第16卷 第1期
2006年1月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.1
Jan. 2006

基于SNMP的统一网管框架的设计与实现

张 朕,应吉康
(华东师范大学 计算中心,上海 200062)

摘 要:随着通信业务量需求的成倍增长,网络规模的日益膨胀,网元设备的种类和数量不断增加,各个网元都有自己的管理系统,这给网络的管理维护带来了很多困难。为了提高网络管理的效率,很有必要对各网元进行集中管理,实现在一个统一的平台上管理各种设备。文中将设计模式的思想融入网络管理框架设计中,实现了一个基于SNMP的统一网管平台,该网管平台提供一个高分布性、高扩展性的架构来保证应用的开发者可以用不同的方法来布置他们的网元管理系统。根据实际应用可以证明,该框架具有良好的灵活性和扩展性。

关键词:电信管理网;简单网络管理协议;网管系统
中图分类号:TP393.09 **文献标识码**:A **文章编号**:1005-3751(2006)01-0027-03

Design and Implementation of Uniform Network Management System Based on SNMP

ZHANG Zhen, YING Ji-kang
(Computer Center of East China Normal University, Shanghai 200062, China)

Abstract: With the increasing of communication portfolio demand and the expansion of network scale, there are more and more various network elements. Each network element has its own management system, which brings many difficulties for the management and maintenance of network. For the purpose of improving the efficiency of network management, it is necessary to build a uniform network management system which can centralize all network elements. In this paper, implements a uniform network management system based on SNMP. It provides a high distributed and high expansible architecture for the developers to configure their own network management system. According to the practical application, this architecture has good flexibility and expansibility.

Key words: TMN; SNMP; NMS

1 网络管理的发展与现状

伴随Internet时代的到来,网络技术的迅猛发展,越来越多的企业、政府、学校、个人等都融入互联网当中,网络已经与人们的学习、工作及生活密不可分[1],而作为整个互连网的基础,电信网稳定、高效、准确地运行就显得极为重要。要做到这一点,除了要依靠网络设备本身和网络架构的可靠性之外,还必须依靠一套有效的网络管理手段来监测和管理整个网络,而传统的单层网络管理模式已经无法适应现代网络管理的需求。为了有效合理地管理现代网络,国际电信联盟电信标准化部门(ITU-T)于1988年,参考OSI系统管理框架提出了具有标准协议、接口和体系结构的管理网络—电信管理网(Telecommunication Management Net,TMN),作为管理现代电信网的基础[2]。考虑将提供业务的电信网及其管理功能进行分离,使管理功能从电信网中独立出来单独组成一个网,即TMN。TMN制定了一系列的标准和管理功能,包括被管网元和网络管理系统之间、网络管理系统之间的接口均被标准化了。只要被管网元和网络管理系统之间遵循TMN标准,完成一定的管理功能,就能够实现不同厂商的不同设备以及不同网络管理系统之间的互连互通操作。TMN体系结构按照不同的管理需求将整个电信网管理功能从低到高分作5层:网元层(NEL)、网元管理层(EML)、网络管理层(NML)、业务管理层(SML)、事务管理层(BML)。

目前,由于实际技术水平和应用条件的限制,网络管理主要集中在网元管理层和网络管理层,真正业务级的管理还只停留在理论研究阶段。网元管理层直接管理物理网络,是整个管理系统的基础。为了适应网络管理系统可扩展性的要求,我们将设计模式的思想融入网络管理框架设计中,保证该框架具有良好的灵活性和可扩展性。

2 网络管理协议

当前最典型的网络管理协议有基于OSI七层模型的公共管理信息协议(CMIP)和基于TCP/IP的简单网络管理协议(SNMP)[3]。OSI/CMIP系统管理模型是目前理论上最完备的网络管理模型,是其他网络管理模型的基本

收稿日期:2005-04-05
作者简介:张 朕(1980—),男,重庆人,硕士研究生,研究方向为现代信息系统及其开发技术;应吉康,研究员,研究方向为现代信息系统及其开发技术。

参考。但由于该模型比较复杂,实现代价高,因此并没有得到广泛的应用。相反,当初只是为了管理 TCP/IP 网络的 SNMP 却得到了迅速的发展和广泛应用。SNMP 网络管理模型的突出特点是简单、易于实现,因而得到了厂商的支持。特别是在 Internet 上的成功应用,使得它的重要性越来越突出,已经成为事实上的工业标准。

SNMP[4]主要包括 SMI(管理信息结构)、MIB(管理信息库)和 SNMP 协议几部分。SMI 给出了管理对象定义的一般框架。MIB 是设备所维护的全部被管理对象的结构集合。SNMP 协议包括 SNMP 操作、SNMP 信息的格式以及如何在应用程序和设备间交换消息。SNMP 采用代理/管理站模型进行网络管理[5],如图 1 所示。

