目录

[摘要 4](#_Toc16235)

[Introduction 5](#_Toc16684)

[第1章：绪论 6](#_Toc6652)

[1.1课题研究背景 6](#_Toc27777)

[1.2课题综述 7](#_Toc14470)

[1.2.1课题研究目的和意义 7](#_Toc14954)

[1.2.2发展现状 8](#_Toc12503)

[1.3论文结构 9](#_Toc3961)

[第2章：简单网络管理协议SNMP 10](#_Toc19272)

[2.1 SNMP概述 10](#_Toc30823)

[2.1.1 SNMP基本概念 11](#_Toc22609)

[2.1.2 SNMP发展 11](#_Toc10221)

[2.2SNMP基本框架 13](#_Toc15230)

[2.2.1 SNMP体系结构 13](#_Toc18511)

[2.2.2管理者（Manager） 13](#_Toc8000)

[2.2.3代理者（Agent） 14](#_Toc26225)

[2.3 SNMP管理信息库 16](#_Toc8252)

[2.3.1基本概念 16](#_Toc10867)

[2.3.2管理信息结构 16](#_Toc29689)

[2.3.3对象信息的描述 19](#_Toc29206)

[2.3.4对象信息的编码 20](#_Toc28193)

[2.4 SNMP协议规范 20](#_Toc17287)

[2.4.1 SNMP协议的定义 20](#_Toc8373)

[2.4.2报文格式 21](#_Toc828)

[第3章：系统需求分析 24](#_Toc495)

[3.1系统整体需求分析 24](#_Toc17)

[3.2系统功能性需求分析 26](#_Toc32695)

[3.2.1检测设备取用功能 26](#_Toc13015)

[3.2.2实时监控功能 27](#_Toc25267)

[3.3系统非功能性需求分析 27](#_Toc6423)

[3.3.1系统性能需求分析 27](#_Toc32495)

[3.3.2正确性 27](#_Toc24977)

[3.3.3适应性 28](#_Toc12189)

[3.3.4稳定性 28](#_Toc3154)

[3.4系统环境需求 28](#_Toc26521)

[第4章：系统设计方案 28](#_Toc28609)

[4.1系统总体设计 28](#_Toc32447)

[4.1.1概述 28](#_Toc15126)

[4.1.2架构设计 30](#_Toc27399)

[4.2模块设计 31](#_Toc2435)

[4.2.1数据采集模块 32](#_Toc19346)

[4.2.2数据处理模块 34](#_Toc29584)

[4.2.3数据存储模块 36](#_Toc7431)

[4.2.4信息显示模块 39](#_Toc14463)

[4.3系统约束要求 39](#_Toc27122)

[4.4数据库设计 39](#_Toc10165)

[4.4.1数据库表的设计 39](#_Toc7943)

[4.4.2数据库的E-R图 40](#_Toc29302)

[4.4.3数据库的关系图 42](#_Toc10858)

[4.5异常容错处理模块 43](#_Toc4216)

[4.5.1设备故障处理 43](#_Toc1660)

[4.5.2设备启用处理 43](#_Toc30234)

[4.5.3用户修改设备ip地址处理 44](#_Toc5042)

[4.5.4 ip与mac不匹配处理 44](#_Toc4416)

[第5章：系统的实现与测试 45](#_Toc644)

[5.1系统开发环境 45](#_Toc27171)

[5.1.1开发平台以及开发工具 45](#_Toc21878)

[5.1.2开发环境搭建 45](#_Toc28569)

[5.2系统实现关键技术 45](#_Toc1225)

[5.3系统模块实现 53](#_Toc29343)

[5.3.1数据采集模块 53](#_Toc22617)

[5.3.2数据处理模块 55](#_Toc13097)

[5.3.3后台核心模块 58](#_Toc29097)

[5.3.4信息展示模块 60](#_Toc12425)

[5.4系统的测试 64](#_Toc3898)

[第6章：结论 65](#_Toc16871)

[谢辞 67](#_Toc7061)

[参考文献 68](#_Toc24478)

[附录 69](#_Toc28086)

# 

**基于SNMP的网元接入感知系统研究**

# 摘要

随着互联网的发展，网络已经越来越深入人们的生活，而网络安全和网络设备管理也逐渐成为了大家关注的焦点。随着网络设备的多元化使网络设备管理的功能也变得越来越复杂，网络安全的问题的重要性也在复杂的网络管理中逐渐得到提升。

本文采用层次化的软件设计思想，基于UDP的通信协议，采用WEB应用的展现方式，设计并实现了网元接入的感知研究系统。该系统有效的对网段中的设备信息进行了实时监控，并对接入网段的设备进行检测和统计，对非法设备的入网管理进行了防范措施，提高了网络中的信息安全性并同时便于管理员对设备的可用性进行实时的查询和掌握信息以便设备信息的核对。系统对网络设备进行参数配置和参数查询，并通过SNMP简单网络管理协议访问网络设备的MIB管理信息库。

本文首先对传统的网络非法设备的入网防范技术进行了研究，在此技术了提出了与传统的网络设备监控的实现方式不同的本文实现技术。其次对系统用到的相关技术进行了详细的介绍，包括：SNMP协议报文传输技术，MIB管理信息库的访问，设备数据采集技术，后台服务器的实现，WEB前端的展现，ARP表项的工作原理。然后，本文根据网元接入感知系统的特点和需要给出了详细的系统分析报告，在此基础上，结合层次的软件设计的思想，实现了网元接入感知系统的总体结构设计以及其数据采集模块，数据处理模块，数据展示模块和非法接入报警模块的设计。并设计了系统的数据库以及相应的数据合法协议。根据系统架构的设计，本文从数据采集技术的封装，数据过滤，分析以及统计算法的实现，报警系统的实现，前端界面的真实几个方面来对系统进行了具体的编程实现和测试，最后，本文该研究的相关工作进行总结，包括论文研究的成果以及目前存在的不足，为下一步研究指明了方向。

关键词：SNMP协议，交换机，MIB，入网感知，ARP

# Introduction

Network security and network management have become the focus of attention with the development of the Internet and network has become more integrated into people's lives.The importance of the network security issues has been raised in the complex network management gradually because of the diversification of network equipment and which enables network equipment management functions are becoming increasingly complex.

In the current paper, Perception of the NE access system was designed and implemented based on hierarchical software design and UDP protocol by visualization through web. The system collects the devices information on the network segment and avoids the illegal devices,which improves the security of network and facilitates to manage the devices for manager at the same time . System configuration and query parameters to network equipment,as well as access equipment MIB through SNMP protocol.

There are some steps for the work.Firstly,we study traditional techniques that avoid illegal devices into network and proposed our method different from it based on the old method.Text works achieve algorithm of the monitoring equipment through the working principle of switch mainly.Secondly,we introduce some related technology about the work including SNMP protocol,the technology of transmission, MIB management,collection of devices data,server structures,visualization technology,working principle of ARP.Thirdly,system analysis report is given according to the access NE perceived characteristics and achieve the overall structural design of the access NE perception system and its data acquisition module, data processing module, data display module and unauthorized access alarm module design based on hierarchical software design ideas.In addition to that,design of the system's database and corresponding data legal agreement also been finished.Finally,the coding of the work is finished according to the design modules and the system has been tested by professional technology.To the end of the work,some summarize about the graduation project and paper is made including some conclusion and shortage.

**Key Words：**SNMP Protocol Switch MIB Perception Network ARP

# 第1章：绪论

## **1.1课题研究背景**

互联网的出现将人们带入了一个高信息时代，它在促进社会经济发展、提高国民生活质量、提高工作效率、促进商业繁荣、促进社会信息化等方面产生了重大的意义。它提供了不同于传统的，崭新的交流和交往方式。从人们基本的工作、学习、娱乐、到社会的政治、经济和文化领域以及社会公共服务的各个行业都在不同程度上依赖于互联网。在计算机为人类带来方便和便利的同时，人们也意识到了网络安全问题的存在，人们发现自己的计算机信息系统不断受到侵害，其形式的多样化，技术的先进化以及复杂化，使人们难以防范。这也使得互联网中的的安全问题变得更加的突出以及复杂[[[1]](#footnote-0)]。随着技术的发展，大部分单位都使用自己的局域网。怎样使计算机网络系统避免受到破坏，使系统的安全可靠性得到提高，已经成为了现实中人们关注和务必解决的问题。目前几年，网络泄密案件多次发生在我国一些重要部门如政府部门、科研院所、军事防备等，严重损害了国家安全和利益，甚至国防事业上也造成了一定的损失。这主要有两个方面的原因，[一方面是互联网系统提供了大批丰富多样的资源让用户可以彼此共享；另一方面，网络系统的大量信息传输和存储让整个系统的脆弱性和复杂性遭受了考验，](http://www.paperfree.cn/report/part/20160426/3038639186761756/htmls/sentence_detail/2.htm" \t "http://www.paperfree.cn/report/part/20160426/3038639186761756/htmls/detail_report/right)[使得网络安全中的各种问题频发凸显,](http://www.paperfree.cn/report/part/20160426/3038639186761756/htmls/sentence_detail/3.htm" \t "http://www.paperfree.cn/report/part/20160426/3038639186761756/htmls/detail_report/right)增加了网络的可攻击性和可受威胁性。据相关调查显示，绝大多数人都把注意力集中在来自外部的网络安全问题上，而事实上，有很一大部分的网络安全问题是发生在单位内部。相比于来自外部的网络安全威胁，来自单位内部的网络安全威胁显得更为隐蔽、更易发生、破坏性也更大[[[2]](#footnote-1)]。

一些重要单位为了让其办公效率和信息化建设的水平得到提高，目前已经基于TCP/IP 和以太网技术建成了千兆为主杆，百兆到桌面的局域网，因为其所承担任务的重要性和特殊性，所以必须让其局域网的安全得到保障，与此同时也面临着如何对入网网元设备的接入进行有效监控与管理的困扰。当前单位的局域网大部分都是基于TCP/IP 协议而建设的，而在 TCP/IP 协议广泛采用的过程中，有许多网络应用的访问权限设置是基于 IP 地址而不是基于用户的。由于修改 IP 地址比较容易，单位内部的个别人员通过对 IP 地址的修改不但可以获得此类应用访问权限，而且还可以对身份进行隐藏，从而逃避追查[[[3]](#footnote-2)]。因此人们希望能够得到一种对入网网元设备的接入进行有效的监控与管理的方法来防止此类事件的发生，维护网络安全及网络正常运行。

尽管这些单位对接入其局域网的网元设备都采取了严格的审批管理，对每一台入网的网元设备都需要进行入网申请流程，并在其得到相关管理部门的批准后方可接入其局域网。但为了保证网络安全，维护网络运行的正常秩序，非常需要一套能够对局域网上的网元设备接入情况进行实时监控并对出现的违规事件进行报警提示、及时处理和事后统计的软件。目前市面上的网络管理软件（如 HP 或 CISCO 公司的网管产品）大都针对网络的综合管理，且价格昂贵，没有特别合适的入网网元设备监控与管理的产品可以适用于局域网。因此，我们设立了入网网元设备监控与管理系统的设计与实现这一课题，对局域网上的入网网元设备接入情况进行自动化的网络监控，对系统的使用实现自动检查和监督，达到进行实时监控与有效管理的要求。

## **1.2课题综述**

### 1.2.1课题研究目的和意义

**1.目的**

本课题从解决非法网元设备接入局域网的实际问题出发，研究在各种网络环境下快速发现非法网元设备的接入问题并且随时对整个局域网采取监控的状态的解决方法，该解决方法应该具有稳定性好、网络配置量小、适应性强、定位准确、实用性强等特点。

**2.意义**

网络中非法设备的接入对网络系统的安全正常运行有很大的影响以及潜在的威胁，所以解决非法设备接入网络的安全的问题，对整个网络的正常运行至关重要，对保障单位局域网的安全信息以及提高单位的网络管理水平有着重要的意义。在我国的政府部门、军事部门、大型企业以及高校中，为了工作的方便以及信息的安全性，大部分都已经建立起自己的局域网，而大部分的局域网都是基于TCP/IP协议来建立的，所以我们设计的网元检测感知系统对这些网络问题有着重大的意义。

我们设计的这个系统是采用WEB技术来设计的，为网络的管理提供了一种新颖的方法，管理员可以通过WEB在多个站点多网络设备进行监控，而不必仅仅拘泥于传统的网络管理工作站。基于WEB的网络设备监控平台对非法设备的接入能够为管理员提供一个图形化界面，并且能够很好的解决平台结构的差异而产生的不兼容问题。所以基于WEB的网元接入感知系统技术是值得探究的一项技术。

### 1.2.2发展现状

目前对防范非法网元接入的方法主要是对主机的非法接入的防范，传统的主机非法接入的防范方法主要有三种：

**1.交换机的控制**：

把 MAC 地址同交换机端口绑定的静态表项手动添加到交换机的 MAC 表中，形成〔MAC，交换机端口〕映射对。只有当判断主机的 MAC 地址同它所连接的交换机的端口绑定的 MAC 地址吻合时，交换机才会对此主机进行数据转发。但是在局域网规模较大的情况下，这种方法会增加网络管理人员的工作量，而且也会使网络的维护工作变得很麻烦。该技术主要是对于IP 地址 MAC地址成对修改所产生的主机非法接入情况进行阻止，对单一修改 IP 地址的非法接入却无能为力[[[4]](#footnote-3)]。

**2.路由器的隔离:**

把 IP 地址同 MAC 地址绑定的静态表项手动添加路由器或三层交换机的 ARP 表中，形成〔IP－MAC〕映射对。仅在用户的 IP－MAC 映射对与设置相同的情况下，数据帧可顺利传送。如果采用这种方法，需要在路由器的 ARP 中绑定所有主机的 IP 地址 MAC 地址，这种方法的缺乏可行性。一旦采取成对修改 IP 地址和 MAC 地址的方法接入，这种方法就失效了。这一技术主要用于阻止单一盗用 IP 地址的主机访问它所在子网外的主机[[[5]](#footnote-4)]。

**3.用户认证：**

主要是采用防火墙与代理服务器相结合的方法来解决非法接入的主机访问外部网络的问题：防火墙用来隔离内部网络和外部网络的网络安全系统，用户通过代理服务器访问外部网络。只有在系统能够对用户所提供的身份和认证信息进行了正确的审查后，代理服务器才能够对外网资源进行访问。不过这种方法也有不足之处，因为使用代理服务器对外部网络进行访问的时候对用户是不透明，并且对每一种服务器都要设计一个代理模块，建立对应的网关层，所以实现起来比较复杂。另外，这种方法也并不能控制非法接入主机对内部网络资源的访问。

这几种入网主机管理技术在解决非法主机接入的问题上都存在一定的局限性。这些技术采用的都是被动防护的思想，它们并不能解决针对非法主机接入的及时发现、准确定位和实时监控等问题。

## **1.3论文结构**

本文首先介绍了课题的研究背景和研究意义，然后对网络安全和网元入网相关技术进行了分析和研究，针对目前的局域网的特点和企业用户的需求设计了一个对网元入网的合法性检测和非法入网的报警系统，他的目的就是及时发现入网的非法设备并对其进行快速的处理，同时对该网段进行合理的监控，方便管理员对设备的取用情况进行掌握和核对，也对传统的检测入网设备的方法进行了有效的改进。

针对网元接入感知系统，本文研究的主要内容包括：基于简单网络管理协议SNMP的相关介绍，网元接入感知数据采集模块的设计和实现，对非法设备的报警系统的处理系统的设计和实现。本文共分为六章，具体的章节安排如下：

1. ：论文的绪论部分，主要对课题研究的背景和目的以及相关意义进行了简单的介绍，在阅读了大量的相关文献以及充分的调研基础上，对网元入网检测以及防范措施的方法上进行了分析和研究，并对他们之间优缺点进行了阐述，在此基础了提出了论文的研究内容。
2. ：对本文用到的相关技术进行了说明。详细介绍了简单网络管理协议SNMP的发展和基本框架，包括snmp协议的规范，MIB管理信息库的结构，报文的格式和传输的实现过程。
3. ：系统需求分析，主要对系统的总体需求分析，功能性需求分析，性能需求分析进行了详细的介绍；
4. ：是系统的详细设计方案，主要是对系统的总体架构设计，模块设计，数据存储设计，容错处理设计进行了详细的描述；
5. ：系统的实现和测试，主要对系统的开发环境的搭建和相关参数的配置进行了描述，对系统的关键技术进行详细的介绍，包括对通过snmp协议对网络设备上进行网元的相关数据的采集技术的实现和封装的技术，对采集器上的数据进行过滤，统计，分析和整理来对设备的合法性进行检测和判断的算法，在此基础上判断是否触发报警系统以及对报警系统模块的实现和技术的详细介绍。针对每个模块和技术，本文都给出了核心代码的实现和部分数据的展示，最后通过对系统的测试给出了相关的测试截图来说明系统在实际环境中的可行性稳定性。
6. ：结论部分。该部分主要是对本文所设计的系统进行总结，包括论文研究的成果以及目前存在的不足，为下一步研究指明了方向。

