

基于 NETCONF 与 Web 的网络管理系统的研究

作者姓名 张烨 学校导师姓名、职称 刘西洋教授

领 域 软件工程 企业导师姓名、职称 张钢高工

申请学位类别 工程硕士 提交学位论文日期 2014 年 10 月

学校代码 10701
分 类 号 TP31

学 号 1210122639
密 级 公开

西安电子科技大学

硕士学位论文

基于 NETCONF 与 Web 的网络管理系统 的研究

作者姓名： 张烨

领 域： 软件工程

学位类别： 工程硕士

学校导师姓名、职称： 刘西洋教授

企业导师姓名、职称： 张钢高工

提交日期： 2014 年 10 月

A Study of Network management system based on NETCONF and Web

A thesis submitted to
XIDIAN UNIVERSITY
in partial fulfillment of the requirements
for the degree of Master
in Software Engineering

By
Zhang ye
Supervisor: Liu xi yang Zhang gang
October 2014

西安电子科技大学 学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名：_____ 日 期：_____

西安电子科技大学 关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，获得学位后结合学位论文研究成果撰写的文章，署名单位为西安电子科技大学。

保密的学位论文在____年解密后适用本授权书。

本人签名：_____ 导师签名：_____

日 期：_____ 日 期：_____

摘要

随着近几年XML的发展，基于XML的NETCONF网络配置协议也越来越被得到重视，很多以NETCONF为传输协议的产品也慢慢取代之前的以SNMP技术为基础的产品。同时随着B/S体系的日趋成熟，基于Web的软件产品也成为市场的焦点。本文主要针对NETCONF与Web技术特点进行分析，在两种技术融合基础上设计实现一种全新的网络管理系统架构。在设计这套系统的过程中遇到的最大的技术难题是如何应用Web技术让系统应用到不同的网络产品上以及如何实现与SNMP兼容管理。

本文首先对NETCONF协议的实现以及其四个层次进行研究，并通过XML视图工具进行具体事件的操作。其次，对Web服务相关技术进行研究，研究包括HTTP和Ajax。其中HTTP的研究涉及内部请求结构、请求方法、状态码以及返回的响应信息，而针对前端交互技术Ajax主要从其实现过程和存在的缺陷进行研究。

然后结合NETCONF和Web技术的各个实现以及其技术融合特点设计综合性网络管理系统进行总体的设计和模型的实现。首先引出XML网络管理所存在的通用性问题；然后从整体出发，搭建系统总体的设计模型；接着对NETCONF关键技术应用进行深入的研究，关键技术主要表示对其各个层次进行详细的解剖，同时对NETCONF操作基本的流程进行深入的分析。研究完NETCONF技术应用再对Web网络管理的关键技术进行应用分析。首先对WBM的优势进行一一讲述，然后研究WBM实现的两种方式，并对WBM在安全性上的管理进行了应用分析。

依据前三章的理论将NETCONF技术与Web技术进行了结合，并对基于NETCONF与Web的网络管理系统进行功能模块的设计以及实现，将针对设计模型内的各个模块进行详细深入的分析 and 实现。主要包含对B/S网络模型的设计实现，NETCONF代理器的设计实现以及DOM转换器和SNMP代理器的设计实现。

最后一步工作是对系统进行了测试分析。依据本人一年实习期间所从事的工作，本文在网络管理模式选择上选择了嵌入式，即利用NETCONF代理实现网络配置管理。然后利用公司的路由设备为中心，搭建运行环境。最后对这套系统进行get和edit-config相关操作的测试分析，同时本文用SecureCRT远程登录检测Web测试的正确性以提高实验的准确性和严谨性。

关键词：NETCONF, Web, 网络管理

论文类型：应用软件技术类

ABSTRACT

With the development of XML in several years, the network configuration protocol which is based on XML protocol also has been taken seriously increasingly. A lot of products which are based on NETCONF are slowly replacing the previous ones using techniques such as SNMP. Due to the development of the B/S system at the same time, Web products have become the focus of heated debate. So if the NETCONF technology and Web technology get together, what chemical reaction will happen? This article will use these two techniques to study a network management system based on NETCONF and Web technology.

Firstly, This paper is started researching work of the model of NETCONF. This paper first discusses the NETCONF protocol and its four levels. And then we do detail events operation by XML viewer or SoapUI tool. Secondly, the paper will also do some related research of the Web-services technology. Firstly we will analysis HTTP/HTTPS protocol which mainly study HTTP internal request, the request method, the response status code and information returned. Lastly we will briefly analysis an important interactive technology -- Ajax (Asynchronous Javascript and XML) from the front-end, and analysis the defect points of this technology.

The second part of this paper will be combined of the two techniques and then researched the application detailly. Firstly, we will lead the general problem existed in network management based on XML, then this chapter will be started entirely to build the overall system model and research the key technologies of NETCONF deeply. The key technologies are meant the detailly anatomical representation of its various levels, also the deeply analysis of basic NETCONF operations. After further research on key technologies of NETCONF Technology, we begin to research the Web network-management analysis. Firstly we will count the advantages of WBM, then study the two ways to achieve of WBM, and then WBM management in security were analyzed.

The third part is the focus of this article, which is based on the theory of the first three chapters. This part is combined powerfully of Web technology and the NETCONF

technology. and we design and complete the functional modules of NETCONF-based and Web-based network management systems. We will design for each module within the designed module. It contains B/S network model design and implementation, NETCONF agent design and implementation as well as DOM converters and SNMP agent design and implementation mainly.

The fourth part we will test the system designed. During the internship year of work, I select embedded network management model, namely using NETCONF proxy for network configuration management. Then we build operating environment by using the company's routing equipment. Finally, the system will be analysed by get-related operations and edit-config-related operations, at the same time we use SecureCRT to test the correctness of Web in order to improve the accuracy of the experiment and rigor.

Keyword: NETCONF, Web, Network Configuration

Type of Dissertation: Application Software Technology

插图索引

图 2.1 NETCONF 层次结构图	5
图 2.2 HTTP 请求报文结构	7
图 3.1 网络管理系统总体模型图	12
图 3.2 网络管理系统总体流程图	13
图 3.3 基本流程图	23
图 3.4 基于 Web 管理的代理方案	24
图 3.5 基于 Web 管理的嵌入方案	25
图 4.1 B/S 模型框架	27
图 4.2 Web 浏览器请求发送图	28
图 4.3 Web 服务器工作模块	30
图 4.4 NETCONF 代理三层次	31
图 4.5 NETCONF 代理层工作流程图	32
图 4.6 DOM 转换器工作原理图	38
图 4.7 NMS、Agent 和 MIB 关系图	39
图 4.8 MIB 树结构示意图	40
图 4.9 子树 OID 与子树掩码对应关系图	40
图 5.1 系统运行环境	43
图 5.2 IPv4 管理模块 Web 页面	44
图 5.3 请求报文 NETCONF 数据	46
图 5.4 Ipv4 管理页面列表显示	47
图 5.5 设备 Vlan 虚接口 ip 信息	48
图 5.6 Loopback6 接口创建窗口	49
图 5.7 Loopback6 接口创建请求报文	50
图 5.8 创建接口响应报文	50
图 5.9 设备 Loopback6 接口信息	51
图 5.10 IPv4 管理编辑页面	51
图 5.11 设置接口请求报文	53
图 5.12 响应报文返回情况	53
图 5.13 设备 Loopback6 接口信息	54

表格索引

表5.1 Ipv4页面get操作测试.....	48
表5.2 Ipv4页面get操作测试结果.....	49
表5.3 Ipv4页面新建Loopback6操作测试.....	52
表5.4 Ipv4页面设置Loopback6接口配置操作测试.....	55

缩略语对照表

缩略语	英文全称	中文对照
SNMP	Simple Network Management Protocol	简单网络管理协议
MIB	Management Information Base	管理信息库
NETCONF	Network Configuration Protocol	网络配置协议
CLI	Command-line Interface	命令行界面
HTTP	Hypertext transfer protocol	超文本传送协议
Ajax	Asynchronous Javascript And XML	可扩展标记语言
XML	Extensible Markup Language	可扩展标记语言
DTD	Document Type Definition	文档类型定义
RPC	Remote Procedure Call Protocol	远程过程调用协议
WBM	Web-Based Management	基于 Web 的网络管理模式
DOM	Document Object Model	文档对象模型
XSD	XMLSchemasDefinition	XML 结构定义
LIPC	LightInter Process Communication	轻量级进程间通信协议
SDK	Software Development Kit	软件开发工具包
XML	Extensible Markup Language	可扩展标记语言
NMS	Network Management System	网络管理系统
EDOM	Encapsulated Document Object Model	压缩文档对象模型
OID	Object Identifier	对象标识符

目录

摘要.....	I
ABSTRACT	III
插图索引.....	V
表格索引.....	VII
缩略语对照表.....	IX
目录.....	XI
第一章 绪论	1
1.1 选题背景及意义.....	1
1.2 国内外现状分析.....	1
1.3 论文工作内容.....	2
1.4 论文组织结构.....	3
第二章 相关技术概述	5
2.1 NETCONF 理论	5
2.2 HTTP 技术.....	6
2.2.1 HTTP 技术概述	6
2.2.2 HTTP 请求结构	7
2.2.3 HTTP 请求方法	8
2.2.4 状态码	8
2.2.5 响应信息	9
2.3 Ajax 技术.....	9
2.3.1 Ajax 技术简介.....	9
2.3.2 Ajax 的缺陷.....	10
2.4 本章小结.....	10
第三章 网络管理系统的总体设计	11
3.1 通用性问题分析.....	11
3.2 系统需求分析.....	11
3.3 系统的总体模型设计.....	12
3.4 NETCONF 关键技术应用	14
3.4.1 RPC 层技术应用	14
3.4.2 操作层技术应用	15
3.4.3 内容层技术应用	22
3.4.4 基本流程实现应用	23
3.5 Web 网络管理关键技术应用.....	24
3.5.1 WBM 的优点	24

