第 34 卷 第 3 期 计 算 机 工 程 2008 年 2 月
Vol.34 No.3 Computer Engineering February 2008
·网络与通信· 文章编号：1000—3428(2008)03—0136—03 文献标识码：A 中图分类号：TP393

基于内容类型的网络信息管理体系架构及实现

夏晓忠[1,2]，肖宗水[1]，方长江[1]
(1. 山东大学计算机科学与技术学院，济南 250061；2. 中国人民解放军 72433 部队，济南 250014)

摘 要：从内容管理的思想出发，将各种网络信息抽象为不同的内容类型，基于开源的内容管理开发环境构建了一个网络信息管理平台的体系架构。该架构的实例采用 Python 作为主要的开发语言，完成系统的逻辑功能，结合 AJAX, DTML, ZPT 等页面标记语言实现了系统的表现层。试验表明该平台具有经济实用、稳定高效、开发便捷的优点。

关键词：内容类型；网络管理；Zope 系统

Architecture and Implementation of Network Information Management System Based on Content Type

XIA Xiao-zhong[1,2], XIAO Zong-shui[1], FANG Chang-jiang[1]
(1. School of Computer Science and Technology, Shandong University, Jinan 250061;
2. Unit 72433 of the Chinese People's Liberation Army, Jinan 250014)

【Abstract】From the thinking in content management system, this paper abstracts different content types from various kinds of network information and designs architecture of the network information management platform based on the open source content management development environment. This architecture is programmed with Python as primary language to achieve logistic function, and integrates AJAX with page marked language, such as DTML and ZPT to realize exhibition layer. Experimental results show that the platform is economical, practical, stable and efficient. It is convenient to development.

【Key words】 content type; network management; Zope

面对网络规模持续扩大、网元设备日趋多元化的复杂网络环境，政府机关、科研机构和中小企业的网管人员急需一套安全可靠、使用方便、兼容性强的企业级网络管理工具。目前比较有影响的网络管理套件主要有 Open-View, NetView, SunNet 以及代表未来智能网络管理方向的 SPECTRUM。虽然这些优秀的网络管理套件功能全面、性能优良，但都不可避免地存在着使用复杂、扩展困难、费用高昂等不足，并不适合中小规模、投资有限的单位使用。本文从经济效益、使用维护、拓展开发等角度，提出并初步实现了一种基于内容类型的网络信息管理平台(Network Information Management Platform, NIMP)架构。

1 相关技术

1.1 基于 Web 的网络管理

瘦客户端的网络管理系统有独立的平台，易于控制和使用，并且在地理上和系统上具有可移动性，较 C/S 模式更加灵活方便，使得基于 Web 的网络管理模式成为当代网管系统设计的一种流行趋势。NIMP 使用基于 Web 的网络管理架构在一个管理工作站上运行 Web 服务器，通过 SNMP 协议与被管对象通信，管理员可在任意具有浏览器的客户机上通过 HTTP 协议与 Web 服务器通信，实现对整个网络的工作状态进行监控。

1.2 内容类型

内容是一个比数据、文档和信息更广的概念，是对各种结构化数据、非结构化文档、信息的聚合，它包括一些附加信息的数据单元，有公共类似的属性，可以被具有某种角色的用户添加和编辑，并通过各种方式发布[1]。全部属性相同的数据单元成为一种内容类型(content type)。网络管理的直接对象是各种网元设备，如路由器、交换机等，以及由其MIB信息库中反映出来的各种网络运行状态，这些信息都可依据地址、端口、服务、流量等要素，从面向对象的思想，抽象为不同的内容类型，如路由器、交换机等设备类型和图表、日志、报警等信息类型。通过对各种内容类型进行相应操作，如设备类型的添加、删除、取值，信息类型的绘制、日志记录、报警产生等，形成基于内容类型的网络信息管理系统，达到对网络设备、运行状态进行有效监控的目的。

Zope是一套使用Python开发的开源内容管理系统，集成了Web的网络服务器产品，并可与IIS或Apache等流行Web服务器集成[2]，实现了表现层、逻辑层和数据层的分离，分别提供了脚本工具：DHTML用于编写网页模板，Python用于开发后台组件，方便不同的人员进行不同的网站编写工作。它拥有一个面向对象的内容类型数据库ZODB，同时支持符合ODBC标准的关系型数据库。完全支持SQL, ODBC, XML, HTTP, XML-RPC, SOAP等开放标准。NIMP是构建在Zope之上的管理系统，可以配置多台Zope Server分担网络请求，提供分布式的服务。

