由CMIP代理发送来的m-event-report的PDU(可以调“CMIP-API”来解释)，输出是相应CORBA事件报告。

之所以设计这个模块，其根源是CMIP的管理者/代理者结构是对等的，而CORBA的服务器/客户端结构是不对称的。在CMIP中，代理者也可以发出请求，所以它可以发出M-EVENT-REPORT原语来报告事件，而CORBA的服务器不能主动发出请求(至少在CORBA2.0中还是如此)。有两种解决办法：

一是让“事件报告模块”本身又是一个CORBA的服务器/客户端结构，只不过它同发送M-SET、M-GET等原语的“适配对象管理器”相反，它的客户端在靠近CMIP代理的一方，由这个客户端来向CORBA管理者报告事件。这种方法存在重用性不高、接口不规范、性能不好等缺点，一般不用。

二是使用CORBA的事件服务(Event Service)，这是常用的方法。下面就是根据CORBA事件服务的特点来讨论使用什么样的事件报告模式(是Push还是Pull)。

CORBA的事件服务使用生产者(Supplier)/消费者(Consumer)/通道(Channel)的概念，事件数据(Event Data)由生产者对象产生，放入通道中，再由通道传给消费者对象。在通道中，为每一个生产者建立一个生产者代理(Supplier Proxy)，为每一个消费者建立一个消费者代理(Consumer Proxy)。这样才能实现事件的自动报告。有两种传送事件数据的模式：推模式(Push Model)和拉模式(Pull Model)，其中Push模式是较常用的模式。

在我们这个网关中，使用了CORBA的事件服务，这里生产者对应的是“事件报告模块”，它能根据CMIP代理的通知产生CORBA事件报告。而消费者是CORBA管理，它接受事件报告。由生产者代理把事件提交给消费者代理，而消费者代理则需要不断的在查询是否有数据到来。而拉模式中，消费者代理把生产者代理的数据“拉”出来放入队列，等待消费者处理，而生产者代理则需要不断的查询是否有来自消费者代理的请求，从而完成了“CORBA管理者”和“CMIP代理”之间的连接工作。

3.2 采用粗细结合法完成GDMO到CORBA的映射

“对象生成器”负责创建适配对象，实际是根据CORBA到CMIP的信息模型的映射规则来生成“适配对象功能”模块和“适配对象管理器”模块，这个映射的关键是确定GDMO到CORBA的映射方法。目前比较流行的建模原则是重用那些基于GDMO/ASN.1描述的信息模型，将其转化为对应的CORBA信息模型，具体有细粒度法、粗粒度法和粗细结合法。

细粒度法的基本思想是用一个CORBA实例来管理一个相对应的被管对象实例。此时，CORBA对象(IDL接口)中的方法(成员函数)和GDMO对象的通知(Notification)，动作(Action)是一一对应的，CORBA对象的属性和GDMO对象的属性也是一一对应的。这种方法的优点是分布性能好(可以把每一个对象分布到不同的地方)而且对对象的操作十分灵活(实例间是一一对应的)。其缺点是分布有太多的CORBA对象，占用资源太多，影响运行效率。

粗粒度法的基本思想是用一个CORBA对象实例来管理所有的GDMO被管对象类：这种粗粒度法的优点是具有较好的运行效率(因为只有一个CORBA对象，占用系统资源少)。而缺点是基本不具有分布特性(只有一个CORBA对象，无法分布处理)，违背了采用CORBA的初衷。

粗细结合方法的基本思想是用一个

CORBA对象实例来管理一个GDMO类(实际是管理这个类的生成的所有实例)，所以在定义CORBA对象类时除了要包括原来GDMO类的通知动作和属性外还要增加createInstance()和deleteInstance()这两个方法用来创建和删除对象实例。

本网关采用了粗细结合方法，表2显示一个扬声器设备的GDMO描述和IDL描述。

3.3 CMIP-API模块和适配对象模块

"CMIP-API"模块完成CMIP-PDU的编码和解码，发送和接收。提供给事件报告对象和适配对象管理器调用。该模块的输入是事件报告对象和适配对象管理器对它的调用，输出是CMIP的PDU数据，或者输入PDU数据，由它触发相应的函数(这可以用Java的Listener接口来实现)。

(1) CMIPConnection类：用于保持和代理者的连接，发送和接收PDU；

(2) PDUListener接口：监听代理者发来的PDU；

(3) transform类：实现ASN.1的基本编码的功能以及PDU编码解码时所需的辅助功能；

"适配对象模块"由"对象生成器"生成，它既是一个CORBA代理者(服务端)，又是一个CMIP管理者。它是IDL中定义的接口的实现者，其功能就是收到相应的CORBA调用后，让CMIP代理者进行对应的CMIP操作(其对应规则可以是专用的，自定义的，标准的)。在没有"对象生成器"的专用CORBA-CMIP网关中，该模块是网关的核心，由人工针对某种特定的系统(如无线接入网管系统)，即特定的接口定义开发。

4 结束

该程序已用Java写成，分成六个包AdapterObj、agentDemo、CMIPOp、GateTools、managerDemo、dev，在任何安装了jre1.2以上版本或安装了jdk1.2以上版本(即j2sdk)环境下均可运行，并与编写的管理/代理程序顺利连接。

参考文献

1 现代网络管理技术，孟洛明等,北京邮电出版社，1999年11月

2 <<CORBA教程>>,R.Otte P.Patrick M.Roy著，李师贤译，清华大学出版社。

表2

```
--GDMO/ASN.1 描述
speaker MANAGED OBJECT CLASS
  DERIVED FROM "Rec. X.721:1992 ":top;
  CHARACTERIZED BY
     speakerPackage     PACKAGE
        BEHAVIOUR
           speakerBehaviour  BEHAVIOUR
              DEFINED AS  "该对象描述扬声器设备";;
        ATTRIBUTES
           volume     GET;;
        ACTIONS
           turnUp ;
        NOTIFICATIONS
           overMaxVol ;
        -- 省略CONDITIONAL PACKAGE
     ;;
REGISTERED AS {111 2};
-- end class speaker
```

```
//IDL 定义(粗细结合法)
interface speakerHandler    //管理 speaker 的接口
  {
     VolType get_volume(in string id);
     //nonconfirm 模式的 action
     string action_TurnUp(in string id);
     string action_TurnDown(in string id);
     //实现通知 OverMaxVol
      string notifications_OverMaxVol(in string id,in EventInfo EI);
      //下面两个是附加的，固定的
     string createInstance(in string id);
     string deleteInstance(in string id);
  };
};
```