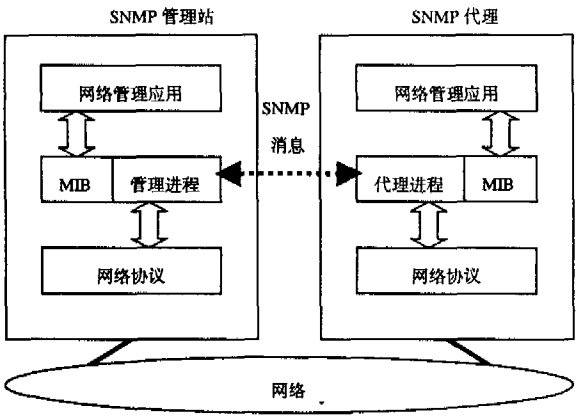

图 1 SNMP 网络管理模型

3 基于 SNMP 的统一网管框架设计

3.1 总体结构设计

统一网管平台提供一个高分布性,高扩展性的架构来保证应用的开发者可以用不同的方法来布置他们的网元管理系统。从横向分层的角度来看,平台可以分为支撑层(Support Layer)、框架层(Framework Layer)和应用层(Application Layer)。统一网管平台是 C/S 结构,包括网管服务器和网管终端两大部分,通过网管服务器的级联构成多级网管的无缝集成。其中框架层和支撑层位于服务器端,应用层位于客户端,如图 2 所示。

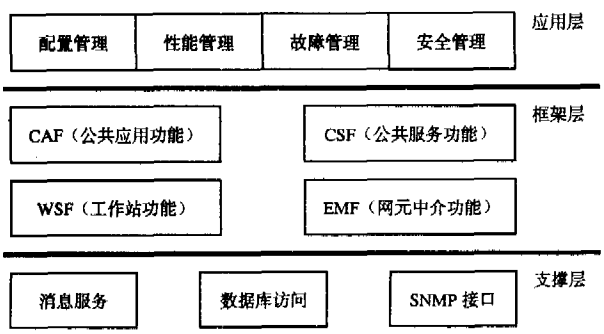

图 2 总体结构示意图

3.2 支撑层的设计

支撑层作为系统支撑平台为应用提供系统级的服务,包括消息服务、数据库访问和 SNMP 接口。支撑层是整个统一网管平台的基础。

3.2.1 消息服务

消息服务为系统中所有模块的交互提供支持,实现消息的路由转发,根据消息类型,转发到相应的处理模块。消息的格式如下所示:

版本号	消息头	消息数据

消息头的定义如下:

```
typedef struct msg {
  void * mlink; //指向结构的指针,形成消息链表
  PID sender_pid; //发送进程标识
  PID receiver_pid; //接收进程标识
  BYTE msg_type; //消息类型(2 或 3,表示同步或异步进程消息)
  WORD event_type; //事件类型
  WORD msg_length; //消息长度,不包括头
}MSG_HEAD;
```

系统使用的是非阻塞的异步消息机制,系统维护一个消息队列,用来存放系统各个模块产生的消息。程序中有一段程序代码,叫做消息循环,用来从队列中取出消息,并且将它们发送给相应的消息处理程序。代码片断如下:

```
while (GetNmMessage (&msg, NULL, 0, 0))
{
    TranslateNmMessage (&msg);
    DispatchNmMessage (&msg);
}
```

3.2.2 数据库访问

数据库访问模块提供所有访问数据库的接口调用,并维护数据库连接池,其他模块只需向其申请连接即可。

3.2.3 SNMP 接口

网管系统与网元的交互是基于 SNMP 协议的,在 SNMP 协议栈的实现中,采用了被广泛使用的 CMU(Carnegie Mellon University)开发的开放源代码的 CMU-SNMP 软件包,该软件包对 SNMPv2c 做了很好的实现。但是对于上层的开发人员来说,调用起来不是很方便。例如,一次 SNMP 操作,就要先后调用以下这些函数:

```
char * winsock_startup(void);
void snmp_synch_setup(struct snmp_session * Session);
struct snmp_session * snmp_open(struct snmp_session * session);
struct snmp_pdu * snmp_pdu_create(int command);
int snmp_synch_response(struct snmp_session * Session,struct snmp_pdu * PDU,struct snmp_pdu * * ResponsePDUP);
int snmp_close(struct snmp_session * session);
```

为了方便业务开发人员,对该协议栈进行了简单封装。将 CMU 格式的 SNMP 消息结构转换为该网管平台内部使用的数据结构,并提供一系列便于使用的 SNMP 原子命令接口和其它辅助功能接口,为其上层应用和 Trap 监听进程提供统一的支持。