1.3 XML-RPC

XML 远程方法调用使用 HTTP 作为传输协议，采用 XML 词汇表作为消息有效负载进行远程过程调用的简单规范。大多数语言都有了标准的或已经可用的 XML-RPC 实现，Python

作者简介：夏晓忠(1975－)，男，硕士研究生，主研方向：计算机网络综合管理；肖宗水，副教授；方长江，硕士研究生

收稿日期：2007-06-22 E-mail：xiaxiaozhong@sdu.edu.cn

语言中已经捆绑了 xmlrpclib(一种 XML-RPC 的客户库实现)。

NIMP 架构基于内容类型的思想，以 Zope 为应用服务器，采用 Python 作为开发语言，使用 XML-RPC 远程调用 Zope 脚本，实现系统各部件之间的通信。

2 体系架构

NIMP 体系架构分为对象层、服务层、数据层、逻辑层和表示层，各层关系如图 1 所示。

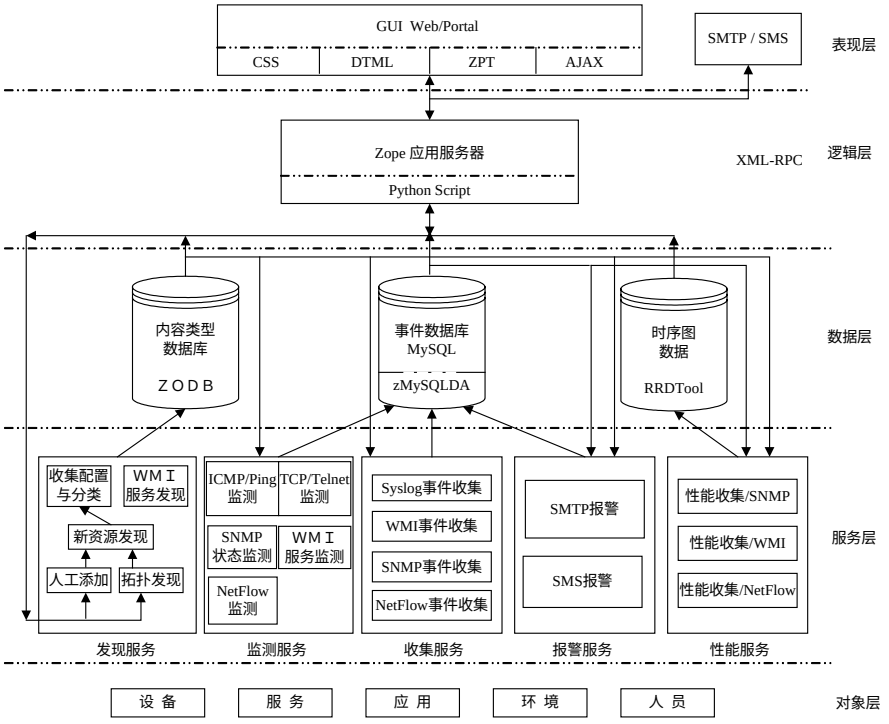

图 1 NIMP 系统体系架构

2.1 服务层

服务层采用插件形式，分为发现、监测、收集、报警和性能 5 类服务，各类服务又细分为多个独立的进程。多数进程之间通过数据层进行通信，避免了进程之间的耦合，便于功能的删减、部署和插件的开发。

发现服务组是系统的入口模块，管理员既可通过指定设备地址和有关参数手工添加设备，也可向拓扑发现服务提供种子地址和有关参数进行自动发现。网络的 3 层拓扑发现比较容易实现，2 层拓扑可以采用有关SNMP和ICMP结合的发现算法[3]得到主要的网络设备。然后经过配置收集与分类服务进程依类型分类处理后存入内容类型数据库，供其他服务进程和逻辑层使用。

检测服务组由 ICMP 监测、TCP 监测、SNMP 监测、WMI 监测和 NetFlow 监测等进程组成，各进程独立运行，从内容类型数据库中读取待监测对象的地址和参数，使用相应的协议完成监测任务，并将监测的结果通过 zMySQLDA 数据库连接对象存储在 MySQL 数据库中，供报警进程和逻辑层处理。