# 第2章：简单网络管理协议SNMP

## **2.1 SNMP概述**

SNMP(Simple Network Management Protocol)即简单网络管理协议，他提供了网络和系统管理者底层的框架，使得系统和管理员的应用系统，网络设备，操作系统等能够得到远程的检测和配置管理。由于snmp简单可靠，并且具备广泛的TCP/IP基础，受到了众多厂商的欢迎，成为目前最为广泛的网管协议。

### 2.1.1 SNMP基本概念

简单网络管理协议（SNMP）,它由一组网络管理的标准组成，包括一个数据库模型（database schema），一个应用层协议（application layer protocol）和一组资源对象，互联网工程工作小组（IETF,internet Engineering Task Force）定义的internet协议簇之中的一部分定义了该协议，因此SNMP在一定程度上受到了标准网络管理框架的影响[[[6]](#footnote-5)]。

### 2.1.2 SNMP发展

在20世纪70年代，SNMP网络管理就出现在最早的TCP/IP网络ARPANED的研究实验过程中，人们采用简单实用的互联网控制协议（ICMP）对其进行管理。1987年11月，简单网管监控协议SGMP（Simple Gateway Monitoring Protocol）发布，开启了提供专用网络工具的纪元，网络管理摒弃了原始的利用ICMP来探测网元是否在线等监控手段的时代[[[7]](#footnote-6)]。

1990年5月，Internet工程任务组（IETF）在RFC1157上正式发布了SNMPv1，并将其作为TCP/IP的正式标准。SNMPv1的MIB和SMI都比较简单，容易实现，但是存在很多安全缺陷，它仅提供简单的口令来验证访问的合法性，缺少对身份的认证（Authentication）,传递的报文也没有进行加密（Privacy）机制的处理，很容易在传递的过程中被截取，造成信息的泄露，并且在实现大量数据传输的过程中也是有困难的，针对以上SNMPv1的缺点，IETF做了大量的工作来改进，并在1993年的时候，发布了SNMPv2。

SNMPv2针对SNMPv1在管理大型网络上的不足，对SNMP进行了一系列的扩充,增加了两种新的SNMP报文：GetBulk和Inform。GetBulk可以用来高效率的查询和处理批量的数据，Inform可以通过被NMS(网络管理系统)发送Trap信息到另一个NMS来实现分布式管理结构。加强了数据定义语言，扩展了数据类型，提高了海量数据的传输能力，并且增加了容错能力，性能和效率也得到了有效的改进，但是SNMPv2并没有完全实现预期的目标，在安全机制方面仍然存在着重大的缺陷，比如用户登录的身份验证，预防重复操作性，分析信息的完整性，加密、授权和完整性控制以及一定程度的远程安全配置和管理能力等都没有具体实现，并且还提高了自身的复杂性，于是，IETF在1996年的时候对SNMPv2进行改版并发布了SNMPv2c,SNMPv2c保留了SNMPv2的相关功能，并沿用SNMPv1的安全机制，由于开发的难度降低了，所以SNMPv2c成为了SNMPv2在实际应用的标准。

1999年4月，IETF正式发布了SNMPv3。SNMPv3是建立在SNMPv1和SNMPv2的基础上的最新研究成果，他解决了SNMPv2未能解决的安全机制方面的问题，不但包含了SNMPv1和SNMPv2的已有的功能实现的框架体系，还包含了验证服务在内的一种全新的安全机制。它采用了USM（基于用户的安全模型）和VACM（基于视图的访问控制模型）技术，USM提供了认证和加密功能，VACM确定用户是否允许访问特定的MIB对象以及访问方式。SNMPv3定义了统一的SNMP管理体系结构，总结了网络界对SNMP安全特性的发展，具有很强的适应性，一方面能够对简单的网络进行有效的管理，另一方面又能够对大型复杂网络的管理需求基于满足，已经成为计算机网络中网络管理的实际标准[[[8]](#footnote-7)]。[13]图2.1为SNMP的发展历程[[[9]](#footnote-8)]

SGMP

SNMPV1

SNMPsec

RMON

SNMPV2

SNMPV2u

SNMPV2\*

SNMPV2C

SNMPV3

图2.1 SNMP发展历程

## **2.2SNMP基本框架**

### 2.2.1 SNMP体系结构

SNMP的工作原理是基于TCP/IP协议族上的，属于应用层协议。它采用典型的Manager/Agent体系结构。SNMP管理系统由四个关键元素组成，分别是：管理者（Manager），代理（Agent），管理者和代理之间的通信协议（SNMP），管理信息库（MIB）。图2-2是网络管理的一般模型:

Network Management Station

MIBS

Manager

管理操作 响应/通知

Managed Resources

管理操作

响应通知

Manged Entity

Agent

MIB

Agent

**图2-2网络管理的一般模型**

### 2.2.2管理者（Manager）

首先介绍下管理站（Management Station），管理站是管理员和管理系统之间的接口，管理员的命令将会被管理站转换成对远程网络元素的监视、操作、控制，管理站同时也会从被管设备的MIB中获取相关的信息[[[10]](#footnote-9)]。管理站是整个网络控制的中心，可以是一台主机也可以是一个程序运行平台，而管理者是运行在管理站上的一组具有分析数据，发现故障，监视网络性能等功能的管理程序，它通常是作为一个独立的设备运行在管理站上的，通过轮询的方式获得各个设备的具体信息，然后对其进行分析处理，从而负责完成网络管理的各项功能，是整个网络控制的核心。图2-3是管理者的功能。

Get

Set

Manager

Trap

**图2-3 管理者功能**

Get:读取网络设备的状态信息，通过发送Get-Request报完来完成

Set:通过网管系统来远程配置设备参数，通过发送报文Set-Request来完成。

Trap:自陷阱，能够让管理站随时可以通过代理主动而获得设备的一些重要信息，属于非请求性质的，是网络设备主动向管理站发送信息。

### 2.2.3代理者（Agent）

代理者是驻留在被管理对象中的管理软件的服务端，是一段运行的后台的管理程序。一切配置了SNMP服务的设备都可以作为代理者，比如：交换机，主机，路由器，网桥等。因为该设备的代理都是对位提供了统一的、标准的SNMP访问接口。代理者主要是对管理站发来的动作请求和信息请求进行应答并且随时以产生陷阱（Trap）的方式主动向管理者报告相关设备的重要信息和事件。图2-4是管理者和代理之间的关系:

SNMP Management Station SNMP Agent

Trap

GetResponse

SetReques

tNextReques

Ge

GetRequest

GetRequest

SNMP Agent

UDP

IP

Protocol Dependent On Network

Trap

GetResponse

SetRequest

GetNextRequest

GetRequest

SNMP Manager

UDP

IP

Protocol Dependent On Network

Managed Resource

Management

Application

Manage

SNMP

Managedn Entity

Application

Object

SNMP Manager

UDP

IP

Protocol Dependent On Network

SNMP Message

SNMP Manager

UDP

IP

Protocol Dependent On Network

Internet

**图2-4 SNMP管理者和代理之间的关系**

**The relationship between manager and agent**

MIB对管理者和代理之间进行了接口的统一，同时对每个设备中的被管理对象进行了详细的定义。管理站和代理者通过实现相应的MIB对象来互相识别对方的数据，从而完成通信。代理者在UDP的161端口接收NMS(Network Management Station)的读写请求，管理者在UDP的161端口接收代理者的通告消息。所以一旦获得设备的访问权限，就可以访问设备信息了。由于采用的是UDP协议，所以不需要代理和管理者之间保持联系。

## **2.3 SNMP管理信息库**

### 2.3.1基本概念

管理信息库（Management Information Base,MIB）是网络管理数据的标准，该标准制定了一系列规则来对网络代理设备必须保存的数据项目，数据类型，以及允许在每个数据项目中的操作进行管理[[[11]](#footnote-10)]。MIB相当于一个虚拟的树形数据库，存放着网络中可管理的网元节点资源的信息，这些节点以对象的形式被保存在MIB中。通过对这些对象信息的访问来获取个网元设备的具体信息，这些信息由管理者和代理者共享，通过对这种获取对象的方式来对网元设备进行监视，以及修改相应的参数来完成一个动作，从而实现基本的网络管理。

MIB管理信息库随时保持着动态刷新，它包括网元设备的配置信息，数据通信的统计信息，端口的性能数据，安全信息和设备私有信息。这些信息作为网络管理的数据来源，更新在MIB信息库中。MIB分为公共的MIB和私有的MIB，公共的的MIB有标准的文献进行定义，而私有的MIB主要是企业内部自定义的MIB，不同的企业为了丰富各自管理元素的多样性，在公共的MIB基础上定义了自己私有的MIB数据。

### 2.3.2管理信息结构

MIB中保存的对象的基本格式由SMI（Structure of Management Information）来定义。管理信息结构SMI是SNMP的基础部分，对SNMP框架中的信息的组成，结构和表示进行了定义，奠定了MIB对象和协议进行交换信息的基础。互联网了定义了多个版本的管理信息结构，目前最通用的是国际互联网MIB-II（RFC 1213），它由三个部分组成：对象标识符OID（Object Identifier）、对象信息的描述和对象信息的编码[[[12]](#footnote-11)]。

1. 对象标识符OID

对象标识符即被管对象的命名，标识一个MIB-II对象的数据类型。MIB的树形结构提供被管信息的分级组织模型，通过信息的逻辑分类来构造树的分枝，被管理的对象处于MIB分枝的叶节点上。也叫对象命名树（object name tree）。每个叶节点上的对象就用对象标识符来进行命名，对象标识符由一组句点隔开的整数唯一标识，并确定在对象命名树上的唯一位置。通过对象命名树的树状结构，可以对一个指定的MIB对象通过根节点到该对象的叶节点的路径进行查找。每一个对象除开有一个唯一的OID还有一段简单的文字对对应的进行描述，叫做该对象的名字（name），用于方便管理员的操作和记忆。图2-5是MIB的树状分层结构图:

ROOT

JOINT-ISO-CCITT(2)

IS O(0)

CCITT(0)

ORG(3)

DOD(6)

INTERNET(1)

PRIVATE(4)

EXPERIMENTAL(3)

DIRECTOR(1)

MGMT(2)

MIB-2(1)

ENTERPRISES(1)

SYSTEM(1) INTERFACE(2) AT(3) IP(4) ICMP(5) TCP(6) UDP(7) EGP(8) SNMP(11)

**图2-5 MIB分层结构图（对象命名树）**

对象命名树的根节点没有名字和编码，由上图可知，树的顶级对象有三个，即CCITT,ISO,JOINT-ISO-CCITT，它们都有个自的子树，并且分别由国际标准化组织，国际电信联盟，国际标准组织和国际电信联盟的结合来管理。ISO下应该有四个节点（这里没有全部标出），其中的一个ORG（标号是3）是被标识的组织，其下面的一个子树是分配给美国国防部（Department of Defense）的，其下面的子节点internet是分配给internet体系结构委员会来管理的。Internet下有四个子树，PEIVATE下的一个子节点为ENTERPRISES，企业可以在该节点下申请自己的子树，然后在子树下定义自己专门自己的设备管理所需要的管理信息对象。树中的子节点的命名规则可以用下面的一个例子来说明，这里采用MIB-II中的一个管理对象SYSNAME为例：

名字表示：ISO.ORG.INTERNET.MGMT.MIB-2.STSTEM.SYSNAME

数字表示：1.3.6.1.2.1.1.5

下面给出MIB-II定义的功能组，包括了管理对象的10个典型的网络信息。具体内容如表2-1所示：

**表2-1 管理对象的网络信息**

|  |  |
| --- | --- |
| 系统组（system group） | 提供了系统的基本信息 |
| 接口组（interface group） | 提供了关于主机的统计信息和配置信息，该接口一定要实现。该接口组的对象的主要功能是性能管理和故障管理 |
| 地址转换组（address translation group） | 提供了网络地址到物理地址的一个映射表 |
| IP组（IP group） | 提供与IP协议相关的信息 |
| ICMP组（ICMP group） | 提供了和ICMP的相关信息 |
| TCP组（TCP group） | 提供了和TCP协议相关的操作和实现的信息 |
| UDP组（ UDP group） | 提供了关于UDP数据包和本地接收端点的信息 |
| EGP组（ EGP 组） | 提供了和EGP路由器转发和接受EGP报文的相关信息，和EGP邻居的详细信息等 |
| 传输组（transmission group） | 提供关于每个系统接口的传输模式和访问协议的信息 |
| SNMP 组（SNMP group） | 提供关于系统中snmp的实现和运行信息 |

### 2.3.3对象信息的描述

MIB中的管理对象采用抽象语法表示法ASN.1（Abstract Syntax Notation One）来描述。它在SNMP中定义了SNMP的协议数据单元PDU（Protocol Data Unit）和管理对象MIB的基本格式。PDU是网络中传输的数据包，在物理上，每一种SNMP操作都对应着一个PDU。SNMP使用的是ASN.1的一个子集，SMI就是SNMP利用ASN.1的语言特性定义的一些类型宏和自定义类型而组成的结构。SMI规定了管理对象的描述必须由五个字段组成[7]：对象描述OBJECT、对象类型的抽象语法SYNTAX、存取权限ACCESS、状态STATUS和对象类型文本描述DEFINITION。在SMI中，数据类型分为两类，一类是通用数据类型（Universal Data Type）。另一类是范用数据类型（Application-Wide Data Type）

通用数据类型如表2-2所示：

**表2-2 通用数据类型**

|  |  |
| --- | --- |
| 整型INTERGER | 基本的数据类型 |
| 字符串型OCTET STRING | 长度从0到65535的字符序列 |
| 对象标识符OBJECT IDENTIFIER | 对象的名字，用点分开的十进制表示，反映了它在OID树中的具体位置 |
| 空类型NULL | 占字符 |
| 聚合类型SEQUENCE SEQUENCE-OF | 用于列表和表格 |

应用数据类型如表2-3所示：

**表2-3 应用数据类型**

|  |  |
| --- | --- |
| 网络地址（Network Address） | 表示不同类型的网络地址 |
| IP地址（IP Address） | 以网络序表示的32位网络地址 |
| 计数器（Counter） | 计数器， 从0到232-1递增的整数，到最大值后归零 |
| 量规(Gauge) | 计数值，取值范围在0到232-1之间，达到最大值后锁定，直到复位 |
| 时间变量（Time Ticks） | 计算从某一时刻开始以0.01秒为单位递增的时间计数值，取值范围0到232-1 |
| 模糊变量（Opaque） | 特殊的数据类型，它把数据转换成OCTET STRING，用于记录任意的ASN.1数据 |

### 2.3.4对象信息的编码

管理者和代理之间的通信要求对管理信息的统一编码，于是SMI就规定了管理信息的编码采用基本编码规则BER(Basic Encoding Rules)。BER对管理对象的信息编码和解码规则进行了定义，使得他们可以在传输介质上正常传输。BER定义了标签（Tag），长度（Length），内容（Value）三个字段来表示数据[8][[[13]](#footnote-12)]，TLV编码方式的各个字段的含义如下：

Tag：表示ASN.1类型以及类型的类别和编码是基本的还是结构化的；

Length：表示实际取值表示法的长度，用于记录任意ASN.1数据。

Value：表示ASN.1类型作为字符串的取值。

这是一种递归的结构，对于任意一个或多个字段的ASN.1取值，TLV编码的“Value”部分本身就是由一个或者多个TLV组成。每个字段均是由一个或者多个8位组成，字节的最高有效位在字节的最左边（bit7），最低有效位在最右边（bit0）。-

## **2.4 SNMP协议规范**

### 2.4.1 SNMP协议的定义

传输层定义了SNMP报文的传输方式和方法，大多数的SNMP实现使用的是无连接的UDP协议，也可以使用CLTS（Connectionless Transport Service）协议，该协议在OSI结构中是支持SNMP操作的。SNMP的Agent在161端口对管理者发送的GetRequest,GetNextRequest,SetRequest命令进行监听，NMS在162端口对代理者发来的Trap消息进行监听。SNMP通过UDP分组进行简单的报文传递，当SNMP的请求的的报文发送出去后，接收端在收到请求后，对该请求进行处理，处理完后再回应一个SNMP报文：当完成一次请求和应答的实现后，就完成可一次SNMP操作。但是在这些操作中，Trap操作是不需要接收方回应的。IP,UDP,SNMP报文,SNMP和UDP的关系如图2-6所示：