3.5.2 WBM 实现的两种方式	24
3.5.3 基于 Web 管理的安全性	25
3.6 本章小结.....	26
第四章 网络管理系统的详细设计与实现	27
4.1 B/S 模型的设计实现.....	27
4.1.1 Web 浏览器	27
4.1.2 Web 服务器	30
4.2 NETCONF 代理处理器实现	30
4.2.1 NETCONF 代理总体结构	30
4.2.2 NETCONF 代理工作流程	32
4.2.3 NETCONF 代理解析设计	34
4.2.4 NETCONF 数据模型	34
4.3 DOM 转换器与 SNMP 处理器的实现	37
4.3.1 DOM 转换模块	37
4.3.2 SNMP 处理器	39
4.4 本章小结.....	40
第五章 网络管理系统测试与分析	43
5.1 运行环境的搭建.....	43
5.2 网络管理系统测试分析.....	44
5.2.1 get 操作测试	45
5.2.2 edit-config 操作测试分析	48
5.3 测试分析小结.....	54
第六章 结束语.....	55
6.1 论文工作总结.....	55
6.2 后续工作展望.....	55
参考文献.....	57
致谢.....	59
作者简介.....	61

第一章 绪论

本章主要阐述目前 SNMP 技术在网络管理上的缺陷, 以及兴起的基于 XML 的 NETCONF 技术, 再结合国内外对于 NETCONF 技术的研究, 从而宏观上对论文所描述的系统有一个清晰的了解。本章重点在于论文的研究工作介绍和论文架构的安排, 本论文主要从六大方面进行介绍和研究, 实现 NETCONF 与 Web 技术相结合的综合网络管理系统。在这个过程中遇到的困难是如何在以 SNMP 管理技术为主的网络产品上实现此综合网络管理系统的无缝集成。

1.1 选题背景及意义

SNMP技术是目前主流的网络管理技术, 它从20世纪80年代末开始就应用于IP网络的管理^[1]。因为SNMP的数据结构和协议操作过于简单让它迅速取得了成功, 同时这也成为它自身最主要的缺点。目前虽然SNMP仍然是一种广泛使用的网络管理协议, 不过随着网络市场一步步的发展, SNMP已经开始暴露出自身的缺陷, 管理者用户也在寻找另外的技术来弥补SNMP的缺陷。

为了解决SNMP操作简单和结构简单的问题^[1], 人们一直在寻找一种新的技术。在这个前提下, XML技术应运而生, 它的出现给网络管理注入了全新的力量, 而基于XML的网络管理方式也迅速成为国内外研究和开发的热点。由于XML之于SNMP的诸多优势, 人们已经逐渐开始研究基于XML的网络管理技术, 令人振奋的是, 这几年来应用XML的网络管理技术发展很快, 国内外的研究机构及企业基于这项新技术的研究推出了诸多相关的网络产品和新型应用。与此同时, 在国内外的研究机构及企业不断深入研究下, 新的实践难题又出现了, 那就是如何在实现基于XML的网络管理的标准化的情况下完成通用性。根据这个棘手的难题, 研究机构提出了一种全新的基于XML网络配置协议—NETCONF, 这个协议在数据编码上与XML一致, 同时在网络传输上与Web服务技术相结合, 这样就可以真正解决网络管理上的标准化和通用化问题了。这种基于NETCONF与Web技术的应用研究将会进一步推动网络管理服务的发展。

1.2 国内外现状分析

IETF(Internet Engineering Task Force)在2003年5月成立了NETCONF工作组, 该工作组主要是为了提出一个全新的基于XML的网络配置协议(NETCONF)。该工作组于2006年2月8日提出了最新的NETCONF协议的草案NETCONF Configuration Protocol Internet—Draft, 2006年12月NETCONF协议以RFC4741-4744文档作为标准^[2]。标准分别从数据编码, 传输方式和操作控制实现等方面来定义, 规定在编码上

采用XML语言，在传输方面采用可靠的面向连接的传输TCP协议实现，同时在操作控制方面应用简单可控的远程调用方式（RPC）实现方便简捷的控制^[2]。相比于SNMP操作，NETCONF操作语言能够具有更好的内在逻辑性，更复杂的操作，对于操作对象而言会有更好的操作效率和结构性。总之，NETCONF技术在提出后变成了市场上研究机构和企业研究网络管理的焦点。目前，市场上已经有很多根据NETCONF和Web服务技术来开发的网络管理产品，包括S3公司，appinf公司以及瑞典的Trail-f systems等。下面分别介绍各个公司以NETCONF为基础的网络管理架构的实现。

首先是S3公司(Silicon&Software Systems)。该公司利用NETCONF技术研发创建了一种可扩展的架构^[2]，为网络开发者提供了方便的操作行，使开发人员可以利用插件即可增加扩展系统架构的功能。其可扩展性是利用四个模块，分别是CLI,Web/XML,SNMP,NETCONF来实现的。

再者就是澳大利亚的appinf公司开发POCO Open Service Platform(OSP)C++具有强大的跨平台性^[2]。它在POCO系统上用C++开发实现了NETCONF原型。PocoNETCONF支持SOAP方式进行传输，支持Web Service技术。

然后就是瑞典Trail-f Systems的以XML为基础包含扩展的NETCONF代理的ConfD^[2]。该方案不仅具有扩展性，支持传统的SNMP管理模式，而且通过提供开放的Web接口实现了用户的自由访问。与S3公司的embeddedMind项目提出的NETCONF的实现方案类似，它为用户提供了很好的API接口，这些接口可以为用户的需要添加或删除相关组件。

当然国内很多研究所也在积极发展研究NETCONF技术，例如华为的北京研究所。他们在SNMP为基础的MIB管理上无缝的实现技术转移^[2]，在兼容SNMP的前提下，过渡到NETCONF技术为基础的设备体系架构，这一过程不仅包括SNMP MIB的原语的转换而且还包括数据模型的转换。

NETCONF作为一项全新的网络管理实现协议，虽然得到了国内外的热烈追捧以及得到了广范围的技术研究支持，但是这一技术还不够成熟，尤其在国内仍然有很长的路要走，不仅仅是追赶国外的技术，更主要的是在我过的产品中成功应该NETCONF这一技术，以带来更多的方便快捷的服务和更大的收益。本论文主要是实现NETCONF协议在互联网产品管理上的技术应用和实现，通过Web友好的图形界面实现对整体网络的监控配置和管理。

1.3 论文工作内容

本文根据目前国内外发展现状设计并实现一套全新的综合网络管理系统。这个系统将以NETCONF技术为系统数据信息格式，向用户提供更为复杂全面的配置

操作和更多的传输信息。同时该系统以Web技术为基础，支持B/S网络通信架构，在浏览器端向用户提供一套友好的Web用户界面，用户能够利用图形界面进行实时网络监控管理。在服务器端包含NETCONF代理对NETCONF请求进行解析处理，同时实时操作设备配置信息。总体来说，本篇论文的研究内容主要包括以下几个方面：

- 1) 分析国内外的NETCONF技术发展现状和其优点以及带来的网络管理效率；
- 2) 提出NETCONF与Web技术的融合，通过对HTTP,Ajax的技术实现，深入研究了这一结合的实现过程，以及在实际研究开发中遇到的技术难点；
- 3) 在NETCONF与Web技术融合管理中，先从实现WBM网络的管理入手进行研究，之后再为构建综合的网络管理一目标制定架构模型；
- 4) 详细设计并实现基于NETCONF与Web技术的综合网络管理系统，包括对模型内部B/S网络通信架构，NETCONF代理，DOM转换器和SNMP代理详细的设计与实现；
- 5) 测试系统的NETCONF准确性，本文从get操作和edit-config相关操作两大操作入手，以系统Ipv4模块为例，分别完成前端Web页面操作和SecureCRT远程登陆界面CLI命令行操作，对两种方式进行结果上的比较验证，从而测试这套系统的准确性。

1.4 论文组织结构

本文的结构是这样安排的：

第一章，绪论。分别从国内外对于网络管理系统的研究状况进行分析，以及当前管理中的缺陷矛盾进行阐述，然后重点介绍本论文所要完成的任务，以及实现目标的总体架构规划。

第二章，相关技术概述。论文主要实现主要NETCONF与Web技术的融合，所以分别从两方面进行技术实现的应用，一是NETCONF的实现，而是Web技术中HTTP和Ajax这两个重要协议的实现。

第三章，综合网络管理系统的总体设计。该章节是论文体系架构最重要的章节之一，先对综合网络管理架构进行需求分析，之后设计了架构实现模型。

第四章，综合网络管理系统的详细设计。在综合管理架构模型建立起来之后，就结合相应的技术进行设计实现。整个过程是按照实现流程图和模型来进行DOM转换也处理器的设计的。