事件收集服务组由 Syslog 事件、WMI 事件、SNMP 事件和 NetFlow 事件 4 个事件收集服务进程组成，独立运行，事件收集的对象来自内容类型数据库，结果存储在 MySQL 数据库中。其中，Syslog 事件收集服务负责收集和分类系统事件，提供一个面向系统模型的事务信息，如记录管理平台各部件的运行情况、对资源的请求等；WMI 事件收集服务配合 Windows 主机中的客户端 WMI 来获取系统信息，WMI 是一种桌面管理的规范和基础结构，通过它可以访问、配置、管理和监视 Windows 资源；SNMP 事件收集服务根据内容类型数据库中的对象、参数和目标 OID，使用 SNMP 协议收集网络设备中的 SNMP 代理信息，如设备全局信息、接口数量、端口信息等；NetFlow 事件针对思科设备的流信息收集进程。

报警服务提供 SMTP 和 SMS 报警。报警服务进程读取内容数据库中管理员对该内容类型所设定的事件规则，将其应用于 MySQL 数据库中由监测服务进程和事件收集服务进程所产生的信息，生成报警信息，存储于 MySQL 数据库中，供 Zope 应用服务器发送该报警。

性能收集服务有 SNMP 协议收集、WMI 协议收集和 NetFlow 协议收集等进程，各进程从内容类型数据库中获取配置参数，依据 MySQL 数据库中的事件收集信息，经 DDRTool 组件生成设备各项时序数据，存储为性能时序图。

2.2 数据层

数据层是服务层与逻辑层之间的桥梁，保存着系统中各信息资源的参数和历史信息，是服务进程工作的对象和加工的数据源，也是磁盘读写非常频繁的部分。在数据库的选取上既要考虑存储对象的自身特点又要考虑存取的效率，因此，NIMP 采用了内容数据库和关系型数据库 2 类数据库，还采用了 DDRTool 时序图文件作为性能数据的存储形式。

内容数据库是 Zope 内置的支持事务机制的高性能对象数据库，用于存储发现服务进程所产生的内容对象及配置信息。NIMP 将各种网络信息资源定义为不同的内容类型，用内容类型数据库存储使得网络信息资源的存储、审核、发布、管理等事务流程更安全、更快捷。

NIMP 将监测、收集和报警服务进程输出的数据保存在 MySQL 数据库中，既实现了数据的高效访问，又便于分布式部署，起到合理配置资源的作用。

本系统将流量数据利用RRD数据库格式储存，该数据库采用十分紧凑的方式存放数据，能储存任何类型的数据，并采用循环的方式使用数据库，确保数据库的大小不会无限制地增长，减少了额外的维护。RRDtool采用简单的语句格式来实现数据库的创建、存取和更新，使系统的实现具有很高的效率[4]。

2.3 逻辑层

NIMP 体系架构使用 Zope 应用服务器作为逻辑层开发框架。该框架带有可扩展的内置对象和强大的安全集成模块，使用 Python 语言编写，在性能敏感的部件上采用了 C 语言，并提供了一套进行管理和开发的完整的 Web 环境，用来控制 Zope 以及各种对象，或者进行开发 Web 应用程序。

逻辑层通过数据库连接对象建立和管理对外部数据库的连接。所有的 ZSQL 方法都必须和一个数据库连接对象关联，通过数据库连接对象执行 SQL 代码。NIMP 使用 ZMysqlDA 数据库接口与 MySQL 数据库进行会话，执行与关系数据库

的数据交换。

Zope 完全支持使用 Python Script 和 External Method 进行开发，NIMP 在 Zope 上对复杂的逻辑处理使用的是 Python 的子集——Script 脚本，该 Script 主要对 Python 中涉及安全的语句进行了裁减，系统在不可避免地与安全有关的逻辑上可以使用 External Method 进行编写。NIMP 使用 Zope 提供的 2 种以模板方式来处理文本、XML 和 HTML 这样的表示层数据：(1)文本模板标记语言(DTML)；(2)Zope 页面模板(ZPT)。

2.4 表现层

NIMP 体系架构的表现层有一个 Web 界面，用于完成各项网络管理任务。界面采用 Portal 的思想，划分成 4 个 Portlet，界面布局及内容如图 2 所示。根据各级用户角色分配权限的不同，登录后各 Portlet 展示的内容有所不同，结合 Zope 的认证和权限分配机制，实现了安全的用户管理。

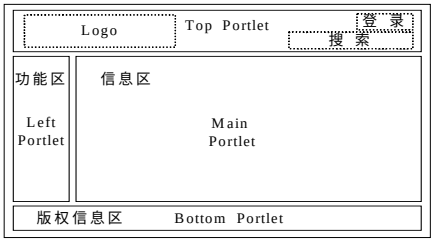

图 2 界面布局及内容

使用层叠样式表单CSS、DTML标记语言和Zope页面模板来自定义系统的皮肤和内容布局，保持了界面的风格一致。使用AJAX[5]向Zope应用服务器发出异步请求，以执行更新或查询数据库。当请求返回时，使用JavaScript和CSS来相应地更新用户GUI，避免刷新整个页面，将使用者从请求/响应的循环中解脱出来。