IP数据报

UDP数据报

SNMP报文

公共SNMP首部 get/set首部

IP首部 UDP首部 版本（0） 共同体 PDU类型（0~3） 请求标识 差错状态（0~5） 差错索引 名称 值

Get/set变量部分

PDU类型 企业 代理地址 Trap类型（0~6） 特定代码 时间戳 名称 值

Trap数据报 变量

**图2-6 IP,UDP,SNMP报文,SNMP和UDP的关系**

### 2.4.2报文格式

SNMP报文由三部分组成，分别是：版本标识符（Version Identufuer），SNMP共同体标识符（Community）和一个协议数据单元（PDU），报文的基本格式如图2-7所示：

PDU头 Version Community SNMP PDU

SNMP报文格式

# **图2-7 SNMP报文格式**

1. 版本标识符Version：是Integer型，表示SNMP的版本信息，它值的定义是版本号减一，FRC1175是第一版，所以第一版的取值是0，默认是第一版。SNMPV2取值为1。
2. 共同体标识符Community：是Octet String 型，每一个进行通信的SNMP组都拥有一个八进制串形式的无二义的共同体名（Community Name），这是对访问权限的控制，相当于口令，可以用户自己定义，它与消息体一起传输。
3. 协议数据单元SNMP PDU：PDU协议数据单元是SNMP消息的具体编码，它采用的是抽象句法ASN.1格式来进行定义的。SNMPv1版本（RFC1157）中一共定义了五种协议数据单元，分别是：GetRequest-PDU，GetNextRequest-PDU，SetRequest，GetResponse-PDU，Trap-PDU，前面四种协议数据单元的格式是一样的，Trap有一点区别，具体格式如图2-8所示：

PDU type(s) Request-id Error-status Error-index Variable-bindings

Name(1) Value(1) Name(2) Value(2) ... ... ... ... Name Value

# (A)PDUtype:getRequest,GetNextRequest,SetRequest,GetResponse

PDU type(b) enterprise Agentaddr Generctrap Specifictrap Timestamp Variable-bingdings

# 

Name(1) Value(1) Name(2) Value(2) ... ... ... ... Name value

(B)PDU type:Trap

**图2-8 SNMP PDU 字段结构**

PDU公共部分结构的字段含义如表2-4所示

**表2-4 PDU公共部分结构字段含义**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 含义 |
| PDU type |  | PDU类型：  0代表GetRequest-PDU  1代表GetNextRequest-PDU  2代表SetRequest  3代表GetResponse-PDU  4代表Trap-PDU |
| Request-id | Integer | 发送实体给每一个PDU赋一个ID，使得每一个到同一个代理的请求能够被唯一识别，代理应答时返回同样的Request-id使得双方的应答和请求可以匹配，并且也能够让双方有能力处理UDP中产生的重复发送的消息 |
| Error-status | Integer | 表示在处理请求时出现的异常，取值为：  0: noError  1: tooBIG  2: noSuchName  3: badValue  4: readOnly  5: getErr |
| Error-index | Integer | 当error-status非0时，如果错误由variablebindings里绑定的某个对象引起，error-index设置为这个对象在variableebindings里的索引。 |
| Variable-bindings | Sequence | 所要求的实例列表 |

Trap PDU字段含义如表2-5所示

**表2-5 Trap PDU字段含义**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| PDU type |  | 4代表trap |
| Enterprise | Object-Identifier | 基于sysObjectID所产生的Trap的网络管理系统。如果是企业自定义的Trap，此值为企业在enterprise子树下的注册树 |
| Agent-addr | Integer | 产生Trap的SNMP实体地址，一般为代理地址 |
| Generic-trap | Integer | 通用Trap类型，取值一般为：  0: coldStart  1: warmStart  2: linkDown  3: linkup  4: authentication-Failure  5: egpNeighborLoss  6: enterprises-Specific |
| Specific-trap | Integer | 当Generic-trap为6时，Specific-trap指明确具体的企业自定义Trap类型。 |
| Time-stamp | TimeTicks | 上次初始化网络实体和产生Trap的时间间隔。包含sysUpTime值 |
| Variablebindings | Sequence of | 和trap相关的附加信息 |

SNMP五种协议数据类型的具体功能如下：

1. GetRequest-PDU：该操作用来提供一个SNMP协议实体向代理读取一个或多个实例值，请 求者可以在Variable-bindings中绑定多个对象实例，如果处理成功，将返回这些实例的 值。为了提高效率，GetRequest运行一次将提交多个对象实例，这样也不会使某一个实 例不能取值而导致整个请求失败。

b.GetNextRequest-PDU：该操作用来请求所提供的对象标识符或实例的下一个可用的实例标 识和值。适合于遍历各个表或快速的查询连续对象。

c.SetRequest：该操作用于管理者给管理设备设置相应的管理信息。对象标识符作为参量和 该报文一起发送。如果代理能处理带有规定值的SetRequest命令，则发回一个Request 包确认操作有效，如果出错，则创建一个Response包并将相关的出错消息发回给请求 者。

d.GetResponse-PDU：该操作是用来判断一个请求是否成功发送，当操作成功时，error-status 的值为0,；当操作失败时，error-status的值为相应的错误代码，如如果错误由 variablebindings里绑定的某个对象引起，error-index设置为这个对象在variableebindings 里的索引。

f.Trap-PDU：该操作用来SNMP实体应网络管理代理应用程序的请求发出Trap，它被用于向 管理站异步的通报重要事件的发生。

# 第3章：系统需求分析

## **3.1系统整体需求分析**

基于用户目前的实际网络情况，整网内各类物理设备、应用业务、中间件、数据库、终端设备、特别是摄像头和传感器数量巨大，物理位置分布过散，给物品管理带来了一定的困难。当内部工作人员从物品管理库中提取出设备，是否真正用于网络之中，或者何时接入网络之中，管理员是无法知晓的。对于这样庞大而复杂的网络环境，实时掌握设备的接入状况，是十分必要的。

本文采用的是在基于SNMP协议上通过访问局域网的固定物理设备，比如交换机，用轮询的方式对这些物理设备上记录的接入局域网的网元设备进行查询，从而及时发现新加入的设备，并采用一定的方式对这些新加入的设备进行检测他们的合法性，从而统计出相关的信息汇报给管理员。一般对于在局域网内做网元入网管理最直接的方法是采用MAC地址绑定，但是在有些应用场景,MAC地址绑定并不适合，并且有些用户不想做绑定，因为在使用上会受到限制，而本文所采用的方法就很好的规避了MAC绑定的局限性，通过局域网中固定设备的工作原理，监控网元的实际接入情况来达到实时掌握网络中的具体情况。系统的用例图如图3-1所示：

《include》

发现新接入设备

监控网络

检测设备合法性

显示合法事件列表

采集设备信息

监控系统

标记违法设备

启动报警系统

相应报警请求

启动系统

报警系统

通知管理员

查询设备信息

删除设备信息

管理员

解除设备警报

处理违法设备

**图 3-1 系统用例图**

## **3.2系统功能性需求分析**

网元接入检测系统主要是通过网络中交换机的工作原理来实现对新接入设备的感知功能。利用SNMP协议获取到交换机的信息以及它的ARP表，然后对表上的设备信息进行统计和归纳，每隔一定的时间访问一次相同设备，一旦发现新接入的设备就进行记录，并通过检测设备的合法性来生成时间列表。具体的系统功能图如图3-2所示：

网 元 接 入 检 测 系 统

检 测 设 备 取 用

实 时 监 控

判断合法性

发现非法设备

产生告警信息

生成接入信息事件列表

发现新的设备

**图3-2 系统功能需求图**

### 3.2.1检测设备取用功能

网络的飞速发展，对于大型的网络设备管理变得更为复杂，管理员对网段设备的取用与否很难在短时间内进行准确的统计并且核实，网元键入检测系统在一定程度上对这个问题给出了一个比较高效的解决办法。对于每个新接入网络中的合法设备，系统将自动生成接入信息合法事件的列表，支持将一段时间内的接入事件生成列表导出，从而方便管理员核对设备的取用情况。

### 3.2.2实时监控功能

因为系统采用的轮询的方式，每隔一定的时间对相关的物理设备发出请求进行收集该设备的网络连接信息，一旦发现非法的设备接入网络，就产生告警，从而通知管理员以便及时阻断内部人员有意或者无意的非法设备接入的违规事件。从而很好的对该网络进行了监视功能，对该局域网的安全性起到了很好的维护作用。

## **3.3系统非功能性需求分析**

### 3.3.1系统性能需求分析

交换机的arp表项会有一个默认的老化时间（本文实验网络中的是20分钟，可通过命令arp expire-time修改老化时间），当到达20分钟时，发N次arp探测（可通过命令arp detect-times修改arp探测次数），若一直没有收到回应报文，则表项删除。

如要系统能够及时发现新接入的设备，需要保证采集每个交换机的arp表项记录的前后间隔时间不能长于arp表项的老化时间。假设，将对交换机arp表项的默认采集时间间隔设定为300s。理论上，对于新接入网络的设备，系统最长需要300s的时间可以发现。

注：这里需要排除一种特殊情况，即有内部人员恶意操作交换机将arp表项手动删除。

### 3.3.2正确性

一般情况下对交换机的ARP表项的默认采集时间间隔都不会很长，所以一旦有新设备加入该网络，系统就能够及时的发现，具有较高的精确性和正确性。

### 3.3.3适应性

网络的发展与进步已经促使用户希望网络是透明进行工作的，这种需求需要有弹性，速度和安全的控制，通过交换技术，能够满足今天以及未来的用户需求，所以交换机已经越来越成为了广大用户和企业必不可少的网络设备了。SNMP协议的发展使得它本身具有更加简单易操作并且能够让企业自定义一些符合他们自己的设备管理功能，所以绝大部分的交换机上都配备了该协议相关的设置，这就使得我们这个系统的适用性和推广度得到了大幅度的提高。

### 3.3.4稳定性

大数据时代的到来加大了对一个管理系统的稳定性的考验。该系统能够对大型网络设备进行有效的管理和处理事件。

## **3.4系统环境需求**

与网络设备的交互主要基于SNMP协议，采用B/S架构，实施方案所需条件如下：

部署：LAMP(Linux+Apache+Mysql+PHP),GCC

# 第4章：系统设计方案

## **4.1系统总体设计**

### 4.1.1概述

1. **免费Arp**

目前，免费arp的作用有两种。第一种是宣告广播的作用，以告诉整个广播域，目前这个IP所对应的MAC地址是什么。第二种是看看广播域内有没有别的主机使用自己的IP，如果使用了，则可以发现“IP冲突”。普通arp请求报文广播发送出去，广播域内所有主机都接收到，计算机系统判断arp请求报文中的目的IP地址字段，如果发现和本机的IP地址相同，则将自己的MAC地址填写到该报文的目的MAC地址字段，并将该报文发回给源主机。所以只要发送arp请求的主机接收到报文，则证明广播域内有别的主机使用和自己相同的IP地址。免费arp的报文发出去是不希望收到回应的，只希望是起宣告作用；如果收到回应，则证明对方也使用自己目前使用的IP地址。在所有网络设备（包括计算机网卡）up的时候，都会发送这样的免费arp广播，以宣告并确认有没有冲突。

最理想的状态，就是当设备发送免费arp的时候，在交换机上获取到设备的IP和MAC，经过分析和比对，判断该设备是否为新接入的设备。

1. **实现的基本原理**

核心思路是控制网络中的交换设备（Switch），采用定时轮询的方式，查询各个交换机上的arp表项，与基准校验池维护的合法设备的IP和MAC实时对比，快速发现交换机各个端口出现的不在基准校验池中的设备，再进一步分析，该设备是否合法。如果属合法设备，则将该设备加入基准校验池，并生成一条事件信息，在界面显示的同时记录日志（2015-xx-xx，有一个新的网元接入网络，IP地址为：xxx.xxx.xxx.xxx，MAC地址为：xx-xx-xx-xx-xx-xx）；如果属非法设备，系统应该产生告警，同时通知管理员进行人工处理。

系统核心功能为发现新接入的设备，在此基础上可以进行扩展，例如：

(1)如果在检查中发现交换机某个Eth端口对应多个不同的MAC，可推断该端口私接了HUB；

(2)在于基准校验池的对比过程中，发现某设备的MAC地址变化，可进一步判断是否有IP地址盗用行为（ARP欺骗）；

(3)如果同一个MAC地址同时出现在不同的交换机的非级联端口上，可能意味着IP-MAC被成对盗用。

系统原理图如图4-1所示：

对返回的数据进行检验

通过 Server Databases

SNMP协议 回应请求

访问交换机 发回ARP表项信息



Switch

**图4-1 系统功能原理图**

### 4.1.2架构设计

本系统的整体框架采用MVC框架，MVC是软件工程中的一种软件架构模式，是当前流行的WEB应用设计框架的事实标准。它把软件系统分为三个基本部分：模型（Model）、视图（View）、控制器（Control）。目的是实现一种动态和可持续的程序设计，使后续对程序的修改和扩展简化，并且使程序某一部分的代码或功能能重复利用成为可能[[[14]](#footnote-13)]。

本系统的核心算法采用的是基于SNMP协议的管理者和代理模式的简单集中式网络管理。整个网络性能管理系统架构于SNMP模式的管理者和代理者结构之上。系统和SNMP管理进程的交互是通过开源工具NET-SNMP提供的相关接口函数实现的。具体的架构设计图如图4-2所示：

用户浏览器

（Web Browser）

代理进程

代理进程MIB

管理进程

管理进程MIB

MySql数据库

网元接入检测系统

（Web Server）

HTTP协议

数据交互 接口调用

SNMP协议

**图4-2 系统架构设计图**

## **4.2模块设计**

系统在模块上划分为三个模块，分别是“数据采集模块”、“数据处理模块”、“信息展示模块”。

（1）数据采集模块通过被管网元对象接口采集数据，按照一定数据准则传输至数据处理模块对数据进行处理，采集功能主要基于国际通用标准SNMP协议进行开发。

（2）数据处理模块对原始数据分析处理，将原始数据进行一系列提取、聚合、判断、最终形成有效的数据格式并存储至动态数据库，为信息展示模块做好数据准备。

（3）信息展示模块主要是结合智能监管平台的拓扑图页面，对于在智能监管平台范围内的设备，让用户直观的看到设备加入网络的过程；对于在智能监管平台范围外的设备，发现并及时展现告警信息，为用户提供方便快速直观的展现形式。

系统的模块设计图如图4-3所示：

**后台服务模块**

**前端展示**

**模块**

**数 据 采 集 模 块**

基于SNMP协议进行访问

存储数据

获取数据

访问物理设备

**数 据 处 理 模 块** 合法

非法

生成合法事件列表

数据库

对比数据

标准池

发出警告

**信 息 展 示 模 块** 合法 非法

检测

弹出警告窗口并通知管理员

将合法事件列表展示在WEB界面上

**图4-3 系统模块设计图**

### 4.2.1数据采集模块

数据采集模块位于网元接入检测系统的后台服务模块，负责和SNMP代理模块进行信息交互，并提供接口供后台核心模块调用。

1. **基于SNMP协议的数据采集技术**

管理站从被管理设备中采集信息一般有两种方法：主动对管理设备进行访问和接受管理对象的告警信息。主动访问管理者对对象是通过向SNMP代理发送get的报文请求从而收到代理的回复信息，将这些信息进行分析过滤后存入本地的数据库中；被动接收管理设备的告警信息是通过监听陷阱端口（通常为UDP162端口）来得到来自SNMP代理的Trap报文从而采集到设备的信息。流程设计如图4-4所示：

数据采集

等待后台核心调用

N Y

被调用

确定搜索设备

和SNMP代理进行交互

调用后台核心模块的数据处理接口

**图4-4 数据采集流程设计图**

1. **数据采集范围**

该系统的数据采集范围是某一个局域网中的所有交换机（Switch）设备。在本试验中应该与上层系统智能监管平台的管理实例保持一致。通过拓扑发现该局域网的所有符合要求的设备，然后基于SNMP协议请求这些设备上的数据信息。因为该系统的核心算法是基于在知道了一定的交换机设备数量上进行的数据采集，所有这个系统可以用在只要有交换机存在的场合。

1. **数据采集时间间隔**

为保障采集到的数据的可靠性和及时性，数据采集时间间隔应该小于ARP表项的老化时间（超时时间），因此，针对每台交换机，系统应在数据库中维护该设备ARP表项的老化时间，并设定一个定时查询时间，及时更新每台交换机的ARP表项的老化时间（正常情况下这个动作不必过于频繁，一般不会有人经常改ARP表项的老化时间，最短1小时查询一次，也可以一天查询一次，甚至一周查询一次）。