第五章，综合网络管理系统测试与分析。在综合网络管理系统实现之后，还有对其性能和功能进行测试研究，确保系统的稳定性和安全性，以及是否满足用户需求。

第六章，总结与展望。总结本文工作，指出存在问题并指出下一步的研究工作。

第二章 相关技术概述

网络管理是对路由器为核心的局域网、企业网、城际网进行网络监控和管理，它是保障网络安全、稳定、可靠、高效运行的重要条件。网络管理发展的过程中出现了很多网络管理技术，这些技术应用在计算机网络的各个方面，其中包括基于SNMP和基于NETCONF的网络管理技术。本章主要概述NETCONF与Web服务相关技术的理论知识。

2.1 NETCONF 理论

NETCONF (Network Configuration Protocol, 网络配置协议)^[3]主要用于对网络设备进行远程管理和监控。NETCONF 采用 C/S 模式为用户提供了快速简便的配置方法来实现对设备的管理。这样客户端/服务器的模式，可以为用户提供方便的途径来查询设备不同模块上参数的设置，并通过对参数的修改达到对设备的控制管理。在这一过程中，客户端发送请求给服务器时采用的报文是 XML 封装之后的报文格式。当服务器收到此格式的报文之后会返回相同格式(XML 格式)的报文给客户端。

NETCONF 采用独立的分层结构^[3]，每个层独立完成相应模块的封装操作，并向上层提供相应服务。这种分层结构有一个特别好的好处就是这样可以使每个层次都专注于自己对应的方面，优化以及简化实现过程，同时可以减少层与层之间的耦合度和相关性，这样在各层内部进行变更的时候可以减少对其他层的影响^[4]。NETCONF 包含提供管理对象集合的内容层，对管理对象进程控制的操作层，进行请求封装的 RPC 层以及通信协议层。具体分层如图 2.1 所示。

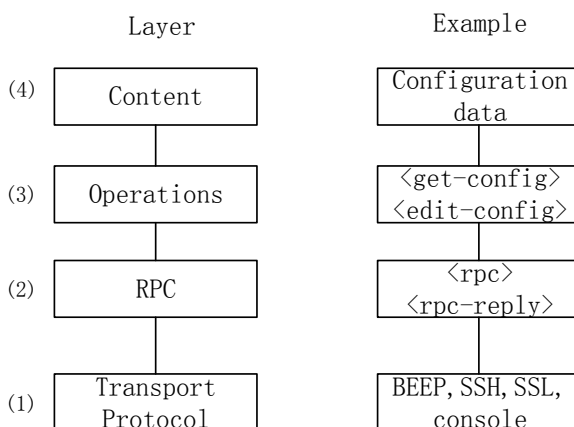


图 2.1 NETCONF 层次结构图

内容层^[5]表示被管理对象的集合。在内容层，里面的对象内容规范需求都来自数据模型中，而原有的像 MIB 这种数据模型对配置管理存在着一些缺陷，例如不

允许创建和删除行，不支持复杂的表结构等。正因为内容层复杂的表结构以及诸多不确定性，才使得内容层结构没有定义在 RFC4741 中。到目前为止，虽然由于 NETCONF 技术正在大力发展生成了一些 NETCONF 建模语言以及数据模型，例如 XSD。但是对内容层的规范化还是没有办法做到，只能做到对对应设备标准进行规范设计。

操作层是对被管理对象进行设置的操作动作进行定义的^[5]，包括 NETCONF 基本功能集即操作基本原语集合。在 SNMPv1 中，SNMP 为了简化能力只定义了五种基本操作，即取值、设值和告警三个方面。但是 NETCONF 则不同，它包含了九种基本操作，在功能上主要包括四个方面：取值操作、配置操作、锁操作和会话操作，取值相关操作包括 get、get- config、get-bulk，设置相关操作则包括 edit-config、copy- config 和 delete- config。锁操作则是包括 lock 和 unlock，这个操作是为了防止并发执行产生的混乱。会话操作则是包括 close- session 和 kill- session，这两个操作都是用来关闭和结束会话操作的，而且这是相对比较上层的操作。

RPC 层表示 NETCONF 的一种模块机制，这种模块机制与协议本身无关。它是通过使用 <rpc> 标签元素来对 NETCONF 协议的客户端请求进行封装，相应的用 <rpc- reply> 标签元素来对服务器端的响应数据进行封装。如果没有出错的情况下，服务器端将客户端所需的数据或配置成功的提示信息封装进 <rpc- reply> 元素，但是如果客户端请求报文出错或者服务器端处理出错时，服务器端将一个包含详细错误信息的 <rpc- error> 元素来封装到 <rpc-reply> 元素中进行返回。

NETCONF 是采用 TCP 传输协议^[6]，不再是 SNMP 管理系统中的 UDP 协议，不仅提供了可靠的面向连接的服务，而且通过通信端口之间保持永久连接^[7]，提供了有序的数据传输。在传输层，NETCONF 为了能通过加密和认证等方法来保证网络的安全性^[8]，特别制定了 RFC4742、RFC4743 和 RFC4744^[9]这三种文档技术，分别对应 SSH、SOAP、和 BEEP 三种传输协议。

2.2 HTTP 技术

超文本传输协议（HTTP）^[10]是分布式，协作式，应用于超媒体系统上的应用层协议。HTTP 是数据通信中的万维网的基础。它使用超文本的结构来包含文本节点之间的逻辑链接（超链接）的文本。下面对 HTTP/HTTPS 进行简述。

2.2.1 HTTP 技术概述

HTTP 是一种作用在客户端/服务器计算模式的请求/响应的协议。当进行 Web 浏览时，客户端常常利用该协议发送 HTTP 请求报文已得到服务器的响应得到想要的服务，当服务器接受并解析请求后，会返回需要提供的资源，例如 HTML 文件等内容，或代表客户机执行其它功能，完成后返回响应消息给客户端。该响应

包含有关该请求的完成状态信息，并且还可以包含其消息体请求的内容。

例如采用用户代理（UA）的浏览器服务。其他类型的用户代理包括由搜索提供商（网络爬虫），语音浏览器，移动应用程序的索引软件和其他软件访问或显示网页内容。

HTTP 被设计成允许调解网络单元，以改善或允许客户端和服务端之间的通信。高流量的网站往往受益于由上游服务器提供的 Web 缓存服务器，该服务器通过提供内容元素来提高响应时间。Web 浏览器的缓存以前访问网络资源，并为了可以减少网络流量而重新使用它们。私有网络里的 HTTP 代理服务器可以方便客户在没有全局可路由地址情况下沟通，这主要通过与外部服务器转发邮件。

HTTP 放在互联网协议里的应用层。它通常是基于下一层即传输层的传输控制协议（TCP）进行可靠的传输的。不过 HTTP 也可以使用不可靠的协议，如在简单服务发现协议（SSDP）的用户数据报协议（UDP）。

HTTP 的资源在网络中通过统一资源标识符（URI）进行识别和定位。或更具体地说是在使用 HTTP 或 HTTPS URI 模型的统一资源定位器（URL）上。URI 和超文本标记语言超文本链接（HTML）文档组成了内联超文本文件的网络。

HTTP / 1.1 的是原始的 HTTP（HTTP/1.0）的修订版。在 HTTP/1.0 中，到同一台服务器单独的连接是由每一个资源进行请求。 HTTP / 1.1 的可以重复使用性连接多次来下载图片，脚本，样式表等页面已交付之后的事情。因此，HTTP/1.1 通信会感觉建立 TCP 连接这一开销上延迟很短。

2.2.2 HTTP 请求结构

HTTP 请求从客户端发出去的信息体包括以下几个部分：

第一部分为 RequestLine（请求行），存放着请求报文发送的 URL 路径，对应下图的 METHOD 行。

第二部分为 RequestHeader(请求头)，存放着 HTTP 请求相关属性的值，对应图中的 Header-Name: value 所在的区域。

第三部分为请求内容（可选），存放着请求报文内详细的配置信息，就是图中的 Optional request body 部分。

METHOD/path-to-resource HTTP /Version-number
Header-Name-1:value Header-Name-2:value
Optional request body

图 2.2 HTTP 请求报文结构

2.2.3 HTTP 请求方法

HTTP 定义的方法（有时也称为动词），用来指示要在所识别的资源上执行所需的操作^[10]。无论是动态生成还是预先存在的数据，资源的显示都依赖于服务器的实现。通常情况下，资源对应一个文件或驻留在服务器上的可执行文件的输出。在 HTTP/1.0 规范中定义的 GET, POST 和 HEAD 方法基础上，HTTP/1.1 规范增加了 5 个新方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT。任何客户端可以使用任何方法将服务器配置为所支持的方法的任何组合。如果一个方法是未知的，它将被视为一个不安全的方法。可以定义的方法的数量是没有限制的，而且此允许指定后续的方法，当然前提是不破坏现有的基础设施。例如，WebDAV 的定义的 7 个新方法和 RFC5789 规定的 PATCH 方法。

GET- 请求指定资源的表示。使用 GET 只能检索数据并且要求没有其他影响。W3C 已经发布的指导原则说明了这种区别，他说，“Web 应用程序的设计应以上述原则为知情，同时也应得到相关的限制。” 请求头要求相同，将请求处理的数据以键值对形式放在 URL 后面，响应将对应于 GET 请求，但是没有响应主体。这种方法主要用于获取包含在响应报头的元信息，而不必传输整个内容。