另外 NIMP 对系统和各资源发生的事件根据用户自定义的规则进行评估，当评估值超过用户设定的阈值，并在给定的延迟时间内该评估值没有恢复正常，则通过电子邮件 SMTP 或短信 SMS 的形式向相关用户报警。

3 设计实现

NIMP 架构的核心代码使用 Python 语言编写，部分代码使用了 JavaScript 和 DHTML。主要部件实现及说明如下。

3.1 内容类型的设计

内容类型是网络资源的抽象，是整个管理平台的基本构件，使用 Python 编写，各内容类型通过向 Zope 注册，成为可被识别、管理、操作和发布的对象。各网络资源都是某种内容类型的一个实例，存储在内容类型数据库中。设备内容类型的定义、功能和注册如下：

```
class Device(ManagedEntity):
#设备属性定义
_properties=ManagedEntity._properties+(
{'id':'manageIp','type':'string','mode':'w'},
…)
factory_type_information=(#内容类型注册到 Zope 的信息
{'id':'Device',
'meta_type':'Device',
…
'actions':       #注册内容类型执行的动作
({'id':'status'
,'name':'状态'
,'action':'deviceStatus'#该动作执行的页面
,'permissions':(permissions.view,)},#许可显示
…)},)
def __init__(self,id):#实例初始化函数
def sysUpTime(self):#函数定义
InitializeClass(Device)#内容类型初始化
```

Device 内容类型是设备类型的基类，具体的设备内容类型可通过继承该类并重载或添加各自的属性和动作来定义。

3.2 服务进程的设计

服务是 NIMP 架构具体功能的实现模块，采用插件的形式开发，以后台进程的方式运行，大致步骤如下：

```
class Daemon(CmdBase):
def __init__(self,noopts=0,keeproot=False):#初始化
def becomeDaemon(self):
try:
pid=os.fork()#创建子进程
except OSError,e:
raise Exception,"%s [%d]" % (e.strerror,e.errno)
if (pid==0):#pid=0 表明是子进程
os.setsid()#设为主会话
os.chdir(WORKDIR)        #改变路径：WORKDIR="/"
os.umask(UMASK)#设置默认权限：UMASK=0
…
else:
os._exit(0)
```

Daemon 是各服务进程的父类，可以在此基础上重载或添加函数，完成相应的功能。

3.3 通过 XML-RPC 向 NIMP 发送事件的步骤

发送事件定义了一个字典类型，表示事件信息。字典中的键至少有 device, component, summary 和 severity，分别表示产生事件的设备名称、部件、摘要和严重程度值。步骤如下：

```
from xmlrpclib import ServerProxy
serv=ServerProxy(Server)#设置服务器
evt={'device':'交换机 1','component':'eth0',
'summary':'eth0 离线','severity':4}#设置事件描述
serv.sendEvent(evt)#发送事件
```

4 结束语

该管理平台部署在山东大学南区网络中心，使用一台 P4 2.8 GHz、1 GB 内存、100 Mb/s 网卡的工作站对主干网上 20 台交换机和 2 台服务器主机进行监控(未开启 NetFlow 相关功能)，通过该系统对管理平台主机某天的 CPU 利用率和网络流速进行监测，显示 CPU 利用率的瞬时峰值为 49.7%，平均值为 18.9%、网络进出流速的瞬时峰值为 20.28 KB/s 和 24.23 KB/s，平均为 17.30 KB/s 和 18.21 KB/s，表明使用一台普通工作站微机完全可以胜任对一个中等规模局域网各项网络信息的监控管理。

参考文献

[1] What is the CM System[Z]. (2001-01-01). http://www.contentmanager.eu.com/history.htm.

[2] Richter S. Zope 3 Developer's Handbook[M]. Anaheim, CA: Amazon Inc., 2005.

[3] 郑洪方, 王玉峰, 王光兴. 基于 IP 网络的物理拓扑自动发现算法[J]. 小型微型计算机系统, 2006, 27(1): 17-21.

[4] Oetiker T. How Does Rrdtool Work[Z]. (2006-02-03). http://oss.oetiker.ch/ rrdtool/doc/rrdtool.en.html.

[5] Gross C. Ajax Patterns and Best Practice[M]. Berkeley: Apress Inc., 2006.