### 4.2.2数据处理模块

数据处理模块位于系统的后台服务中心，主要是对管理站从SNMP代理处得到的设备信息进行过滤和统计，处理完后存入数据中心。

当系统从交换机上得到ARP表项的的内容后，将里面和接入设备相关的数据过滤出来，首先与基准校验池（Basic-Check-Pool）的数据进行比较，看该设备的信息是否在BCP里面，如果在，则判断为合法，继续下一条信息的读取与识别；如果不在，在前往标准池（Topo-Host-Node）中进行查找，若能找到该设备的相关信息，则说明该设备合法，同时将该设备的信息加入到BCP中，并给出合法事件列表，如果不在THN中，在说明该设备不合法，进而启动系统的告警流程。

**1.数据的一致性处理**

第一，接入发现子系统的数据采集范围，应该与上层高系统智能监管平台监控的交换机保持一致，即表项同时创建同时删除。逻辑上是智能监管平台操作接入发现子系统的采集范围（Collect-Scope-Info）数据，因此对智能监管平台没有影响，只需在操作Collect-Scope-Info之前查询一下接入发现功能是否开启即可。

第二，接入发现子系统的基准校验池，应该与智能监管平台监控的设备实例保持一致。如果出现不一致的情况，可能是定时器未触发下一次采集。基准校验池只能小于或等于智能监管平台监控的设备实例范围，不存在大于的情况。

**2.数据分析**

（1）未知IP地址发现流程

设备已经在监控范围内，但是接入发现功能的定时器未触发下一次轮询，此时监控范围大于基准校验池。当定时器触发轮询时即可发现差额设备。系统在基准校验池中找不到对应的IP和MAC信息，则需要到Topo\_Host\_Node进一步查询。通过IP地址，到Topo\_Host\_Node表中（对应objip）查询id，再通过id到Topo\_Interface表中查询phys\_address，此时发现，这一对IP和MAC已经在智能监管平台的监控范围内，则系统自动将新的IP和MAC地址加入基准校验池，并生成一条事件信息（2015-xx-xx，发现一组新的设备地址，IP地址为：xxx.xxx.xxx.xxx，MAC地址为：xx-xx-xx-xx-xx-xx），在界面进行显示，并记录日志。

设备不在监控范围内，接入发现子系统先采集到设备的IP和MAC地址存在时，先判断未知设备在网络中的位置，在系统界面闪烁显示告警：“发现未知的IP地址”。具体的流程如图4-5所示：



**图4-5 数据处理流程图**

**（2）告警消除流程**

当发现未知IP和MAC地址时，系统产生告警，告警消除流程需要通知用户处理才能达到良好的实时性（因为系统自动发现设备离开网络与设备真实撤离的时间点有一定的时延，至少要延迟到ARP老化超时之后，这样给用户的感受会认为系统的实时性不好）。具体分为以下两种情况进行分析。

①用户清除设备后同时在系统中处理告警

用户处理完告警后，系统下次轮训是很可能还会采集到该未知的IP和MAC地址，因为arp的老化时间还没有过，交换机上尚未清除该arp表项，此时系统不应该再次告警。处理方式是，在产生告警的时候，在数据库中缓存该arp表项的数据alarm-info，并且把老化时间+1进行倒数计时，在清除告警的时候，数据库中该条记录设置为已经清除的告警，下次轮训，如仍然发现这对IP和MAC既不在Basic-Check-Pool也不在Tope-Host-Node，此时不直接告警，而是到alarm-info中查这对IP和MAC是不是已经清除的告警，如果是，不处理，如果不是，产生告警。标记为已经清除的告警，在老化时间倒数计时为0时，从alarm-info中删掉该条数据。

②用户清除设备后未在系统中处理告警

这种情况可能是内部用户非法接入了一小段时间，随后立即撤离了。这个告警会在撤离后延迟N分钟（arp老化）自动清除，但是系统需要在界面显示曾经的接入信息并且记录日志，便于事后的追查和处理。

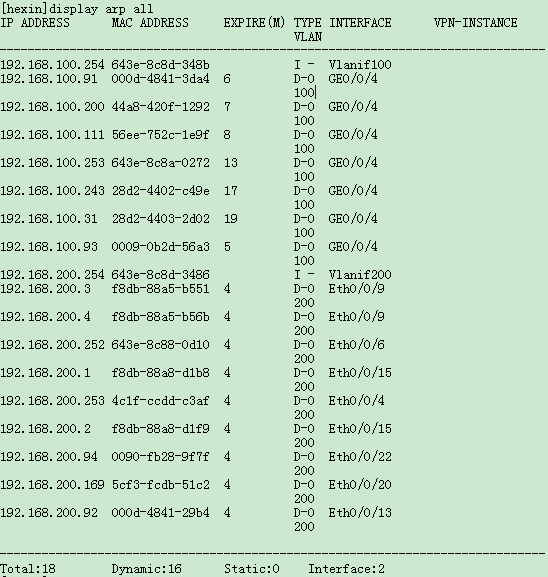
### 4.2.3数据存储模块

1. **基准校验池的创建**

接入发现功能初次启用时，建立基准校验池Basic-Check-Pool。步骤如下：

查询智能监管平台数据库中topo\_host\_node表，找到所有objname为Switch的设备以及该设备对应的IP；

通过SNMP协议到各个Switch上采集arp表项信息，得到与命令行display arp all一致的数据信息，如图4-6所示：



**图4-6 ARP表项信息**

表4-1对上图的项目的具体类容进行了描述：

**表4-1 上图具体内容解释**

|  |  |
| --- | --- |
| 项目 | 描述 |
| IP ADDRESS | IP地址 |
| MAC ADDRESS | MAC地址 |
| EXPIRE（M） | 剩余存活时间（分钟） |
| TYPE | 映射类型（D-Dynamic，S-Static，I-Interface） |
| VLAN | VLAN ID |
| INTERFACE | 接口名称 |
| VPN-INSTANCE | VPN实例的名称 |

用TYPE或者INTERFACE过滤数据，仅分析TYPE为D和S，或者INTERFACE为GE端口和Eth端口的数据；

针对每条数据，首先通过IP地址，到topo\_host\_node表中（对应objip）查询id，再通过id到topo\_interface表中查询phys\_address，这里可能出现同一个IP地址对应多个物理地址的情况，只要能找到对应的数据，即认为ARP表项这条数据合理，如图4-7所示：

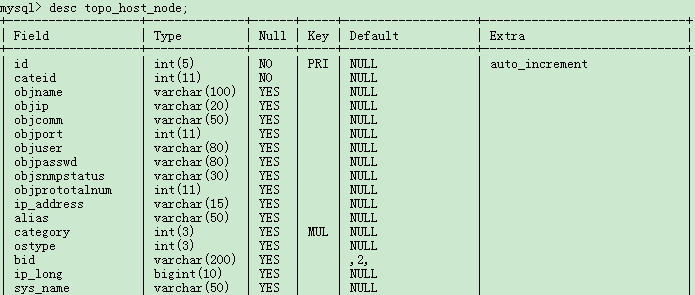
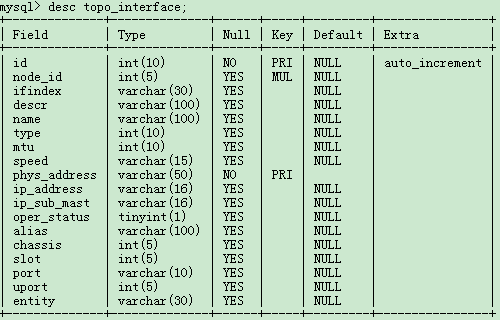


图 4-7（A） topo-host-node图



**图 4-7（B） topo-interface图**

将确认合理的数据加入基准校验池Basic-Check-Pool，Basic-Check-Pool维护的数据至少应该包括ID、IP ADDRESS、MAC ADDRESS、TYPE、VLAN、INTERFACE。

**（2）基准校验池数据的删除**

为保证智能监管平台监控范围与基准校验池Basic-Check-Pool数据的一致性，智能监管平台操作数据库删除监控实例时，应同时删除Basic-Check-Pool的对应数据。（Basic-Check-Pool拟与topo\_host\_node采用相同的键值，便于查找）。

### 4.2.4信息显示模块

本系统采用的是web界面显示，共分为三个板块，一个是基准校验池的信息展示，一个是标准池的信息展示，一个是当前交换机上的设备的连接状况展示。当系统开启监控后，查询的信息便展现在当前的交换机页面上，一旦出现新加入的设备，界面将会有一台设备的动态加入显示过程，在系统进行设备合法性判断后，如果合法就显示一条合法设备加入的信息列表，如果非法将弹出提示框给管理员，展示具体的非法信息。

## **4.3系统约束要求**

（1）代码处理流程上，允许智能监管平台操作接入发现子系统数据库表结构的数据，相反的，不允许接入发现子系统操作智能监管平台的数据；

（2）信息的采集过程不会影响到被管理的网元对象。

## **4.4数据库设计**

### 4.4.1数据库表的设计

本系统用的是MYSQL数据库，该数据库建立在本地服务器上，具体表的设计如下：

Alarm-Info: 该表是存放关于设备消警信息的凭证，详见表4-2；

Basic\_Check\_Pool：基准校验池，存放当前子系统合法设备信息记录，详见表4-3；

Collect-Scope-Info：存放交换机老化时间信息的，便于时间触发器的启动，详见表4-4；

topo\_host\_node：标准池，存放整个系统合法设备信息记录，详见表4-5；

**表4-2 Alarm-Info**

|  |  |  |
| --- | --- | --- |
| 字段 | **类型** | **默认** |
| id | int(5) | NULL |
| ArpDynIpAddr | varchar(12) | NULL |
| ArpDynMacAddr | varchar(64) |  |
| Status | int(12) |  |

**表4-3 Basic\_Check\_Pool**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** |
| SwitchId | int(11) | 是 | NULL |
| ArpDynIfIndex | int(11) | 是 | NULL |
| ArpDynIpAdd | varchar(64) | 是 | NULL |
| ArpDynVrf | int(11) | 是 | NULL |
| ArpDynMacAdd | varchar(64) | 是 | NULL |
| ArpDynVlanId | int(11) | 是 | NULL |
| ArpDynOutIfIndex | int(11) | 是 | NULL |
| ArpDynExpireTime | int(11) | 否 | 0 |

**表4-4 Collect-Scope-Info**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** |
| SwitchId | int(11) | 是 | NULL |
| IpAddr | varchar(64) | 是 | NULL |
| Expire-Fake-Time | int(11) | 否 | 0 |

**表4-5 topo\_host\_node**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** |
| SwitchId | int(11) | 是 | NULL |
| ArpDynIfIndex | int(11) | 是 | NULL |
| ArpDynIpAdd | varchar(64) | 是 | NULL |
| ArpDynVrf | int(11) | 是 | NULL |
| ArpDynMacAdd | varchar(64) | 是 | NULL |
| ArpDynVlanId | int(11) | 是 | NULL |
| ArpDynOutIfIndex | int(11) | 是 | NULL |
| ArpDynExpireTime | int(11) | 是 | NULL |

### 4.4.2数据库的E-R图

如图4-8所示：

ArpDynIfIndex

SwitchId

ID

ArpDynVrf

ArpDynIpAddr

Basic-Check-Pool

ArpDynOutIfIndex

ArpDynMacAddr

ArpDynExpireTime

ArpDynVlanId

No intersection

ArpDynMacAddr

include

Expire-Fake-Time

ID

Collect-Scope-Info

Alarm-Info

IpAddre

SwitchId

include

ArpDynIpAddr

**图 4-8 系统E-R图**

### 4.4.3数据库的关系图

数据库的关系图如图4-9所示：

Topo-Host-Pool

PK ID

SwitchId

ArpDynIfIndex

ArpDynIpAddr

ArpDynVrf

ArpDynMacAddr

ArpDynVlanId

ArpDynOutIfIndex

ArpDynExpireTime

Basic-Check-Pool

PK ID

SwitchId

ArpDynIfIndex

ArpDynIpAddr

ArpDynVrf

ArpDynMacAddr

ArpDynVlanId

ArpDynOutIfIndex

ArpDynExpireTime

Alarm-Info

PK ID

ArpDynIpAddr

ArpDynMacAddr

Status

Collect-Scope-Info

PK SwitchId

IpAddre

Expire-Fake-Time

**图4-9 数据库关系图**

①Basic-Check-Pool:基准校验池。是snmp管理站从代理商采集到数据后在基准校验池进行 对比， 判断数据是否合法。

②Topo-Host-Pool:标准池。标准池是上一层系统智能监管平台下所管理的合法设备，是该系 统 进行检测设备的权威标准。当数据处理模块在基准校验池中找不到设备信息的时候， 就 进一步到标准池中进行查找，若能找到就说明该设备合法，同时增加这一条事件信 息到 基准校验池中；若是找不到，则说明该设备非法，启动报警程序。。

③Collect-Scope-Info:数据采集范围。这是记录该系统能到采集数据的交换机的信息。该信息 以 智能监管平台上的采集范围为标准

④Alarm-Info:消除报警信息统计。该表主要是记录某一个报警信息的设备时候在进行处理后 对起报警信息进行了消除。即系统检测到了非法设备后，应该首先来该数据池中进行查 找看是否有改设备的记录，并检查其消警状态，从而再判断是否对该设备发出警告信息。 保障整个网络的安全。

对数据库中的部分数据段进行解释，如表4-6所示：

**表4-6 数据库中字段解释说明表**

|  |  |  |
| --- | --- | --- |
| 字段 | **类型** | **说明** |
| SwitchId | int(11) | 交换机的id |
| ArpDynIfIndex | int(11) | 生成动态ARP表项的三层接口索引。值0是非法的。 |
| ArpDynIpAdd | varchar(64) | 动态ARP表项的IP地址。 |
| ArpDynVrf | int(11) | 动态ARP表项的VPN名称。 |
| ArpDynMacAdd | varchar(64) | 动态ARP表项的MAC地址。 |
| ArpDynVlanId | int(11) | 动态ARP表项的VLAN。 |
| ArpDynOutIfIndex | int(11) | 动态ARP表项的出接口的接口索引。 |
| ArpDynExpireTime | int(11) | 动态ARP表项的老化时间。 |
| Expire-Fake-Time | int(11) | 采集数据时间间隔 |
| Status | int(2) | 标记该ip对的警告是否消除 |

## **4.5异常容错处理模块**

### 4.5.1设备故障处理

设备类型为交换机时，可能导致采集不到数据，不处理；其他设备故障时，对接入发现子系统无影响。

### 4.5.2设备启用处理

当设备弃用时，用户应该从智能监管平台手动删除设备，智能监管平台删除自己监控实例的时候需要同时删除Basic-Check-Pool中的数据项（根据ID）；当设备类型为交换机时，还要删除Collect-Scope-Info中的数据项。

若设备弃用时，用户未从智能监管平台删除设备，接入发现子系统应该采集不到该设备相关的数据，不需要做特殊处理。

### 4.5.3用户修改设备ip地址处理

当用户修改IP地址时，应该从智能监管平台手动修改该设备对应的IP，智能监管平台修改自己监控实例的时候应该同时修改Basic-Check-Pool中的数据项（根据ID）；当设备类型为交换机时，还要修改Collect-Scope-Info中的数据项。

当用户修改IP地址，但为从智能监管平台手动修改该设备对应的IP时，智能监管平台仍然用原来的IP对该对象进行监控，应该采不到数据产生告警，接入发现子系统采集到新的IP地址，可认为这是新加入网络的网元对象，在界面告警。

### 4.5.4 ip与mac不匹配处理

在数据分析的过程中，如果发现对于同一个IP后面采集到的数据的MAC地址不同于基准校验池中已经存储的MAC地址，此时可能的情况是：

（1）双机热备的交换机发生了主备倒换；

（2）IP地址冲突；

（3）原来配置给A设备（网卡）的 IP地址，A设备弃用后，并未在智能监管平台将设备删除，又把该IP地址配置在了另外的设备上接入网络；

对于上述三种情况（实际情况可能更加复杂），从代码逻辑角度难以正确区分IP与MAC不匹配的根本原因，所以不做进一步分析，直接生成告警。

# 第5章：系统的实现与测试

## **5.1系统开发环境**

### 5.1.1开发平台以及开发工具

操作系统：Linux Ubuntu15.04

数据库：MySQL5.6.24

C程序开发：VIM+GCC

SNMP开发：Net-Snmp5.7.3 shell

PHP开发：LAMP（linux+Apache+Mysql+PHP） myadminPHP

WEB开发：HTML，Jquery，CSS，Javascript,浏览器chrome

### 5.1.2开发环境搭建

1. 首先在Linux环境下安装SNMP的环境，从http://www.net-snmp.org/网站上下载net-snmp源程序包进行编译安装，配置相关的参数和环境变量；
2. 安装LAMP，包括Apache，Mysql，PHP的相关软件的安装以及参数的配置。
3. 安装myadminPHP，并且和本地的Mysql进行绑定。