POST -请求服务器接收包含在请求的由 URI 标识的网络资源下属的实体信息。发布数据可能是一个公告板，新闻组，邮件列表，或者评论跟帖留言。HTTP 主体的数据块是提交 Web 表单到数据处理过程或者添加到一个数据库的结果。

PUT-请求发送 URI 下封闭的实体。如果 URI 是指已经存在的资源，它将被修改。如果 URI 不指向一个现有的资源，那么服务器可以创建该 URI 的资源。

DELETE -删除服务器 URI 指定的资源。

TRACE- 回显接收到的请求，使客户端可以看到什么（如果有的话）。

CONNECT-转换请求连接到一个透明的 TCP/IP 隧道，通常以方便 SSL 加密通信（HTTPS）通过未加密的 HTTP 代理。

OPTION-返回服务器支持对指定 URL 的 HTTP 方法。这可以用于通过请求“*”，而不是一个特定的资源来检查 Web 服务器的功能。

PATCH-适用于局部修改的资源。

HTTP 服务器至少需要实现 GET 及 HEAD 方法，若在可能的情况下，也应包括 OPTIONS 方法。

2.2.4 状态码

从 HTTP/1.0 开始，HTTP 响应的第一行称为状态行，响应包括数字状态代码（如“404”）和一个文本原因短语（例如，“未找到”）。用户代理主要处理的响应的方式取决于代码和次要的其他响应报头字段。如果用户代理遇到不能识别码，就可以使用代码的第一位数来确定一般级响应。

此外，标准的原因短语只是建议，Web 开发人员可以自由裁量换成当地等值的语句。如果状态代码表示有问题，用户代理可能会显示原因短语向用户提供有关问题的性质的进一步信息。该标准还允许用户代理来尝试解释原因短语，虽然这可能是不明智的，因为该标准明确规定，状态代码为机器可读而原因短语是人类可读。HTTP 状态码主要分为五组，客户端和服务端之间的返回名为：信息 1XX，2XX 成功，重定向 3XX，客户端错误 4XX 和服务端错误 5XX。

2.2.5 响应信息

该响应消息包括以下内容：

- 1) 一个状态行，其中包括 HTTP 协议的版本、服务器的响应状态码和状态文本描述。（例如，HTTP / 1.1 200 OK，这表示客户端的请求成功）；
- 2) 响应头域，如 Content-type: text / html；
- 3) 一个空行。
- 4) 一个可选的消息体。

在状态行和其他头字段都必须结束于<CR><LF>（回车后跟一个换行符）。空行必须只包含<CR><LF>，并没有其他的空格。

2.3 Ajax 技术

Ajax^[11]（简称异步 JavaScript 和 XML），是一组在客户端用来建立异步 Web 应用程序相互关联的 Web 开发技术。使用 Ajax，Web 应用程序可以异步地发送数据到服务器和从服务器取回，而不与显示的现有的页面的行为干预。数据可以使用 XMLHttpRequest 对象进行检索。尽管名字有 XML，不过目前不需要使用 XML，JSON 常被用来代替（AJAJ），并且请求也不需要是异步的。

Ajax 不是一种单一的技术，而是一组技术。HTML 和 CSS，可以组合使用，以标记和风格信息。DOM 的访问使用 JavaScript 动态显示（并允许用户进行交互）提供的信息。JavaScript 和 XMLHttpRequest 对象提供了浏览器和服务端之间的异步数据交换，以避免全页面重载的方法。

2.3.1 Ajax 技术简介

该 Web 技术 Ajax 已经广泛应用到与后台服务器通信的 Web 应用程序^[12]中，而不干扰页面的当前状态。主要由下列几种技术组成：

- 1) HTML（或 XHTML）和 CSS 呈现；
- 2) 用于动态显示文档对象模型（DOM）的交互数据；
- 3) 以 XML 进行数据交换，而 XSLT 进行 XML 操作；
- 4) 异步通信的 XMLHttpRequest 对象；
- 5) JavaScript 来把这些技术融合在一起。

然而，目前已经出现了一些发展中的应用技术与曾经的 Ajax 技术相似。不过 XML 不需要用于数据交换，而 XSLT 不要求对 XML 数据进行操纵。虽然其它的格式，如预格式的 HTML 或普通文本也可以被使用，不过 JavaScript 对象符号(JSON) 还是经常被用作于数据交换的另一种形式中。

2.3.2 Ajax 的缺陷

在支持 HTML5 的浏览器，使用连续的 Ajax 请求动态生成的网页并没有自动与浏览器的历史记录引擎来注册自己，所以点击浏览器的“后退”按钮可能不会返回浏览器页面的早期状态，也许不是回到了之前访问过的最后一个完整的页面。这样的行为（在页面之间进行导航，而不是页面的状态之间进行导航）可能是可取的，但如果需要细粒度的跟踪页面的状态的话，支持 HTML5 后有一个解决办法是使用隐形 iframe 来触发改变浏览器的历史记录。通过 Ajax 技术实现的解决方法是，在访问启用了 Ajax 的页面时更改 URL 片段标识符(URL 之后的“#”的部分)，并监测其是否有变化。HTML5 提供了全面的 API 标准的浏览器的历史记录发动机工作。

根据 Ajax 应用程序的性质，动态页面的更新可能会中断地干扰用户的交互，尤其是如果工作在不稳定的因特网^[13]连接。例如，编辑搜索领域可能引发查询到服务器进行搜索的完成，但用户可能不知道，在搜索完成后弹出窗口即将到来，如果互联网连接速度很慢，在弹出的列表中可能会出现用户已经着手做别的事情等不方便的时候。

用户的浏览器不支持 JavaScript^[14]的 XMLHttpRequest 或者干脆禁用此功能的时候，将无法正确使用依赖于 Ajax 的页面。设备如智能手机和掌上电脑可能没有所需的技术支持，虽然这是越来越少的问题。让用户进行功能的唯一途径是退回到非 JavaScript 的方法。通过确保链接和表格都可以正确解决这些问题。

本网络管理系统就是应用 Ajax 技术在浏览器端发送携带 NETCONF 配置的 HTTP/HTTPS 报文，进而对路由进行网络配置管理。

2.4 本章小结

本章首先介绍了网络管理的基本理论，然后介绍了网络管理中的技术——NETCONF 技术以及 Web 服务相关技术，对 NETCONF，HTTP/HTTPS 以及 Ajax 技术进行理论上的分析介绍。下一章将对基于这两种技术的综合网络管理系统进行总体的设计分析。

第三章 网络管理系统的总体设计

NETCONF的网络管理技术在有限的时间内得到了快速发展,同时也暴露出很严重的问题,即保证NETCONF在网络管理中的通用性。于是提出了基于NETCONF与Web服务的综合网络管理技术,该技术得到快速发展和应用,目前应用该技术的网络管理产品已经投放市场。本章主要从整体设计以NETCONF与Web技术为基础的综合网络管理系统,并且对NETCONF的应用进行详细的分析。

3.1 通用性问题分析

在基于XML的网络管理技术兴起之后,由于这是一门尚未成熟的技术,所以这一路上会有种种不完善的地方以及漏洞出现^[16]。因此该技术模型的发展还是需要一步步的完善和修正。当前传统的网络管理体系还很不完善,存在各种问题,比如网络管理者只能通过特定的计算机来管理网络。而基于Web的网络管理模型为管理者了提供更方便的管理方式,不必再受限于固定的工作站,网络管理脱离了操作平台的束缚。这样我们通过以XML为介质,以Web服务作为传输工具,就可以实实在在的解决这个平台通用性这个大难题了。

3.2 系统需求分析

本人学习一年时间,一直从事 Comware v7 Web 前端的模块研发设计,包括 Web 页面交互开发以及测试。这一年让本人对 Web 开发有了更深的理解和感悟,对这套网络操作系统有了更多的了解,因此本文将参照这个项目来设计一套基于 NETCONF 与 Web 的网络管理系统。

本文的综合管理系统主要是基于本人于实习公司 H3C 的项目。这个项目叫 Comware v7 Web 项目, Comware v7 是一套应用于该公司路由产品上的网络操作系统,用来对网络的配置和管理。而 Comware v7 Web 项目则是将这套原先通过命令行式的配置管理模式转化为通过浏览器的页面可视化界面配置管理模式。而这就利用到了 Web 技术以及 NETCONF 技术。

这个系统将以基于 NETCONF 的 Web 服务为基础,应用 JavaScript+ CSS+ HTML^[14] 前端技术进行 Web 前端页面开发,应用浏览器/路由器模式进行对路由设备的操控配置。如果这样的系统放到路由器的页面上,那么就会发生很大的变化。首先,以前在路由器搭建网络平台的时候必须进入这个命令行进行一项项的配置,并且这对平台会有各种各样的限制。但是如果系统应用浏览器形式的操作页面,用户会发现无论用什么媒体什么平台,只要有浏览器,能上网,用户就可以通过 HTTP/HTTPS 进行 URL 访问,进入到这个操作平台界面;其次,这种友好的图形

界面使那些普通的非专业客户也可以进行专业的网络配置和管理，再也不需要去敲那些繁琐生硬的命令行了。

针对用户目前的需求，系统的目标可以概括为下面四点：

- 1) 让系统真正实现网络操作跨平台性；
- 2) 实现支持前端语言 JavaScript 进行 NETCONF 基本原语和扩展能力集的操作；
- 3) 实现系统获取、设置操作功能，并与命令行操作的结果吻合，实现 Web 端与 CLI 命令行端共同管理；
- 4) 通过 JavaScript ajax 的 XMLHttpRequest 对象传输 NETCONF，真正实现异步通信。