通常,通过 SNMP 协议向 SNMP 代理发送请求,是通过使用 MIB 中定义的对象标识符(OID)来标识的。OID 是用句点隔开的一组整数,使用起来很不方便,可以通过

MIB 树中叶子的名称来访问和设置 MIB 树中的数据,而不是通过枯燥的数字,方便开发人员进行开发。

以 RFC1213 定义的 MIB－Ⅱ中的.iso.org.dod.internet.mgmt.mib－2.system.sysName 为例,其定义如下:

```
sysName OBJECT－TYPE
SYNTAX DisplayString (SIZE (0..255))
MAX－ACCESS read－write
STATUS current
DESCRIPTION
    "An administratively－assigned name for this managed node. By convention, this is the node's fully－qualified domain name. If the name is unknown, the value is the zero－length string."
::={system 5}
```

如果要查询系统的名称,在发送 GetRequest 命令时需要使用 1.3.6.1.2.1.1.5.0 去向 SNMP 代理检索它,这一长串的 OID 字符串在程序中的可读性很差。文中采取的措施是建立一个 Hash 表,在 SNMP 模块初始化的时候先对程序所使用的 MIB 进行分析,对 sysName 这个字符串进行 Hash 操作,将 sysName,OID 和 DisplayString 保存在 Hash 表中。在向上层应用提供的 SNMP API 中只需要对 sysName.0 进行操作,就能查询到 sysName 的值,而不是通过枯燥的数字,方便开发人员进行开发。

MakeHash 函数如下所示:

```
ULONG MakeHash(char * str, ULONG len)
{
  ULONG n;
  n=0;
#define HASHC n= *str++ +65587*n
  if (len>0)
  {
    int loop;
    loop=(len+8－1)>>3;
    switch (len & (8－1))
    {
      case 0:
        do
        {
        HASHC;
      case 7: HASHC;
      case 6: HASHC;
      case 5: HASHC;
      case 4: HASHC;
      case 3: HASHC;
      case 2: HASHC;
      case 1: HASHC;
      } while (--loop);
    }
  }
  return n;
}
```

另外,由于使用的 CMU 的 SNMP 协议栈是非线程安全的,而上层应用是基于多线程的,因此,在对其进行封装的过程中一个重要的任务就是使其线程安全,目前采取的方法是采用事件机制,每个 SNMP 原子命令执行前,都调用系统的同步机制函数等待一个事件对象 mm_hSnmpEvent 的触发。

3.3 框架层和应用层

针对网管系统这个特定的应用领域,抽取出一些公共的软件框架,这些程序框架统称为框架层。所有具体的管理应用都应该基于这个框架层提供的全部或者部分框架来开发。具体应用中,通过框架导出的 API 进行二次开发。如图 1 所示,框架层分为公共服务功能、公共应用功能、工作站功能和网元中介功能 4 个部分。公共服务功能提供网管系统所公有的,和具体网元无关的服务功能,如安全、日志等;和网元相关的管理功能,如告警、性能、配置等,抽取其中的公共部分称为公共应用功能;工作站功能实现客户端 GUI 的展现功能,并提供图形控件;网元中介功能实现网管系统和网元之间的接口转换和适配功能。

应用层就是在平台框架层的基础上,提供具体的管理功能,如配置管理、性能管理、故障管理、安全管理等。这些功能都是可以拆卸的,应用在使用统一网络管理平台时,可根据自己的具体需要来选择需要装载哪些管理功能。

4 结束语

文中介绍了针对网元层的基于 SNMP 的网络管理框架,该框架是在 Windows 平台上利用 Visual C++ 开发出来的,实现了在一个统一平台上对各种网元进行管理,具有良好的灵活性和可扩展性,通过对各网元进行集中管理,极大地提高了网络管理的效率。但是该系统依赖于 Windows 平台,在移植性上尚存诸多不足。目前,Java 平台在移植性上做得比较成功,但是对于电信级的网管平台来说,其性能并不能满足要求,与此同时,CORBA 应用日益成熟,C++ 领域高性能的分布式网络框架 ACE 的异军突起,为人们提供了跨平台的最好选择。可以用 Java 开发客户端 GUI;用 ACE 框架来实现服务器端的网管功能;用 CORBA 实现客户端与服务器端的通信,从而真正实现高性能、可移植的网络管理框架。

参考文献:

[1] Subramanian M. 网络管理－原理与实践(影印版)[M]. 北京:高等教育出版社,2001.

[2] 郭　军. 网络管理[M]. 北京:北京邮电大学出版社,2001.

[3] 岑贤道,安常青. 网络管理协议及应用开发[M]. 北京:清华大学出版社,1998.

[4] RFC1157. A Simple Network Management Protocol[S]. 1990.

[5] Mauro D, Schmidt K. Essential SNMP[M]. Sebastopol: O'Reilly & Associates, 2001.