## **5.2系统实现关键技术**

因为源代码中没有将该功能的代码进行公开，所以我们在使用的时候需要自己实现该部分的功能。在后面的工作中我们需要频繁用到SNMPBULKWALK功能向代理方获取我们需要的设备信息，所以我们决定将该功能封装成一个完备的类似于SNMP工具中提供的方法，以便后面的使用。

Snmpbulkwalk是利用GetBulkRequest实现对给定管理树进行遍历的工具，对表格类型的管理信息进行遍历读取。它的速度快于snmpwalk。它在snmp工具中的用法是：

Snmpbulkwalk -v SNMP的版本号（1,2,2c） 交换机，路由器或者主机的IP地址 -c snmp的口令 对象标识符OID

例如： ①snmpbulkwalk -v 2c 192.168.6.53 -c public .1.3.6.1.2.1.1

②snmpbulkwalk -v 2c 192.168.6.53 -c public .1.3.6.1.2.1.2

③snmpbulkwalk -v 2c 192.168.6.53 -c public .1.3.6.1.2.1.3

①获取mib-2的system的数据，即.1.3.6.1.2.1.1(iso->identified organization->dod->internet->mgmt2->mib-2->system)

②获取mib-2的interface的数据,即.1.3.6.1.2.1.2(iso->identified organization->dod->internet->mgmt2->mib-2->interface)

③获取mib-2的address translation的数据，即.1.3.6.1.2.1.3(iso->identified organization->dod->internet->mgmt2->mib-2->address translation)

大部分的交换机上都已经部署了SNMP的代理端，所以我们只需要实现相应的管理方端的代码即可。管理方在向代理发送了请求报文后，便开启接收代理方报文的功能。所以管理方同时扮演着发送方和接收方的角色。在实现的过程中我们首先了解管理方的SNMP消息处理的典型流程：

【发送方】

i.使用SMI/MIB/ASN.1构造PDU

ii.将这个ASN.1的PDU结构和发送方的地址、端口、和目的方地址、端口一起发送到鉴 别服务，鉴别服务对这些数据进行符合要求的变化，如加密或者增加鉴别码。

iii.将上一步生成的数据打包进community,version,生成完整的SNMP消息报文

iv.对SNMP报文进行BER编码，然后发送到【传输层】向接收方发送。

【接收方】

i.对消息进行基本的语法检查，不能分析的则删除该消息。

ii.检查版本号，删除版本号不匹配的信息。

iii.将PDU、发送方地址、端口、接收方地址、端口发送到鉴别服务：

1. 如果鉴别失败，鉴别服务发出信号通知协议实体，产生一个Trap并删除消息。
2. 如果鉴别成功，鉴别服务返回PDU。

iv.对PDU进行基本的语法检查，如果分析失败则删除该PDU，否则选择合适的SNMP 访问策略，并对PDU进行相应处理。

接下来用代码实现功能：

1. **基本步骤：**

1，init\_snmp初始化snmp  
 2，snmp\_parse\_oid根据mib分析parse oid循环执行3-7，直到所有pdu都发出去  
 3，snmp\_sess\_init初始化session，然后用一些赋值语句设置session  
 4，设置session中的回调函数(callback)，这个函数负责处理返回的信息  
 5，snmp\_open打开session  
 6，snmp\_pdu\_create创建pdu  
 7，snmp\_send发送包，循环执行8-10，直到没有active hosts  
 8，snmp\_select\_info设置多路复用的几个信息  
 9，select多路复用  
 10，snmp\_read接受所有返回信息，具体处理交给callback  
 11，回调函数(callback)，具体处理返回信息，参数是固定的，参数是：  
 int operation, struct snmp\_session \*sp, int reqid,struct snmp\_pdu \*pdu, void \*magic  
 根据这些可以处理返回信

流程图如图5-1所示：

初始化SNMP

根据命名树分析对象的标识符

初始化并设置session

设置session中的回调函数

打开session

使用SMI/MIB/ASN.1构造PDU

发送包

是否所有包都已经发出去

否

是

设置多路复用信息

Select多路复用

接收方接收信息，交给回调函数处理

是否有active hosts

是

否

执行回调函数

**图 5-1 报文发送流程图**

1. **相应结构体的构造：**

my\_snmp\_val是mib树上的结点信息,它里面放了一个snmp变量的先关信息，my\_oid\_result是根据对象相应的OID从mib树上遍历得到的包含结点信息的对象的结果，该结构体增加了对象的名字，名字长度以及类型，类型包括等相关信息。具体的构造如下：

typedef struct my\_oid\_result {

unsigned long oid\_name[128];

size\_t oid\_name\_length;

size\_t objid\_length;

u\_char type;

size\_t val\_len;

my\_snmp\_val val;

}my\_oid\_result;

1. 方法的具体实现：

具体有3个函数：

①void get\_bulkwalk\_oid\_values(netsnmp\_variable\_list \*vars, my\_oid\_result \*oid\_result);

②void my\_snmp\_bulkwalk\_perror(unsigned int ratval);

③int my\_snmp\_bulkwalk(const char \*peername, const char \*comity, const char \*oid\_argv, my\_oid\_result oid\_results[], unsigned int \*oid\_results\_nums);

第一个函数的主要功能是得到mib树上的结点信息，netsnmp\_variable\_list \*vars是返回的PDU数据类型，在该函数中对vars进行解析，并将相应的参数赋值到my\_oid\_result结构体上，snmp\_variable\_list是variable\_list的typedef。具体结构的解析如下：  
a. snmp\_variable\_list相当于是snmp\_variable\_list类型链表的某一个元素，而不是整个链表。它里面放了一个snmp变量的相关信息。  
b. snmp\_variable\_list里的next\_variable是指向这个链表的下一个snmp\_variable\_list元素的指针。  
c. name是这个元素对应的oid的指针。name指向的是一个int类型的数组，这个数组里面放的就是这个元素的oid  
d. name\_length是name指的那个装有oid的int型数组的长度。即这个元素的oid的长度  
e. type是这个元素的类型。常用的有：ASN\_OBJECT\_ID，ASN\_INTEGER，ASN\_OCTET\_STR，ASN\_FLOAT，ASN\_INTEGER64，ASN\_TIMETICKS，ASN\_IPADDRESS，ASN\_COUNTER，ASN\_GAUGE等等。  
f. val是一个netsnmp\_vardata类型的变量[[[15]](#footnote-14)]。

[netsnmp\_vardata里面g.integer是指向ASN\_INTEGER型结果的指针，string是指向ASN\_OCTET\_STR类型结果的指针，](http://www.net-snmp.org/dev/agent/unionnetsnmp__vardata.html。netsnmp_vardata里面g.integer是指向ASN_INTEGER型结果的指针，string是指向ASN_OCTET_STR类型结果的指针，h.)

h. objid是指向ASN\_OBJECT\_ID类型的指针等等。  
I. val\_len是存放这个元素的值的内存的大小。如果这个值是一个ASN\_OCTET\_STR的元素，那么这个元素的长度就是val\_len/sizeof(char)，同样的，如果这个值是一个oid，那么存放这个oid的int类型数组的长度就是val\_len/sizeof(int)  
j. name\_loc是一个oid类型的数组，这个数组存放的其实就是这个元素的oid。  
name\_loc和name前者是一个指向这个元素oid的指针，后者是存放这个元素oid的数组。这两者都可以取到这个元素的oid  
k. buf是个u\_char类型的数组。如果这个元素的值的在内存中的长度小于40（有百分之九十的可能性是这样），那么buf里面就放的这个元素的值。如果这个元素的值在内存中的长度大于40，buf里面存放的就是乱码。对于ASN\_TIMETICKS类型的值来说，buf里面放的是一个int类型的指针，指针的值是这个元素的值的地址，\*(vars->buf)这样可以取出这个值。对于ASN\_OCTET\_STR类型的值来说，如果长度小于40，那么buf里买面就放的这个字符串。对于ASN\_OBJECT\_ID类型的值来说，buf里面装的是oid的值，只不过是这也样子的：.1.3.6.1.6.3.16.2.2.1在buf里面的值是.1.0.0.0.3.0.0.0.6.0.0.0.1.0.0.0.6.0.0.0.3.0.0.0.16.0.0.0.2.0.0.0.2.0.0.0.1.0.0.0

第二个函数主要是对my\_snmp\_bulkwalk的出错信息进行处理，根据出错信息的类型不同将他们分为5种情况，根据返回的出错代码不同来进行相应的出错处理，这里即是将出错信息打印出来。具体的实现如下：

case 1:printf("[ERROR] my\_snmp\_walk: the argument value oid\_names is empty\n");

case 2:printf("[ERROR] my\_snmp\_walk: the snmp\_open function run failed\n");

case 3:printf("[ERROR] my\_snmp\_walk: the snmp\_parse\_oid function run failed\n");

case 4:printf("[ERROR] my\_snmp\_walk: time out\n");

case 5:printf("[ERROR] my\_snmp\_walk: snmpget sesson\_handls error\n");

case 6:printf("[ERROR] my\_snmp\_walk: end of MIB\n");

case 7:printf("[ERROR] my\_snmp\_walk: response->errstat others error\n");

default :printf("[ERROR] my\_snmp\_walk: no %d error type return value\n", ratval);

第三个函数即是实现SNMPBULKWALK的具体功能：

int my\_snmp\_bulkwalk(const char \*peername, const char \*comity, const char \*oid\_argv, my\_oid\_result oid\_results[], unsigned int \*oid\_results\_nums);

Peername:需要得到信息的设备的ip地址

Comity:snmp的共同体

Oid\_argv:oid的指针

oid\_results[]:得到的对象的数组

oid\_results\_nums:得到的对象的数目

首先进行相应的初始化：

netsnmp\_session session,\*ss;

netsnmp\_pdu \*pdu, \*response;

netsnmp\_variable\_list \*vars;

oid name[MAX\_OID\_LEN];

size\_t name\_length;

oid root[MAX\_OID\_LEN];

size\_t rootlen;

netsnmp\_ds\_register\_config(ASN\_BOOLEAN, "snmpwalk", "includeRequested", NETSNMP\_DS\_APPLICATION\_ID, NETSNMP\_DS\_WALK\_INCLUDE\_REQUESTED);

init\_snmp("snmpapp");

snmp\_sess\_init(&session);

session.version = SNMP\_VERSION\_2c;

session.community = (char\*)community;

session.community\_len = strlen(session.community);

session.peername = (char\*)peername;

session.timeout = 2000000;

session.retries = 2;

init\_snmp("snmpapp");

snmp\_sess\_init(&session);

session.version = SNMP\_VERSION\_2c;

session.community = (char\*)community;

session.community\_len = strlen(session.community);

session.peername = (char\*)peername;

session.timeout = 2000000;

其次根据mib 对oid进行相应的解析：

if(snmp\_parse\_oid(oid\_argv, root, &rootlen) == NULL) {

printf("[ERROR] my\_snmp\_bulkwalk: call snmp\_parse\_oid function failed\n");

snmp\_close(ss);

SOCK\_CLEANUP;

return 2;

}

打开session，创建相应的PDU数据包，并发送到代理的端口，同时将接收模式设置为异步处理。

while(running) {

pdu = snmp\_pdu\_create(SNMP\_MSG\_GETBULK);

pdu->non\_repeaters = non\_reps;

pdu->max\_repetitions = reps;

snmp\_add\_null\_var(pdu, name, name\_length);

status = snmp\_synch\_response(ss, pdu, &response);

..........

根据不同的情况vars的type不同的情况，进行相应的my\_oid\_result的构造

.......

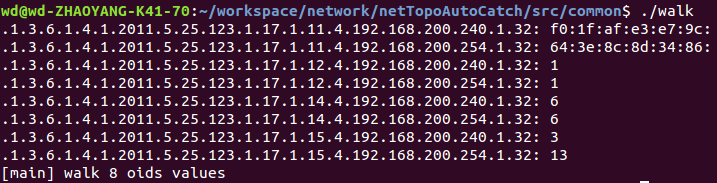
\*oid\_results\_nums = numprinted;

snmp\_close(ss);

SOCK\_CLEANUP;

}

实验截图如图5-2所示：



**图5-2 实验截图**

## **5.3系统模块实现**

### 5.3.1数据采集模块

1. **基于SNMP协议的数据采集技术**

数据采集模块负责响应客户端的要求，通过SNMP代理采集被管设备的信息。需要采集的数据包括交换机上连接的设备的信息，维护交换机ARP表项信息。这两种采集方式都采用主动采集，利用之前实现的my\_bulk\_walk方法对交换机设备进行访问。他们的具体描述如下：

方法gethwArpDynTable是通过snmp协议对交换机进行访问，它通过报文GetRequest请求并返回相应消息的接口。框架图如图5-3所示：

**gethwArpDynTable**

-Count:unsigned int

-ip\_count:unsigned int

-oid\_results\_nums : unsigned int

-ip\_arpdyn\_table\_nums : unsigned int

-tmp[8]: unsigned char

-str[32]: unsigned char

-mtmp[12]: unsigned char

-mstr[40]: unsigned char

-\*ip\_arpdyn\_table\_rows :0

-\*oid\_results : my\_oid\_result

+get\_ArpDyn\_table\_info\_from\_device\_by\_snmp(const unsigned char \*ip, const unsigned char \*comm,IP\_ArpDyn\_TABLE\_INFO\*ip\_arpdyn\_table\_info,unsigned int \*ip\_arpdyn\_table\_rows):int

**my\_snmp\_bulkwalk**

-oid\_name[128]:unsigned long

-oid\_name\_length:size\_t

-objid\_length: size\_t

-Type: u\_char

-val\_len: size\_t

-my\_snmp\_val val

-get\_bulkwalk\_oid\_values(netsnmp\_variable\_list\*vars, my\_oid\_result \*oid\_result):void

- my\_snmp\_bulkwalk\_perror(unsigned int ratval):void

+my\_snmp\_bulkwalk(const char \*peername, const char \*comity, const char \*oid\_argv, my\_oid\_result oid\_results[], unsigned int \*oid\_results\_nums);:int

**图 5-3 框架图**

从图中可以看出，gethwArpDynTable类对外提供一个接口，即get\_ArpDyn\_table\_info\_from\_device\_by\_snmp，访问该接口需要提供的信息如下：

const unsigned char \*ip：被访问的设备ip地址

const unsigned char \*comm：SNMP的共同体名

该接口是通过my\_snmp\_bulkwalk方法访问交换机，得到交换机的动态arp表项，里面包括的设备信息如下：

get the ArpDynIfIndex,ArpDynVrf,ip\_addr

get the ArpDynIfIndex,ArpDynVrf,ip\_addr

get the ArpDynIfIdex

get the ArpDynVrf

get the ip\_addr

get the MAC addr table entity ip \*

get the Arp\_Dyn\_VlanId

get the Arp\_Dyn\_OutIfIndex

get the Arp\_Dyn\_ExpireTime

get the ArpDynIfIdex

get the ArpDynVrf

get the ip\_addr

get the MAC addr table entity ip \*

get the Arp\_Dyn\_VlanId

get the Arp\_Dyn\_OutIfIndex

get the Arp\_Dyn\_ExpireTime

这里我们定义一个结构体来封装这些信息：

typedef struct IP\_ArpDyn\_TABLE\_INFO {

typedef struct IP\_ArpDyn\_TABLE\_INFO {

unsigned int SwitchId;

unsigned long ArpDynIfIndex;

unsigned char ArpDynIpAdd[16];

unsigned char ArpDynMacAdd[20];

unsigned long ArpDynVrf;

unsigned long ArpDynVlanId;

unsigned long ArpDynOutIfIndex;

unsigned long ArpDynExpireTime;

}IP\_ArpDyn\_TABLE\_INFO;

unsigned int SwitchId;

unsigned long ArpDynIfIndex;

unsigned char ArpDynIpAdd[16];

unsigned char ArpDynMacAdd[20];

unsigned long ArpDynVrf;

unsigned long ArpDynVlanId;

unsigned long ArpDynOutIfIndex;

unsigned long ArpDynExpireTime;

}IP\_ArpDyn\_TABLE\_INFO;