3.3 系统的总体模型设计

在这一节，本人将对基于 NETCONF 与 Web 的综合网络管理系统进行框架模型设计。这个模型将会以浏览器/路由器为网络架构，NETCONF 报文为传输配置信息，HTTP 为传输协议。并且我们将用到目前流行的异步传输技术（Ajax）作为 Web 页面请求方式。在输入输出方面，系统将从用户在前端的页面操作开始，以前端接收设备返回信息并且显示为结束，从中经历了 javascript 将请求报文封装为 NETCONF 格式，Ajax 用 XMLHttpRequest 对象封装 NETCONF 报文，浏览器通过 HTTP 协议传输，Web 服务器接收 HTTP 报文并解析，NETCONF 代理处理器处理请求操作并生成响应报文返回这样一系列的步骤过程。系统的总体模型如图 3.1 所示。

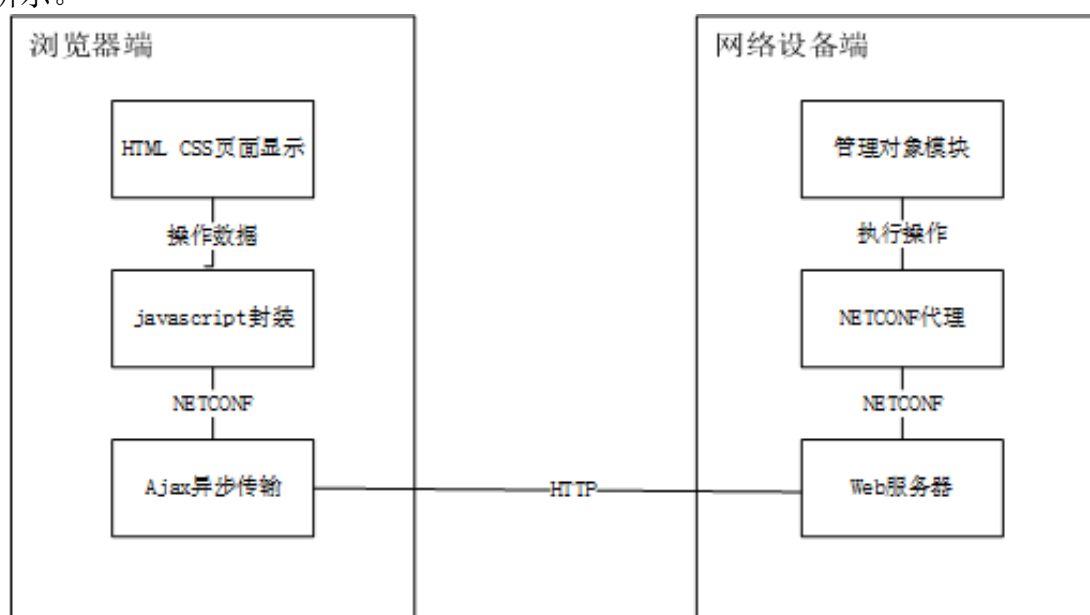


图 3.1 网络管理系统总体模型图

同时该系统的总体模型的工作流程图如图 3.2 所示。

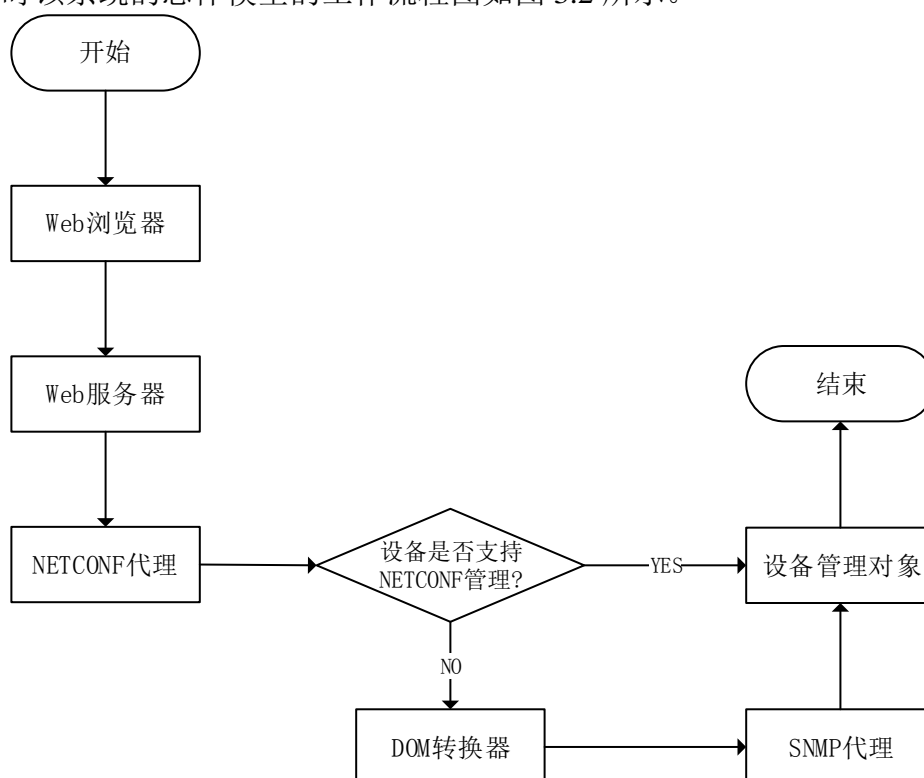


图 3.2 网络管理系统总体流程图

这个模型充分体现了良好的用户交互体验性，让用户可以在提交配置后继续在页面上处理其他的事务，比如查询即时的网络配置参数。而这都是通过使用 Ajax 这种异步机制来实现的。可以这么说，Ajax 大大优化和提高了页面处理事务的效率和性能，从而在这个以用户体验为中心的互联网时代能站稳脚跟。

依照图 3.1 中描述的 NETCONF 综合网络组成系统，分为：Web 浏览器，Web 服务器，NETCONF 代理，DOM^[15]转换器，SNMP 代理。引入 SNMP 代理的原因是目前很多企业还是应用 SNMP 进行网络操作管理，所以要有转换网关这种第二方案技术。以下简单介绍之。

1) Web 浏览器作为用户代理向 Web 服务器申请 Web 管理页面，同时为用户提供可视化操作界面。用户可以通过浏览器页面查看后台设备的管理状况以及对设备进行自定义的配置操作。可以说浏览器代表了客户端中的“设备”，用户操作浏览器等同于在操作远端通过 Web 连接的服务器设备。我们通过从 Web 服务器申请的 javascript 程序文件将配置参数封装在 NETCONF 报文中，并通过 Ajax 技术由 XHR 对象将 NETCONF 报文封装起来，最后通过 HTTP 异步传输给 Web 服务器。

2) Web 服务器负责存放前端 javascript+CSS+HTML 代码文件，同时负责接收浏览器发送的 HTTP 请求报文数据，同时作为中转器将解析的请求报文转发给

相应的程序模块。此时 Web 服务器将解析出来的 NETCONF 请求数据发送给 NETCONF 代理模块，再对响应数据信息进行封装通过 HTTP 返回给浏览器。

3) NETCONF 代理负责接收 Web 服务器发送的 NETCONF 请求报文，对报文进行解析提取操作层及内容层内容。同时再根据设备是否支持 NETCONF 配置来进行输出。如果设备支持 NETCONF 管理则将 NETCONF 代理的输出发送给设备，设备直接操作配置更改。反之则发送给 DOM 转换器，由 DOM 转换器进行 NETCONF/SNMP 的格式更改，最后通过 SNMP 代理对设备进行操作，并封装响应 NETCONF 报文返回给 Web 服务器。

4) DOM 转换器针对设备不支持 NETCONF 管理的情况，将接收的 NETCONF 配置参数转化为 SNMP 的报文请求。并接收 SNMP 发送过来的 SNMP 报文。

5) SNMP 代理接收 DOM 转换器的报文请求，对设备配置进行配置和读取。并将结果返回给 DOM 转换器。

如图 3.1 所示的模型中，系统内部各个模块有相互独立又相互协作的部分。独立关系表现在每个模块都是在自己的内部完整的接口资源体系下完成的功能设计，没有与外部模块产生大量的耦合现象；而相互协作关系就表现在模块之间的请求发送和响应关系上了，当客户进行系统的网络操作配置时，就产生了一系列浏览器到设备再从设备到浏览器的请求/响应，我们称之为请求响应链。这种请求响应链引导着 NETCONF 数据报文完成从发送到将接收再解析，然后再次发送的过程。

3.4 NETCONF 关键技术应用

本节主要对系统内部的 NETCONF 技术进行应用上的研究，主要包括 NETCONF 的各个层（RPC 层，操作层和内容层），着重研究操作层的各个操作动作。同时还研究分析 NETCONF 的基本流程实现。

3.4.1 RPC 层技术应用

NETCONF 协议以 RPC 通信模型为基础^[17]，使用 <rpc> 和 <rpc-reply> 元素作为该协议的通信的请求-响应端口，此外还会涉及到 <rpc-error> 元素。

<rpc> 元素是用来实现封装 NETCONF 请求的。<rpc> 元素通过使用 message-id 属性，实现接收方与发送方的消息匹配。RPC 的接收方直接将它存储为 <rpc-reply> 消息中的 message-id 属性值。

例如：

```
<rpc-reply message-id = "211" xmlns = "urn:ietf:params:XML:ns:NETCONF:
base: 1.0">
```

```
<!--content-->
```

```
</rpc-reply>
```

除了这两个最主要的元素之外,接收端在收到<rpc>请求进行解析发现请求出错之后,就会在返回的<rpc-reply>元素中包含<rpc-error>元素。当然,如果接收端在运行<rpc>的请求时出现多个不一样的错误,那么<rpc-reply>返回元素可以包含多个<rpc-error>元素。

下面举例说明<rpc-error>元素的封装情况。例如:收到的<rpc>元素没有 XMLns 这一属性。

```
<rpc message-id='211'>
```

```
  <get>
```

```
  </get>
```

```
</rpc>
```

下面是返回的包含<rpc-error>元素的<rpc-reply>元素

```
<rpc-reply message-id="211">
```

```
  <rpc-error>
```

```
    <error-type>rpc</error-type>
```

```
    <error-tag>missing-attribute< error-tag>
```

```
    <error-severity>error< error-severity>
```

```
    <error-info>
```

```
      <bad-attribute>XMLns</bad-attribute>
```

```
      <bad-element>rpc</bad-element>
```

```
    </ error-info >
```

```
  </ rpc-error >
```

```
</rpc-reply>
```

当然,在请求成功以后,响应的<rpc-reply>会添加一个<ok>元素来表示ok了。

3.4.2 操作层技术应用

SNMP提供了五种操作,包括取值,配置和警告^[18]三方面。而NETCONF则提供了九种基础操作。分别是get, get-config, edit-config, copy-econfig, delete-config, lock, unlock, close-session和 kill-session。下面就具体介绍这九种基本操作以及新增的一些操作。

1) get 相关操作

网络设备中的资源包括只能进行读操作的状态数据和可进行修改的配置数据。NETCONF 协议采用 get 和 get-config 两种操作来实现处理配置数据和状态数据的分离的。对于状态数据和配置数据这两类资源的不同之处在于 get-config 只取配置数据的值。同时针对只取几项并非全取的情况,我们在原操作基础上又添加了两

个操作，即 `get-bulk` 和 `get-bulk -config`。

(1) get 操作

该操作用于获取运行状态数据和配置数据，举例如下。

```
<rpc message-id="211" xmlns="urn:ietf:params:XML:ns:NETCONF:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://example.NETCONF.com/NETCONF/data:1.0" >
        <Ifmgr><!--特性名-->
          <Ports><!--表名-->
            <Port><!--行名-->
              <PortIndex></PortIndex><!--列索引-->
              <Name></Name> <!--以下为数据列-->
              <IfIndex></IfIndex>
            </Port>
          </Ports>
        </Ifmgr>
      </top>
    </filter>
  </get>
</rpc>
```

如上例中若只指定<Ifmgr/>特性名，则该特性下所有支持获取的相关数据信息都会返回；若指定了<Ifmgr/>特性名和<Ports/>表名，则该表下的所有数据都会返回；若如上例中指定了行名或指定了行中各列的名称，但没有给各列赋值，则相当于对该表进行 `get` 操作；若只指定行中索引列<PortIndex/>的名称，则只返回索引列的值；若只指定了数据列<Name/>的名称，则返回的是索引列<PortIndex/>和指定列<Name/>的值。

若只想返回某行的数据，则需指定该行的索引列值。如上例中指定索引列的值为 1，即<PortIndex>1</PortIndex>，不管其他数据列<Name/>等是否在报文中指定，返回的数据信息都会包括所有列的信息，即上例表中<PortIndex/>、<Name/>和<IfIndex/>的值都会返回，但如果只指定了数据列的值，如上例中只指定<Name/>的值，则返回的数据信息只有索引列<PortIndex/>的值和指定列<Name/>的值。

(2) get-bulk 操作

该操作可以指定 `count` 值和索引，从指定的索引开始获取 `count` 值指定条数的运行状态数据和配置数据。支持获取的模块信息同 `get` 操作。

举例如下。

```
<rpc message-id="101" xmlns="urn:ietf:params:XML:ns:NETCONF:base:1.0" >
  <get-bulk>
    <filter type="subtree">
      <top xmlns="http://example.NETCONF.com/NETCONF/data:1.0" >
        <Ifmgr xc:count="5">                                <!--特性名-->
          <Ports xc:count="3">                                <!--表名-->
            <Port>                                           <!--行名-->
              <PortIndex>1</PortIndex>                      <!--列索引-->
              <Name></Name>                                  <!--以下为数据列-->
              <IfIndex></IfIndex>
            </Port>
          </Ports>
          <InterfaceCapabilities>                            <!--表名-->
            <Interface>                                       <!--行名-->
              <IfIndex></IfIndex>                            <!--列索引-->
              <Configurable></Configurable>
              <Shutdown></Shutdown>
            </Interface>
          </InterfaceCapabilities>
        </Ifmgr>
      </top>
    </filter>
  </get-bulk>
</rpc>
```

get-bulk 操作的 count 值只能在特性名和表名中指定,若在特性名中指定该值,则对该特性下所有表都生效,获取数据时特性下每个表都会获取指定 count 值条数的数据;若在表名中指定该值,则只对指定的表生效;若在特性名中和表名中都指定了 count 参数,则表名中的 count 参数优先生效。

如上例中特性<Ifmgr xc:count="5">中指定了 count 值为 5,特性下的<Ports xc:count="3">表中指定了 count 值为 3,而<InterfaceCapabilities>表未指定 count 值,则进行 get-bulk 操作时,Ports 表返回的是 3 行数据,而 InterfaceCapabilities 表返回的则是跟特性中指定的 count 值一致的数据行,即 5 行数据。

当未指定索引值时,从第一条数据开始输出 count 条数据;当指定索引值时,

则从指定索引的数据的下一条开始输出 count 条数据；当指定的索引值不存在时，则找第一个比它大的索引作为下一个索引值，从该索引值开始输出 count 条数据。若表中数据不足 count 条时，则返回表中剩余所有的数据。

如上例中假设 Ports 表有 5 行数据，索引值依次为 1—5，当未指定索引值时，则返回索引值为 1、2、3 的 3 行数据；当指定索引值为 2 时，则返回索引值为 3、4、5 的 3 行数据；当指定索引值为 0（不存在）时，则返回索引值为 1、2、3 的 3 行数据；当指定索引值为 3 时，则返回索引值为 4、5 的 2 行数据，此处表中数据剩余不足 3 条，因此只返回剩余的 2 行数据。

除上述之外，对于只指定特姓名、表名、索引列的名称或指定所有列的名称等情况下返回数据值的情况 get-bulk 操作跟 get 操作相同。

(3) get-config 操作

该操作与 get 操作类似，只是获取的数据不同，用来获取配置数据且只能获取非缺省配置数据。在报文中指定操作为 get-config 操作，需要注意的是 top 参数中需要指定 config 命名空间：

<top xmlns="http://example.NETCONF.com/NETCONF/config:1.0">如没有配置数据时 get-config 操作将返回空的<data>。

(4) get-bulk-config 操作

该操作获取的数据与 get-config 操作相同，都是用来获取动态配置数据并且只能获取非缺省配置数据的，只是该操作跟 get-bulk 操作类似，可以用来获取指定索引开始的指定条数的数据，跟 get-bulk 操作的约束完全相同。

2) 配置相关操作

1) 创建操作

NETCONF 中 edit-config 操作中共有三种创建操作：merge、create 和 replace。三种操作各有不同，基本报文格式举例如下。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:NETCONF:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <top xmlns="http://www.hp.com/NETCONF/config:1.0"
Web: operation = "xxx">
      <VLAN><!--特性名-->
      <VLANs><!--表名-->
```