1. **数据的采集时间间隔**

数据的采集时间应该小于交换机ARP表项的老化时间，通过snmp协议我们可以采集到交换机arp表项的老化时间，并将它的信息存入到数据库Collect-Scope-Info表项中，在对一个交换机进行采集数据之前，首先访问数据库得到该交换机的老化时间，用old\_time标记，然后用Linux系统自带的sleep()函数进行数据采集时间的控制。

Sleep()函数的原型是：

#include <unistd.h>  函数使用头文件

unsigned int sleep(unsigned int seconds);

参数为毫秒 （如果需要更精确可以用usleep，单位为微秒）

Sleep()函数的功能是执行挂起一段时间，若进程/线程挂起到参数所指定的时间则返回0，若有信号中断则返回剩余秒数。

用该函数我们就可以很方便的控制系统采集数据的时间间隔，我们将时间间隔设定为大于old\_time的秒数。启动数据采集器的时候便启动计时器，在模块是镶嵌在后台核心模块中。当管理员在交互界面开启该模块的功能时，系统便开始运转。实验截图5-4如下;

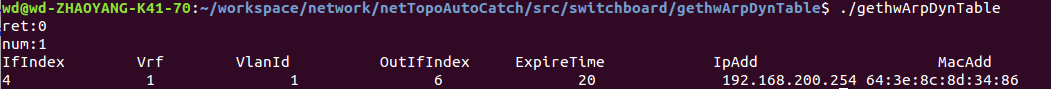


图5-4 实验截图

### 5.3.2数据处理模块

1. **对数据的合法性进行检测**

该模块的核心思想是判断采集到的数据是否和基准校验池以及标准池的数据相吻合，通过其是否吻合来判断数据的合法与否。检测数据合法模块的时序图5-5如下：

[data processor](http://www.baidu.com/link?url=1r6lQtAPOrVlQIhPpz0ssKM0K1lhZtKr5CNVsn8Y5YCvhVIqfefkpCXdiEFoMqCXwsNk4BdaVGX7NKxBT2zWFmNhYJZVYxT79bnZmepx11pYgdE30-9GYru8-4VhRub8" \t "https://www.baidu.com/_blank)

Topo\_Check\_Pool

Basic\_Check\_Pool

[data c](http://www.baidu.com/link?url=1r6lQtAPOrVlQIhPpz0ssKM0K1lhZtKr5CNVsn8Y5YCvhVIqfefkpCXdiEFoMqCXwsNk4BdaVGX7NKxBT2zWFmNhYJZVYxT79bnZmepx11pYgdE30-9GYru8-4VhRub8" \t "https://www.baidu.com/_blank)ollector

1.Get data

2. Return data

3.Flag Judge\_IpAddr\_in\_database\_Basic\_Check\_Pool();

4.Flag=1:legal

5.Falg=0:

Flag Judge\_IpAddr\_inTopo\_Pool()

6.

Falg=1:add in Basic\_Check\_Pool

7. Flag=1:legal

8. Flag=0:illegal

**图5-5 数据检测模块时序图**

在具体实现的时候，将部分代码模块功能相似的进行了封装，使代码具有很好的封装性和可扩展性，具体的代码模块结构图如图5-6所示

**ArpDynCatch**

-mysql:MYSQL

+i,j: Status

+illegal\_flag=1: Flag

+ rawtime: time\_t

+\*timeinfo: struct tm

+switch\_info[64]: SWITCH\_INFO

+switch\_info\_nums: int

+Judge\_IpAddr\_in\_database\_ArpDyn\_Table(const unsigned char \*switchip,unsigned char \*comm,

IP\_ArpDyn\_TABLE\_INFO \*src\_arpdyn\_table\_info,

ArpDynTable\_INFO \*des\_arpdyn\_table\_info):int

+Judge\_IpAddr\_is\_Legal(const unsigned char \*switchip,unsigned char \*comm,

IP\_ArpDyn\_TABLE\_INFO \*src\_arpdyn\_table\_info,

ArpDynTable\_INFO \*des\_arpdyn\_table\_info,

HOST\_INFO \*Resultdata,unsigned int \*relength);:HOST\_INFO\*

-Judge\_IpAddr\_from\_Src\_And\_Des(IP\_ArpDyn\_TABLE\_INFO \*src\_arpdyn\_table\_info,

unsigned int \*src\_num,

ArpDynTable\_INFO \*des\_arpdyn\_table\_info,unsigned int \*des\_num,

HOST\_INFO \*legal,HOST\_INFO \*illegal,

unsigned int \*legcount,unsigned int \*illcount):Flag

-Copy\_Hostinfo\_Arry(int index,int length,HOST\_INFO \*src,HOST\_INFO \*des):void

ArpDynCatch

**图 5-6 数据检测模块代码结构图**

中各个函数的功能分别如下：

Judge\_IpAddr\_in\_database\_ArpDyn\_Table：判断设备的ip信息是否存在基准校验池中，若存在，则flag=1,设备合法；若不存在，flag=0，设备需要进一步的判断；

Judge\_IpAddr\_is\_Legal：判断设备的ip信息是否在标准池中，若存在，设备合法，并将该设备的信息添加到基准校验池中，如不存在，则该设备非法，启动系统的警报系统；

Judge\_IpAddr\_from\_Src\_And\_Des：判断两个表的信息是否完全吻合；

Copy\_Hostinfo\_Arry：拷贝主机的部分信息到另外一个表中；

对于合法的设备，系统生成合法事件列表，对于非法的设备，系统对非法信息进行统计，分析，在这里我们进分析以下几种情况：

1. Ip地址合法，mac地址不合法：生成ip地址被盗的警告；
2. Ip地址不合法，mac地址合法：可能是用户修改了ip地址，这里当做是新加入的网元设备进行警告提示；
3. Ip地址和mac地址均不合法：则这是一个非法的设备，首先应该到数据库

Alarm-Info

中查找相关的设备信息，若能找到相关信息，则查看其消警选项，若该选项是告知已经消除该设备的警告消息的，则不予理会，若没有消除警告信息或者该消警库中没有该设备的信息，则向管理员发出警报，并标记该设备，等候处理。

### 5.3.3后台核心模块

后台开发是基于CodeIgniter的PHP框架。CodeIgniter 是一套给 PHP 网站开发者使用的应用程序开发框架和工具包。 它的目标是让你能够更快速的开发，它提供了日常任务中所需的大量类库， 以及简单的接口和逻辑结构。通过减少代码量。下图是展示了整个系统的数据流程，如图5-7所示：

Models

Application

Controller

Security

Routing

index.php

index

Libraries

Helpers s

View

Caching

Plugins

Scripts

**图 5-7 数据流程图**

①index.php 文件作为前端控制器，初始化运行 CodeIgniter 所需的基本资源；

②Router 检查 HTTP 请求，以确定如何处理该请求；

③如果存在缓存文件，将直接输出到浏览器，不用走下面正常的系统流程；

④在加载应用程序控制器之前，对 HTTP 请求以及任何用户提交的数据进行安全检查；

⑤控制器加载模型、核心类库、辅助函数以及其他所有处理请求所需的资源；

⑥最后一步，渲染视图并发送至浏览器，如果开启了缓存，视图被会先缓存起来用于 后续的请求。

CodeIgniter 的开发基于 MVC（模型-视图-控制器）设计模式。MVC 是一种 用于将应用程序的逻辑层和表现层分离出来的软件方法。在实践中，由于这种分离 所以你的页面中只包含很少的 PHP 脚本。

• 模型：代表你的数据结构。通常来说，模型类将包含帮助你对数据库进行增删改查的方法。

• 视图：是要展现给用户的信息。一个视图通常就是一个网页，但是在 CodeIgniter 中， 一个视图也可以是一部分页面（例如页头、页尾），它也可以是一个 RSS 页面， 或其他任何类型的页面。

• 控制器：是模型、视图以及其他任何处理 HTTP 请求所必须的资源之间的中介，并生成网页。

控制器的类图如下图5-8所示

class Network extends MY\_controller

public function Topo\_Host\_Node()；

public function Basic\_Check\_Pool()；

public function Switch\_Info($SwitchId)；

public function Checking($SwitchId)；

**图5-8 控制器类图**

上图的方法分别是标准池，基准校验池，设备管理，设备检测的控制模块。前三部分主要是和Model进行交互，获得数据库中的数据传到前台进行显示。设备检测这个模块主要是对网络的设备进行监控。当系统启动后，系统通过数据采集器第一次对网络中的设备进行查询，采集相关数据并初始化一些的信息。这个时候我们会有一个count来记录当前的设备数量，当进行轮询的时候，只需要先检查网络中的设备数目是否有变化，因为轮询的时间非常短，所以我们认为在这个时间段不存在一定数目的设备断开并同时连接另外的相同数目的新设备。所以每次轮询我们就会和当前的count进行对比，

$src="/home/wd/workspace/network/netTopoAutoCatch/src/switchboard/gethwArpDynTable/ArpDynCatch ".$ip;

$src="/home/wd/workspace/network/netTopoAutoCatch/src/switchboard/gethwArpDynTable/ArpDynCatch ".$ip;

while(0){

$a=exec($src,$output,$return\_val);

if(count($output)==$num){

sleep(20);

}

while(0){

$a=exec($src,$output,$return\_val);

if(count($output)==$num){

sleep(20);

}

如果数目没有变化，我们认为网络中的设备没有出现设备进出现象。如果数目存在变化，则我们调用数据处理模块来对相应的设备进行检测，从而判断设备的合法性，进而确定是否需要启动报警系统。这里需要注意的是浏览器默认的响应时间是30s，但是我们后台数据处理的时间可能会大于这个时间，所以我们用函数set\_time\_limit(0)来修改响应时间。

模型Model主要是数据的查询，具体的代码类图如图5-9所以：

class User\_model extends CI\_Model

public function getdata($sql)；

public function getalldata($sql)；

public function insert($table,$data)；

public function update($table,$data,$condition)；

public function delete($table,$condition)；

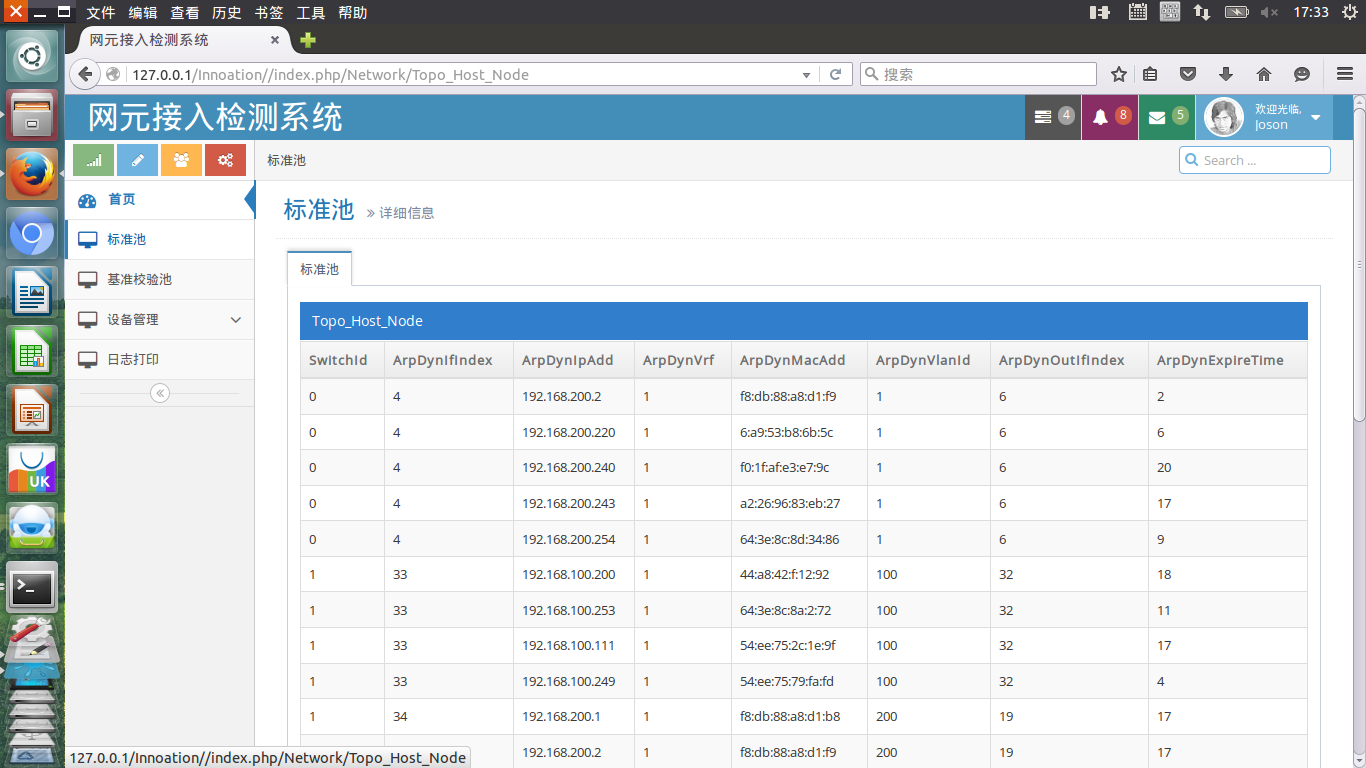
**图5-9 模型类图**

上图所示的model的功能分别是对数据库的查询数据、插入数据、更新数据、删除数据。在需要和数据库进行交互时，可以在控制器中直接调用模型。

相关的信息显示在视图中体现。

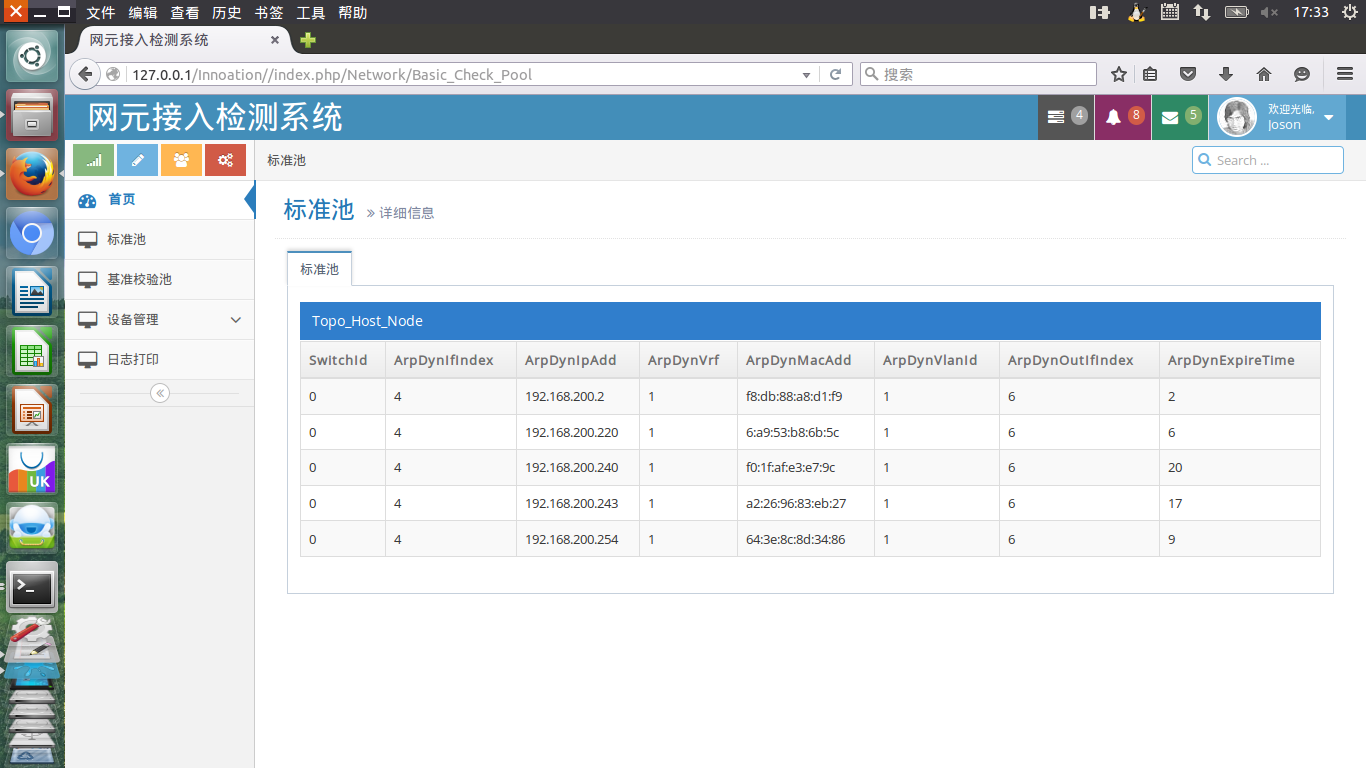
### 5.3.4信息展示模块

信息展示模块使基于web显示展现的，采用的是左边菜单导航右边展示窗口的模式进行设计的。左边的导航包括如下几个项目：基准校验池、标准池、设备管理、日志打印。（1）当点击标准池图标时，出现以下界面，该界面展现的是数据库中标准池的数据。主要便于管理员随时进行标准池的查看。实验截图如图5-10所示：