```

        <VLANID><!--行名-->
            <ID>10</ID><!--列索引-->
            <Description>test-vlan-10</Description><Name>test-
vlan-10</Name>
            <AccessPortList></AccessPortList>
            <Ipv4Address></Ipv4Address>
            <Ipv4Mask></Ipv4Mask>
        </VLANID>
    </VLANs>
</VLAN>
</top>
</config>
</edit-config>
</rpc>

```

其中 `edit-config` 包含有 `target` 标签，表示目标的配置数据库；而 `running` 标签表示运行中的配置数据库。

创建操作都是对配置数据进行操作，因此命名空间为 `config`。operation 中的 xxx 为三种操作中的其中一种。下面详细介绍各创建操作极其不同之处。

`merge` 操作用于修改指定对象的值，其操作简单，对于数据列为空值或者不指定数据列的时候是不会对数据进行配置处理的。假如对象不存在并且允许创建对象的话就可以进行参数配置进行创建操作；假如对象不存在也不允许用户进行创建命令那么 `merge` 会返回失败。

如上述举例中，如果 `vlan 10` 存在，则修改 `vlan10` 的描述和名字，由于端口列表为空值，对其不做处理，该表中还有未列出的 IPv4 地址（即注释内容，相当于未列出，方便该处解释说明，后同），也不对其做处理；如果 `vlan10` 不存在，并且允许进行创建操作那么就可以在对此 `vlan` 进行创建和配置。

`create` 操作只支持创建操作，不支持配置的修改操作。如果 `vlan10` 不存在但是允许被创建，执行 `create` 操作会创建 `vlan10` 并配置 `vlan10` 的描述和名字，对空值的列或该表中未列出的数据列则不作处理；如果 `vlan10` 存在，但是其需要配置的数据列的值（如上例中的描述和名字）为缺省值，则仍可下发配置；但如果指定要配置的列的值（如上例中的描述和名字）为非缺省值，则返回 `data-exist` 错误。

`replace` 操作只替换指定对象的配置，不会尝试创建对象。如果指定的对象不存在，则返回错误信息表示目标对象不存在；如果指定的对象存在则进行替换。该操作在数据列为空值或表中有未列出的数据列时，会将这些数据列的值恢复为

缺省值。

如上述举例中,如果 `vlan10` 不存在,执行 `replace` 操作会返回对象不存在错误;如果 `vlan10` 存在,则将 `vlan10` 的名字和描述替换回举例中的值,对于空值的数据列和未列出的数据列如上例中的端口列表和未列出的 IPv4 地址都将恢复为缺省值。

2) 删除操作

NETCONF 中 `edit-config` 操作共支持两种删除操作: `remove` 和 `delete`。这两种删除操作存在一定的差异,基本报文格式同创建操作,只需将 `operation` 中 `xxx` 部分替换为两种操作中的其中一种。

对于 `remove` 和 `delete` 操作,索引列必须指定值,否则返回错误。在只指定索引列时,会删除索引列指定行的所有数据;在指定了数据列后,则只会删除指定数据列的数据,将其恢复为缺省值。两者的不同之处在于:对于 `remove` 操作,如果指定的索引列或数据列不存在时也是返回成功的,而 `delete` 操作在指定的索引列或数据列不存在时则返回对象不存在错误。

如上例中,若存在 `vlan10` 且只指定了索引列 `<ID>10</ID>`,没有指定其它的数据列,则执行 `remove` 或 `delete` 操作时,会将 `vlan10` 删除;若如上例中写的,还指定了 `vlan` 的名字和描述两个数据列,则只会将这两个数据列的内容删除,变为缺省值,而不会将 `vlan10` 删除。此处需要注意,索引列的值必须指定,多于数据列的值可以指定为任意值也可以为空值,删除操作不会读取该值。如果 `vlan10` 不存在或数据列中描述或名字已经为缺省值,对于 `remove` 操作来说是返回成功的,而 `delete` 操作则返回对象不存在的错误。

3) 其他操作

(1) 事件订阅操作

该操作只对当前连接生效,订阅某事件后设备上如果发生用户订阅的事件则会向订阅该事件的 XML 视图发送相关的信息^[19]。该操作只能在 XML 视图下进行订阅,不支持通过 SoapUI 工具^[20]订阅。

该操作实现的功能类似于日志信息,用户视图下使用的是日志信息方式来提示用户当前所发生的事件,而 XML 视图下则使用事件订阅功能,其中 `code` 字段表示助记符, `group` 字段表示模块名, `severity` 字段表示等级。此处需要注意,终端配置的允许输出的日志信息的最低级别需要与所订阅事件发出的信息级别相同或更低时,XML 视图才能收到该订阅事件的相关信息^[21]。

举例如下。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:NETCONF:base:1.0">
  <create-subscription xmlns="urn:ietf:params:xml:ns:NETCONF:notification:1.0">
```



```

<stream>NETCONF</stream>
<filter>
  <event xmlns="urn:ietf:params:xml:ns:NETCONF:event:1.0">
    <Code>LINK_UPDOWN</Code>
    <Group>IFNET</Group>
    <Severity>Notification</Severity>
  </event>
</filter>
<startTime>2012-12-10T15:10:00</startTime>
<stopTime>2013-12-10T15:20:00</stopTime>
</create-subscription>
</rpc>

```

报文中 stream 事件流目前只支持 NETCONF, Code、Group、Severity。上例中对接口管理模块的接口 updown 事件进行监控, 在订阅时间内发生接口 updown 事件时就会在 XML 视图下收到相关的信息。

```

<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:NETCONF:notification:1.0">
  <eventTime>2013-01-24T10:15:04</eventTime><!-- 事件发生时间-->
  <event xmlns="urn:ietf:params:xml:ns:NETCONF :event:1.0">
    <Group>IFNET</Group><!-- 模块名-->
    <Code>LINK_UPDOWN</Code><!-- 助记符-->
    <Chassis>1</Chassis><!-- 框号-->
    <Slot>4</Slot><!-- 槽号-->
    <Severity>Notification</Severity><!-- 等级-->
    <Context> interface GigabitEthernet1/0/0/1 is down.</Context>
  </event>
</notification>

```

(2) save 操作

```

<rpc message-id="101" xmlns="urn:ietf:params:XML:ns:NETCONF:base:1.0">
  <save>
    <!--file>test.cfg</file-->
  </save>
</rpc>

```

save 是用来保存当前配置的操作命令, 可以通过 <file> 来指定保存的配置文件

名。

(3) 关闭会话操作

关闭会话操作有两种：close 和 kill 操作。close 操作用于关闭当前会话：

```
<rpc message-id="101" xmlns="urn:ietf:params:XML:ns:NETCONF:base:1.0">
  <close-session/>
</rpc>
```

kill 操作用于关闭除当前会话外的其它会话：

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:NETCONF:base:1.0">
  <kill-session>
    <session-id>1</session-id><!--选择要关闭的会话的 session-id-->
  </kill-session>
</rpc>
```

还有一些操作如 lock、rollback、action 等未在本文中列出，这些操作都比较简单。

3.4.3 内容层技术应用

内容层是用户自定义数据的层次，里面包含了实质要进行配置传输的数据。在上述例子中内容层主要包含在操作节点内。如edit-config, get, get-config。

同时根据操作的不同，会有特殊的一些标签节点，如get和get-config内会有filter标签，<filter>这个标签表示获得哪些值。这是NETCONF下协议特有的过滤机制。过滤机制包括子树过滤和XPath两种。

相应的，配置操作里包含了<target>,<config>,<error-option>子标签。

```
<?XML version="1.0"?>
<rpc message-id="100" xmlns="urn:ietf:params:XML:ns:NETCONF:base:1.0">
  <edit-config>
    <target>
      指定编辑目标
    </target>
    <error-option>
      失败时默认操作
    </error-option>
    <config>
      <top xmlns="http://www.h3c.com/comware:5.0/config/edit">
        指定模块名，子模块名，列名，表名
      </top>
```