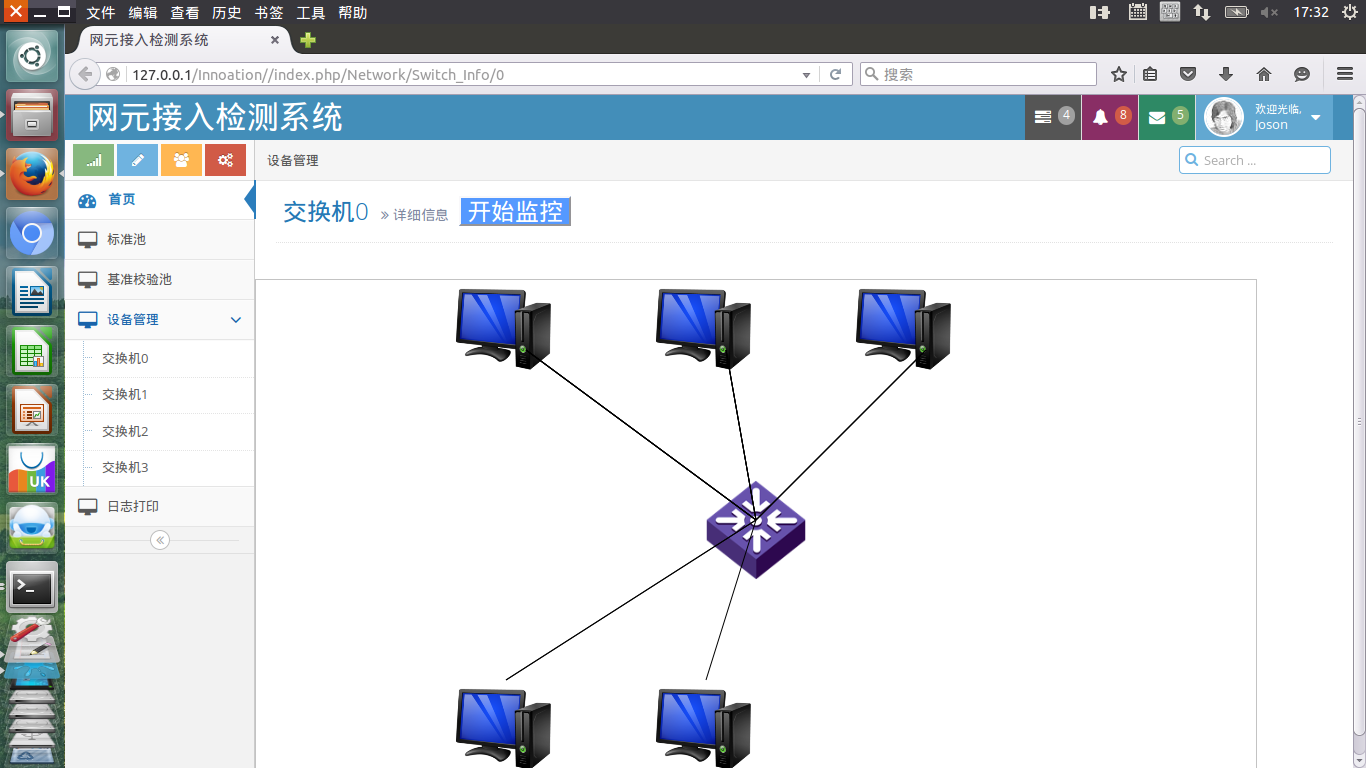
**图5-10 实验截图**

（2）第二部分是基准校验池的数据展示，原理和标准池是一样的，均是从后台的数据库中传过来的数据进行展示，实验截图如图5-11所示：



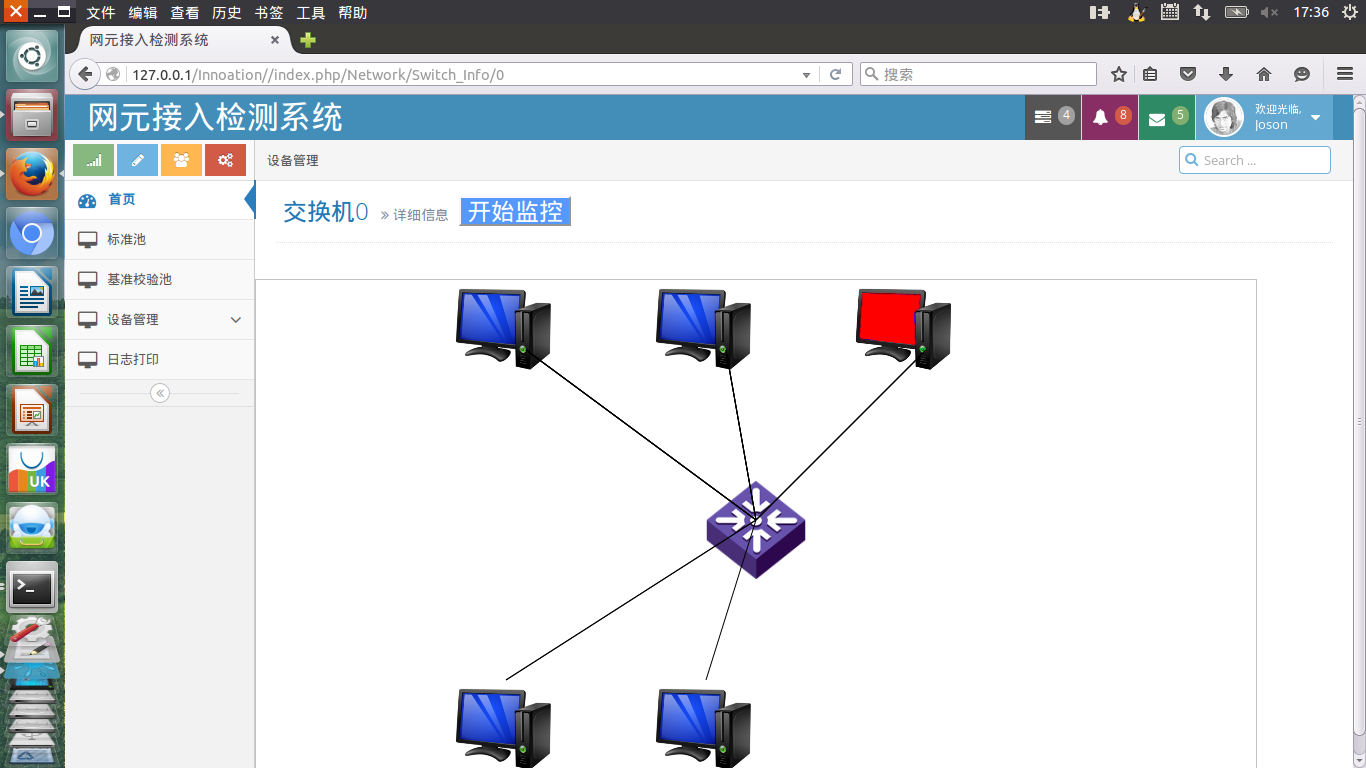
**图5-11 实验截图**

1. 第三部分是设备管理。当点击设备管理的图标上出现下拉菜单，里面是当前网络段的设备的显示，当点击某一设备时，后台调用数据采集器，将采集的设备信息传给后台，在通过控制器将信息传给视图模块，则界面上将出现该设备上的连接信息。这里主要是交换机设备。这里的web技术主要用的HTML5中的Canvas，canvas 元素使用 JavaScript 在网页上绘制图像。当按钮button获得点击事件，便开始使用JavaScript来进行图像的绘制。实验截图如图5-12所示：



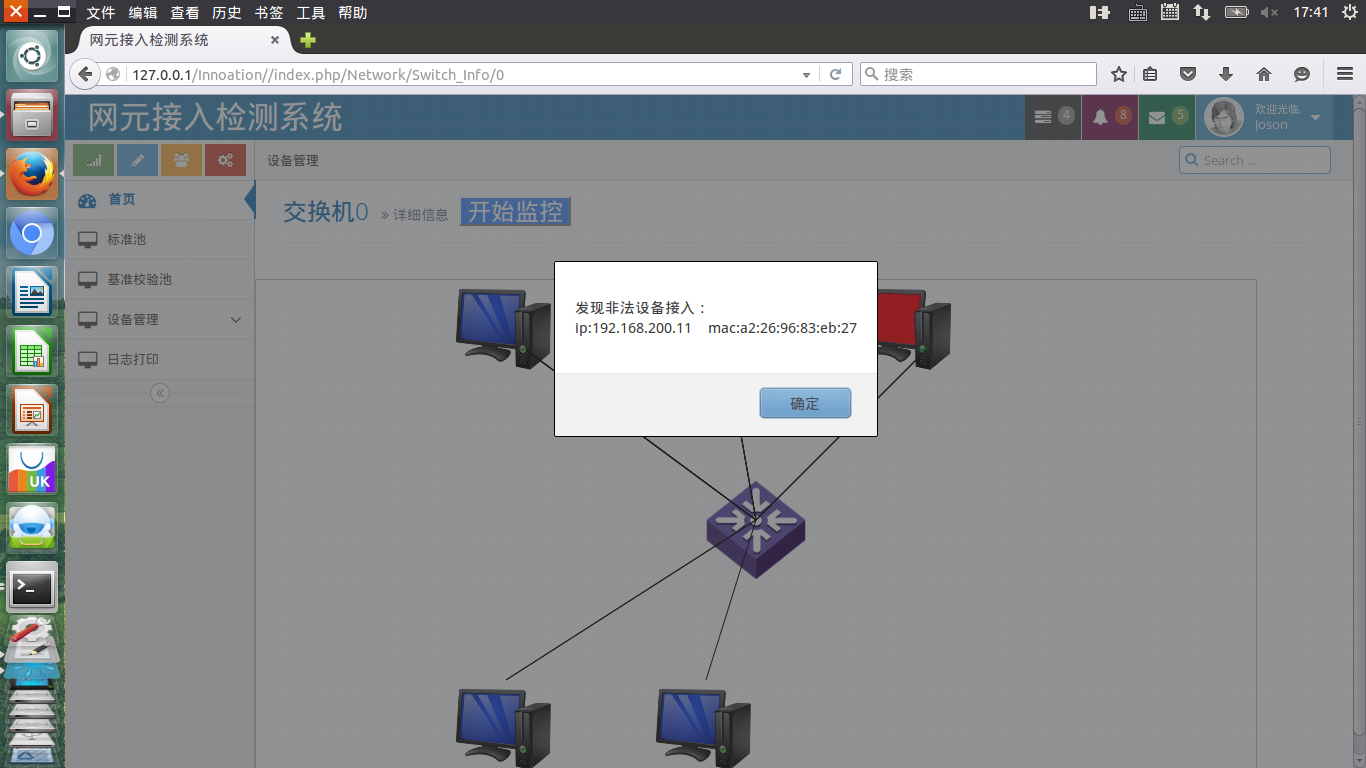
**图5-12 实验截图**

1. 当管理员点击开始监控的按钮时，后台便进入监控模式，如果没有发现新加入的设备则界面没有改变，当有新的设备接入时，后台会先调用数据分析模块对先加入的设备数据进行分析，判断设备的合法性。如果设备合法，则弹出提示框显示新加入合法设备，并且使用canvans在界面上加入一台合法的连接设备；如果设备非法，在界面上加入一台非法的连接设备，实验截图如图5-13所示：



**图5-13 实验截图**

并弹出提示窗口提示管理员有一台非法设备加入，实验截图如图5-14所示：



**图5-14 实验截图**

当管理员点确定时，便会弹出是否将该设备的警告信息消除的提示，如果管理员点击消除该设备的警告信息，则在数据库中将相应的设备信息进行更新并把警告项置为是，否则，置该项的信息为否。

前台和后台的交互用的是异步的AJAX,AJAX = 异步 JavaScript 和 XML。是一种用于创建快速动态网页的技术。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

$('#check').on('click',function(){

$.ajax({

'url':"<?php echo base\_URL();?>/index.php/Network/Checking/<?php echo $SwitchId;?>",

'type':"POST",

success:function(result){

alert(result);

........

}

})

})

$('#check').on('click',function(){

$.ajax({

'url':"<?php echo base\_URL();?>/index.php/Network/Checking/<?php echo $SwitchId;?>",

'type':"POST",

success:function(result){

alert(result);

........

}

})

})

## **5.4系统的测试**

本小节主要完成了对本系统功能的测试，检验各项功能是否可用，测试的功能包括：

对新加入网络的设备是否能够及时发现；发现新设备的延迟时间；对新加入设备的合法性是否能够进行准确的判断；对非法设备能否自动启动警报系统；对合法设备是否能够自动生成合法事件列表；对访问设备MIB信息库的时间戳设计是否合理；web端的响应时间限制是否影响到用户体验。

该测试的实验环境以实验室的实际环境为测试环境，经过多次的实验，得到的结论如下：

1. 在新设备入网检测功能上，系统的核心功能对测试有很高的响应度，基本对新加入的设备都能及时的发现，但是因为ARP表项存在老化时间，所以在设备接入以及到发现的过程中有几秒的延迟时间，这是系统本身不可避免的误差。
2. 在对数据进行过滤，分析统计的测试中，系统对正常的合理数据都能够准确的进行检测，但是对于ip地址和mac地址成对的盗用现象没法进行防范，这也是系统有待加强的一个方面。
3. 在web端进行信息展示时，界面能够清晰有效的向管理员展示当前网段中的设备使用情况以及设备合法性的相关信息。但是在发现违法设备到通知到管理员的这个时间段有一定的延迟，这主要和web端的响应速度有关系，不同的浏览器的相应速度有所不同；还有一个方面是在遇到较为复杂的后台数据处理或者是大量的数据处理的情形时，服务器的压力较大，对web端不能够及时的相应，这在用户体验上造成了一定的影响，也是后期需要改进的一个方面。
4. 报警系统上基本能够及时通知管理员违法设备的加入，系统对违法设备的标记警报信息以及消除警报信息系统均能够很好的进行处理。

# 第6章：结论

本文采用层次化的软件设计思想，基于SNMP简单网络管理协议，利用交换的工作原理，设计并实现了网元接入感知系统。该系统实现了对 一个局域网的设备的基本监控功能，能够及时发现新加入的设备，及时采集到设备加入的相关信息；对加入的设备进行有效的合法性分析，能够对不合法设备的的具体违法情况给出相应的分析报告；对非法设备的加入能够及时启动报警系统，通知管理员进行相应的处理，对合法设备能够生成合法事件列表以便管理员能够对设备的可用性有实时的掌握；同时也给管理员提供了一些对非法设备管理功能，如果该设备不能加入到合法设备的集合中，但是对网络的安全性并没有什么威胁，那么可以给管理员提供消除该设备警报的功能。本系统在经过全面的测试后，能够正常稳定的运行，实现来了需求分析的功能。

同传统的网元检测感知方法相比，本系统主要有一下有点：网络配置工作量最小、不需要在路由器上进行IP/MAC地址的绑定和交换机上进行MAC/端口的绑定，由于采用的是ARP包捕捉技术以及对交换机采用轮询的方式主动检查入网设备的信息，因此能够及时对非法设备的入网进行有效的监测，具有很强的实时性。并且能够度非法设备的部分非法行为进行准确的分析，给管理员提供有效的信息以便能够进行及时有效的处理非法入网的设备。

但是该系统还是有一些不足之处需要进一步改进：

1. arp表项老化时间的延迟对系统的性能方面有轻微的影响，这里需要找到一个设备访问时间的最佳间隔点；
2. 对非法设备中的ip地址和mac地址成对盗用系统没法进行检测；
3. 在对设备的合法性检测的算法中不够完善，仍需要改进；
4. 海量数据以及数据的高复杂性让算法运行在时间效率上有所下降，这造成了在web端的响应速度下降，一旦时间超过了URL自定的响应时间，系统将有可能错过对某个合法设备的信息展示并对报警系统的自启动造成延迟，这对用户体验有一定的影响，有待进一步改进。

# 谢辞

饮其流时思其源，成吾学时念吾师。值此论文完成之际，谨向我敬爱的导师陈竹敏老师以及吕彬老师致以崇高的敬意和诚挚的谢意。在学业的最后几个月里，陈老师将他的知识、经验心得毫无保留的传授给我，给我耐心的指导。陈老师指引我论文的写作方向，推荐了论文的架构，对本论文初稿逐字批阅，指正出其中误谬之处，进行了无私的指导和帮助，使我有了思考的方向，给予我无尽的启迪，他的一丝不苟、严谨细致的作风，将一直是我学习工作上的榜样。吕彬老师也在我写作的过程中给予我了莫大的帮助，在系统的具体实现中，和我一起探讨编程过程中遇到的问题，并耐心给我讲解，衷心的感谢吕彬老师。同时感谢实验室的师兄师姐们给与我的帮助和支持。

感谢山东大学四年里对我的培养，在山大的这四年，是我人生中最重要的一个阶段，是学校的培养，让我畅游在书海，在这浓郁的学术氛围下，学到了宝贵的科学文化知识，同时也提升了我在生活工作中的能力，潜移默化地塑造了我的人格，使我在长久的人生道路上能够有目标有自信有兴致走下去。凡我在处，便是山大，有你在时，那便是家。

# 参考文献

[1]刘远生 网络安全技术与应用实践[M] 清华大学出版社 2010.9

[2]冯国登 徐静 网络安全原理与技术[M] 科学出版社 2010.10

[3]W.Richard Stevens,TCP/IP 详解 卷1：协议[M] 机械工业出版社 2014.06

[4]马利 姚永雷 计算机网络安全[M] 清华大学出版社 2010.8

[5]闫宏生 王雪莉 杨军等 计算机网络安全与防护[M] 电子工业出版社 2010.11

[6]参考百度百科词条SNMP

[7] Subramanian Mani.Network management: principles and practice〔 M〕. Pearson Education Press,2000,141-344.

[8][9]王焕然 徐明伟 SNMP网络管理综述[J] 小型微型计算机系统 2004.03 25（3）

[10]叶锋 基于WEB和XML的网络管理机制研究与实现[J] 武汉科技大学 2004.7.10

[11]James,M.ketchmar.Open source network administration.Prentice Hall.2003.11 10-13

[12]龚尚福 刘二恩 冯健 基于SNMP的MIB树形结构研究与实现[J] 计算机应用与软件 2013.12 30（12）：163-164

[13]雷振甲 计算机网络管理及系统开发[M] 北京.电子工业出版社 2002：96-125

[14] 杨云江 计算机网络管理技术[M] 北京清华大学出版社

[15]戴一平 MVC设计模式在PHP开发中的应用[J] 计算机与现代化2011（3）

[16]周剑 张晓彤 王沁 SNMP协议动态MIB结构与高效查算法[J] 北京 网络与通信 2008.01 34(2)

[17]刘澜涛 李淼 高福祥 SNMP的网络资源管理系统的设计与实现[J] 网络管理 东北大学

[18]柴智 蒋烨 天津通信电信管理网网络管理系统的设计与实现[J] 电信工程技术与标准化 2003 (2): 34-42

[19]Pras A, Schonwalder J, Burgess M, et al.Key research challenges in network management[J]. IEEE Communications Magazine, 2007,45(10)：104-107

[20] Gupta A. Network management: current trends and future perspectives[J].Journal of Network and Systems Management，2006,14(4)：483-91.

[21]杨海燕 宋日聪 一个基于SNMP的网络监视系统的研究与实现[J] 信息技术教育与研究 2007(21)

[22]邱征 王红春 陈长胜 余亚刚 一种基于SNMP协议的网络管理管理端设计与实现方法 科技向导[J] 2013(21)

[23]朱伟 基于SNMP协议的网络拓扑结构系统改进与实现[D] 电子科技大学 2011

[24]权卓 基于SNMP的分布式网络管理系统的设计与实现[D] 西安电子科技大学 2010

[25]常伟 基于SNMP协议的网元管理系统的设计与实现[D] 北京交通大学 2014.06

# 附录

英文文献：

## NETWORK MANAGEMENT BASICS

A network device may be as simple as a desktop PC, modem, printer or laptop. These devices can send or receive information. Sometimes they are managed by a lonely network administrator whose only tool is an outdated protocol analyzer. More sophisticated network management technology includes distributed databases, auto-polling of the network devices, proxies to capture information for legacy devices and high-end workstations generating real-time graphical views of network topology changes and traffic. In general, network management is a service that employs a variety of tools, applications, and devices to assist human network managers to operate networks.

## NETWORK MANAGEMENT ARCHITECTURE

Most network management architectures use the same basic structure and set of relationships. End stations or managed devices, such as computer systems or routers, run software that enables them to send alerts when they recognize problems (for example, when one or more user-determined thresholds are exceeded). Upon receiving these alerts, management entities are programmed to react by executing one, several, or a group of actions, including operator notification, event logging, system shutdown, and automatic attempts at system repair. Management entities also can poll end stations to check the values of certain variables. Agents are software modules that first compile information about the managed devices in which they reside, then store information in a management database, and finally provide it to management entities within network management systems using a network management protocol. Well known network management protocols include the Simple Network Management Protocol (SNMP) and Common Management Information Protocol (CMIP). Management proxies are entities that provide management information on behalf of other entities. There are three version of SNMP in use because the features added to SNMP are hard to use.

## PROTOCOL MESSAGE FORMAT AND OPERATIONS

SNMP defines five special messages: get-Request, get-Next-Request, set-Request, get-Response, and trap. These messages can gather data from network elements for their proxies and set binary values used by the network device to control its behavior. These messages may be used to set specific values in the SNMNP conforming network elements and proxies.

Attributes of any managed object is monitored or set by the network management application using these five operations.

## IMPLEMENTATION CHALLENGES

There are some problems with this approach. Imperative commands are unsupported. There is no standard application programmer interface (API) for the manager-to-manager communication in version 1, but there is one for SBMP v2. SMI uses some fundamental ASN.1 definitions like integer, octet string and object identifier. Using these types ASN.1 allows construct complex one like structure and lists but ASN.1 is arcane and error prone. For example, a Network manager decides for his own reasons to send the decimal number ‘101’ to a device. The complexity of the ASN.1 coding is shown. Note that 101 is an integer and therefore of ASN.1 primitive type 2. The format of the encoding is

Tag type, Tag Number, Length, Value, where

1. Tag types

 Universal = 00

 Application = 01

 Context Specific = 10

 Private = 11

Since101 is a integar number choose the tag type 00000010.

2. Length is expressed in bytes

101=(00000000, 00000000, 00000000, 11000011)2

So 101 represented using ****4**** bytes

Using BER (00000010, 00000100, 00000000, 00000000, 00000000, 0110101)ASN.1

These definitions can be conveniently mapped into a C, or C++ or Java data structure that can be used by application programmers without dealing with the ASN.1 intricacies. They can be translated by special run-time libraries. Application designers may support all data types in a single library or create separate class file for each data type and at run time use instance of class. So creating class file is nothing but simulation of data type.

With the advantage of defining a special language and hiding the ASN.1 complexity errors are reduced and users can create new data types that their applications may need. These data types will in turn be translated into ASN.1 data for the SNMP messages. There is some performance overhead in creating ASN.1 classes. A typical data structure.

This document outlines the higher level network management functionality; but does not necessarily include interface specifics. These may change as the project is proceeds. The end result is a way to simplify the implementation of SNMP message so that journeyman programmers can write applications to keep a watchful eye on a set of IP based network devices. SNMP experts would not be required.

Thus we will retrieve and decode SNMP information from an SNMP agent and use a graphical tool to display the SNMP agent information. Finally, we will provide a set of interacting applets and successfully monitor network data in real-time.

SNMP is implemented in client/server paradigm and the manger/agent paradigm at the same time. The manager agent resides on the server and runs in the background. The client side manager program runs in foreground. Client program create request message, this message is converted into SNMP messages that are converted to IP packets to send into the network. . Packet contains all fields present in the SNMP message, and encoded using BER method. Finally encoded message send to agent. It will decode, and reply will send back to client. Selecting group and variable from group is nothing but selecting particular OID and calling SNMP session. Finally value of particular instance is extracted from MIB and displayed on screen,

## CONCLUSION

Writing programs to manage network devices is tedious and error-prone. It is a complex programming task. But once it works it works very well. We keep the advantages of the power of SNMP network management with a small increase in execution overhead but shed the liability of the demanding software implementation task. To-date we have demonstrated feasibility for the approved Internet Engineering Task Force (IETF) standard. As such tool will confine itself to, SNMPv1, and MIB II - Management information base, version two. Later we may extend implementation to higher version of SNMP, and wireless networks.

More importantly from a software engineering viewpoint we have shown how to hide complexity from the implementation of network management software.

中文译文：

使用SNMP协议的高效网络管理

一个网络设备可能是一台简单的台式电脑，调节器，打印机或者手提电脑。这些涉设备都能够发送和接受信息。有时候他们被某个仅有一个简单过时的网络协议分析器的网络管理员负责管理。更多复杂精细的网络管理技术包括分布式数据库，网络设备的自动轮询，捕捉传统设备和高端工作站产生的网络拓扑结构变化和流量实时图形视图的信的代理。总的来说，网络管理就是一项使用各种工具，应用和设备来帮助网络管理者进行网络管理工作的服务。

大部分的网络管理的体系架构使用了同样的基本结构和一系列的关系协议。终端站点或者管理设备，比如计算机系统或者路由器，当他们意识到出现问题时能够运行相关软件发出警报，例如当一个或者多个用户的超出了规定的阈值时。当收到这些情报时，管理实体就会在系统尝试自动修复时做出一系列的反应，包括通知操作员，记录事件日志，进行系统关闭。管理实体也可以查询终端站检查某些变量的值。代理软件模块首先编译关于它们所在的托管设备的信息，然后将信息存储在一个管理数据库，最后它在网络管理系统使用网络管理协议[1]把信息提供给管理实体。众所周知，网络管理协议包括简单网络管理协议SNMP，公共管理信息协议CMIP，管理代理是提供代表其他实体的管理信息的实体。这里有三个版本的SNMP协议在使用，因为加入到SNMP的功能都很难使用。

SNMP定义了五种特殊的信息：get-Request, get-Next-Request, setRequest, get-Response 和 trap。这些操作可以通过他们的协议从各网元处取得数据信息并且可以通过网络设备设置相应的参数从而控制他们的行为，达到管理的目的。这些消息可被用来设置在SNMP符合网络元素和代理特定值。

网络管理应用程序通过这五种操作对任何托管对象的属性进行监控和设置。

这个方法还是存在一些问题的，不能支持一些必要的命令。在版本1中对于管理者和管理者之间的交流没有一个标准的API给予支撑，同样在SBMPv2中也没有。SMI使用了一些标准的ASN.1来定义一些数据类型，比如整型，字符串型等。使用这些类型的ASN.1允许构造复杂的结构比如结构体和列表，但ASN.1是难懂且容易出错的。举个例子，一个网络管理者需要发送一个十进制的数字‘101’给设备，关于ASN.1的编码复杂度是可见的。注意，101是一个整数，然而ASN.1基本类型是二进制的编码格式，所以具体的编码如下：

1. Tag types  
   Universal = 00  
   Application = 01  
   Context Specific = 10  
   Private = 11
2. 因为101是一个整数所以选择标签类型00000010。
3. 2. Length is expressed in bytes  
   101 = (00000000, 00000000, 00000000, 11000011)2  
   So 101 represented using 4 bytes  
   Using BER (00000010, 00000100, 00000000, 00000000, 00000000,  
   0110101)ASN.1

这些定义可以很方便地映射成C，或C ++或Java数据结构，可以由应用程序员无需处理直接使用ASN.1。他们能够支持一些特殊的运行库。应用设计者能够在简单的库中支持所有的数据类型或者为每一种数据类型创建一个单独的类文件以便在运行的时候单独调用。所以创建类文件就是数据类型的集合。

随着定义一种特殊语言的优势的出现以及隐藏ASN.1的复杂度和出错率，使用者们能够创造出符合他们需求的数据类型。对于SNMP的消息来说，这里会有一些创建ASN.1的性能开销问题。

这些文档概括了更高级别的网络管理功能函数，但是并没有有效的包含具体细节之间的连接问题。这些可能会改变作为该项目的收益。最终的结果是简化SNMP报文的实施，使程序工作员可以编写应用程序来时刻监视基于网络设备的相关IP。这就不会对SNMP团队产生需求了。

我们收到从SNMP代理处发出的消息，对该消息进行破译并用图形工具展示这些信息。最后，我们将提供一套互动小应用程序，对网络数据进行实时监控。

SNMP是被同时用在客户端/服务器以及管理者/代理的实例中。管理者的代理存在于服务器端并在后台运行，而客户端的管理程序将在前段运行。客户端的应用程序会创建一个请求消息，这个消息将被转换成一个snmp消息然后以一个IP包的形式被发送到网络中。数据包中包含了SNMP的所有信息，用BER工具进行编码。最后编码信息将会被发送给服务器端的代理。然后消息会被解码，同时返回一个信息给客户端。选择组以及从组中选择变量并没有什么意义，但是选择特定的OID以及SNMP的版本信息就很重要了。最后特定实例的值是从MIB提取出来并显示在屏幕上。

编写关于管理网络硬件的计算机程序这项工作是冗长乏味并且极易出错的。他是一个非常复杂的程序任务，但是一旦他工作起来就会非常顺利。我们保持着SNMP网络管理权力的优势，但是需要略增长执行的开销略，这就使得当我在严苛执行的软件实施任务的时候需要放松一下标准。我们已经证明了最新的已获批准的互联网工程任务组（IETF）标准的可行性。这样的工具将自己局限于下SNMPv1和MIB II - 管理信息基地的2个版本。以后我们将会尝试实现更高的SNMP的版本以及无线网络管理的领域。

从软件工程的角度看，更重要的是我们已经展示了如何隐藏网络管理软件实施的复杂性。

这个具有前景的项目是一个为了简化创建软件来管理网络和系统的简单而强大的方式。这就好比从汇编代码转移到编译代码。和传统的SNMP包相结合势在必行的。高级语言的力量将通过翻译和面向对象的设计巧妙地结合而运用到到网络管理软件。结合高级编程语言会产生一些的性能开销，但是这会提高硬件的速度，所以这些性能开销是值得的。

1. [] 刘远生 网络安全技术与应用实践[M] 清华大学出版社 2010.9 [↑](#footnote-ref-0)
2. [] 冯国登 徐静 网络安全原理与技术[M] 科学出版社 2010.10 [↑](#footnote-ref-1)
3. [] W.Richard Stevens TCP/IP 详解 卷1：协议[M] 机械工业出版社 2014.06 [↑](#footnote-ref-2)
4. [] 马利 姚永雷 计算机网络安全[M] 清华大学出版社 2010.8 [↑](#footnote-ref-3)
5. [] 闫宏生 王雪莉 杨军等 计算机网络安全与防护[M] 电子工业出版社 2010.11 [↑](#footnote-ref-4)
6. [] 百度百科词条：http://baike.baidu.com/link?url=DWWRTYursGCgvWJb2GoWxt-GT-gGmxzYMDSIc8iz7dsMc-FNQBTpwXe4dYPdETcMSm1ForDGTeQFyGUnmEczqK [↑](#footnote-ref-5)
7. [] Subramanian Mani Network management: principles and practice[M] Pearson Education Press 2000:141-344. [↑](#footnote-ref-6)
8. [] 雷振甲 计算机网络管理及系统开发[M] 北京.电子工业出版社 2002：96-125 [↑](#footnote-ref-7)
9. [] 杨云江 计算机网络管理技术[M] 北京清华大学出版社 2008 [↑](#footnote-ref-8)
10. [] 戴一平 MVC设计模式在PHP开发中的应用[J] 计算机与现代化 2011 3:34-39 [↑](#footnote-ref-9)
11. [] James M.ketchmar Open source network administration[J] Prentice Hall.2003.11: 10-13 [↑](#footnote-ref-10)
12. [] 龚尚福 刘二恩 冯健 基于SNMP的MIB树形结构研究与实现[J] 计算机应用与软件 2013.12 30(12):163-163 [↑](#footnote-ref-11)
13. [] 王焕然 徐明伟 SNMP网络管理综述[j] 小型微型计算机系统 2004.3 25(3):34-36 [↑](#footnote-ref-12)
14. [] 叶锋 基于WEB和XML的网络管理机制研究与实现[D] 武汉科技大学 2004.7.10 [↑](#footnote-ref-13)
15. []netsnmp\_vardata的文档： http://www.net-snmp.org/dev/agent/unionnetsnmp\_\_vardata.html [↑](#footnote-ref-14)