```

    </config>
  </edit-config>
</rpc>

```

这是客户端发送给服务器端的请求报文，里面涵盖了这三个子标签。当成功后，会有<ok>标签的响应报文返回。

NETCONF定义了一个配置数据库数据库,用来存储网络设备中正发挥作用的所有配置数据。

3.4.4 基本流程实现应用

请求用户端与接收服务端的 NETCONF 通信流程类似于 TCP，请求端会先向服务器端发送建立请求的报文，如<hello>标签。同时等待返回的响应报文。如果服务器端没有正在处理的请求且处于空闲状态，则会返回响应报文，里面包含了操作请求的序列号。请求用户端得到请求序列号并应用于这次的具体配置操作请求报文头。操作的具体类型包含 get、get-config、edit-config、save 等。完成配置之后，请求用户端发送<close>标签，服务器端接收请求并断开连接。基本流程如图 3.3 所示。

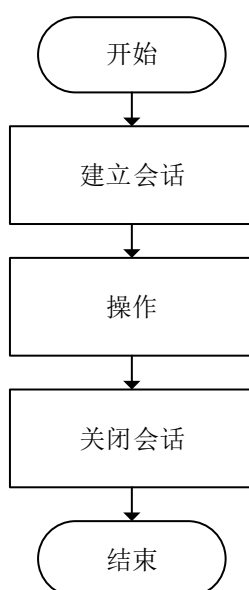


图 3.3 基本流程图

对于建立会话，可以通过 TELNET/SSH/Console 登录设备，在用户视图下输入 XML 命令进入 XML 视图，通过 hello 报文交互能力集，建立 XML 会话，然后进行相关操作。该方式没有回显功能，只能通过拷贝写好的报文来下发配置（此处需要强调下，在 XML 视图下发送交互报文时需要添加结束符：]]>]]>），对于获取数据和配置等操作返回的应答是逐条显示的。还可以通过 SoapUI 工具来建立会话，需要在设备上配置本地用户，通过报文交互，验证用户名和密码，发送 hello

报文建立会话。该方式操作相对简单，且容易查看，但对于获取数据和配置等操作返回的应答是先获取到所有的数据或配置后再统一返回，如果中间任何一条出现错误，则所有数据或配置都会返回失败。

3.5 Web 网络管理关键技术应用

随着越来越多的企业运用 Intranet, Intranet 网络管理也开始在寻求着不同于以往的突破和改变，因此 WBM 应运而生。WBM 可以让管理者通过 Web 管理企业的网络,大大提高网络管理的效率。这是因为它将不再是只能通过特定的应用程序或者管理计算机来配置网络，而是可以用 Web 浏览器这种用户代理来进行管理，WBM 是一次真正的网管变革。

3.5.1 WBM 的优点

WBM^[22]比传统工具更加方便且功能强大，可以提供更直接更好用的图形界面，方便管理者通过任何浏览器来访问网络配置，把工作者从固定的工作站中解放出来。

3.5.2 WBM 实现的两种方式

WBM 有两种基本的实现方法，并且这两种方法互不干扰^[23]。第一种方法是利用代理^[24]通过在网络管理站点增加 Web 服务器实现的，具体过程如图 3.4 所示的系统模式。该模式下一方面 Web 浏览器与代理服务器直接进行 HTTP 通信，另一方面网络设备与代理之间通过 SNMP 协议进行通信，从而整体上实现了网管工作站与网络设备的通信的保持维护。

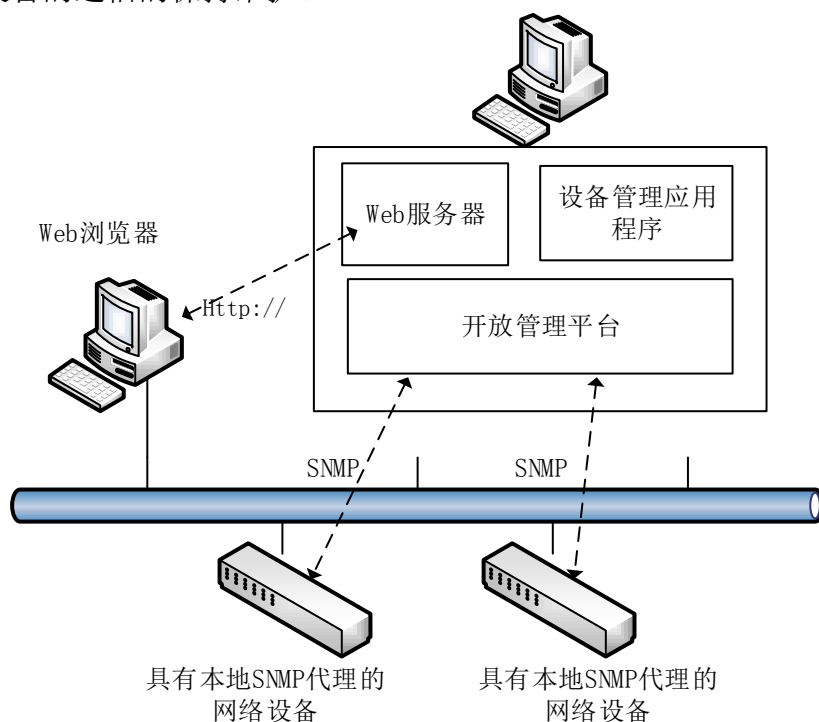


图 3.4 基于 Web 管理的代理方案

第二种是采用嵌入式^[24]的方式进行通信,其实现过程见图 3.5 所示。每个设备用户拥有互不干扰的 Web 服务器,再通过 Web 服务器与网络设备之间的集成管理,管理员就可以简单快捷的应用浏览器访问服务器实现网络设备管理配置。

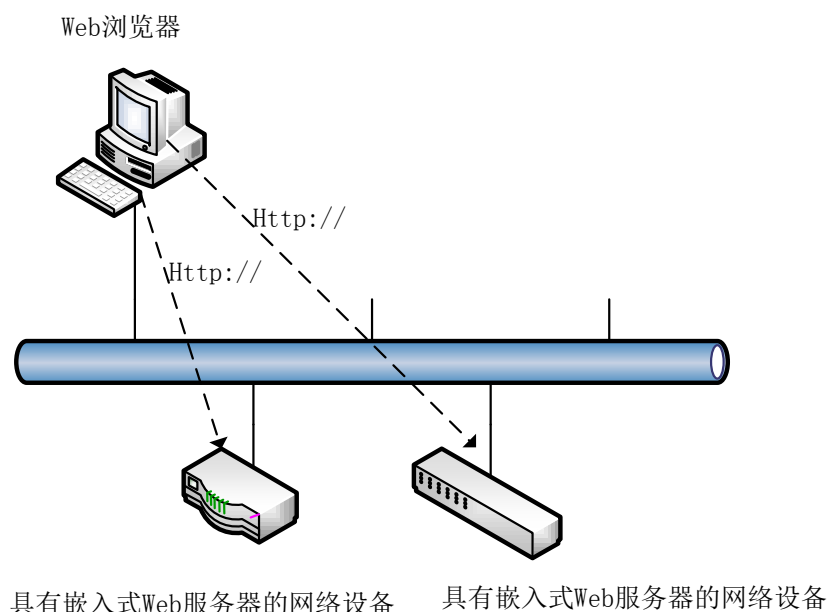


图 3.5 基于 Web 管理的嵌入方案

在这两种方案中的代理方案都增加了浏览器的访问,不仅保留原先网管系统的优势,更重要的是通过代理实现网络设备管理采用的依然是传统设备管理中的 SNMP 协议,仅仅改变的是代理与服务器之前的访问而已。这种方法都可以实现局域网内的设备集成映射,且方案的实现也比较简单不需要做过多的改变。

这两种方法中的嵌入式方案通过独立的 Web 服务器让每个设备都可以通过各自的界面进行管理控制,不再向以前在传输之前通过 SNMP 做格式转换,不仅实现了个性独立的管理而且提高了网络设备的传输速率。

这两种方案是发展的必要趋势,肯定会广泛应用于各个企业组网中。一方面各个企业还不可能完全抛弃以 SNMP 为主的网络管理系统,这就需要在产品上实现兼容,采用 WBM 时,最好使用代理,这样不会造成设备的破坏,也不需要设备做更多的变更,减少了开支,通过将会拥有更广泛的市场。

3.5.3 基于 Web 管理的安全性

企业网络中最需要解决和关注保障的问题就是网络的安全^[25]。一般来说企业网络都是使用防火墙来通过对外部的未授权的非法访问的拒绝来实现安全,从而达到网络的隔离。要想访问服务器就得使用正确的密码才可以进行访问,因为设备采用安全认证进行加密,一般外来设备是无法直接破解访问的。要想对企业服务器网络进行访问就得在身份认证用户界面输入正确的用户名和密码,即使登陆

网络设备了也得在设备授权前提下才可以进行配置管理。同时保障传输的安全性上企业应用 HTTPS（全称: Hyper Text Transfer Protocol over Secure Socket Layer）来进行加密传输。这种方式就是为了防止信息被窃取或者拦截后暴露出来。

3.6 本章小结

本章首先讲述了以总体出发设计基于 NETCONF 与 Web 技术的网络管理系统，对该系统进行总体模型的设计。然后对系统内部的两大关键技术难点 NETCONF 以及 Web 技术进行应用研究。第四章将介绍综合网络管理系统的具体实现设计。

第四章 网络管理系统的详细设计与实现

上面一章对综合网络管理系统进行了总体上的设计以及核心技术的研究。这一章开始主要进行系统各个模块详细设计。NETCONF 与 Web 相结合的技术与之前的技术诸如 SNMP^[26]相比既能完成后者的所有功能，又可以很好地弥补后者自身的缺陷，相信在互联网全球化的大背景下，一套基于 NETCONF 与 Web 的系统能够吸引更多新生客户的眼球，打开更大的市场。

4.1 B/S 模型的设计实现

本小节将应用 Web 技术中的 B/S 模型来搭建企业局域网络。这里就得要依靠浏览器以及嵌入设备中的 Web 服务器。通过浏览器请求获取 Web 服务器内的页面以及 javascript 文件，再将用户配置的信息发送给 Web 服务器进行处理。在 Web 接收到 http 请求后，将请求解析并转给相应模块进行动态处理，相应模块会调用数据库来获取设备的配置信息并返回。最后 Web 服务器封装生成响应页面返回给浏览器。这样来实现路由设备的网络配置的管理。B/S 模型如图 4.1 所示。

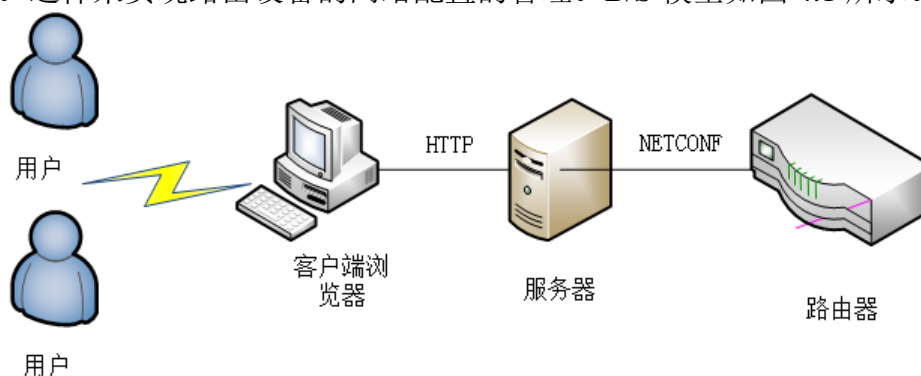


图 4.1B/S 模型框架

4.1.1 Web 浏览器

在图 4.2 里，我们看到模型的入口点是 Web 浏览器，具体的来说分为如下几个步骤：

- 1) 用户在页面表单填写网络配置参数；
- 2) 页面通过 JavaScript 动态包装成 NETCONF 格式的数据报文传给 Ajax；
- 3) Ajax 封装 NETCONF 数据并通过 HTTP 传输发送到目的服务器；

下面针对 B/S 模型的具体实现进行研究设计。在页面层，JavaScript 对表单的元素和值对存放入指定对象，在用户点击提交后，JavaScript 将按照该对象进行 NETCONF 内容层的自动代码生成。然后加上操作层，rpc 层。最后通过 \$.ajax(opt) 传送给后台。这里的 opt 包含了 type, url, data, complete, success 和 failed。具体如图 4.2 所示。

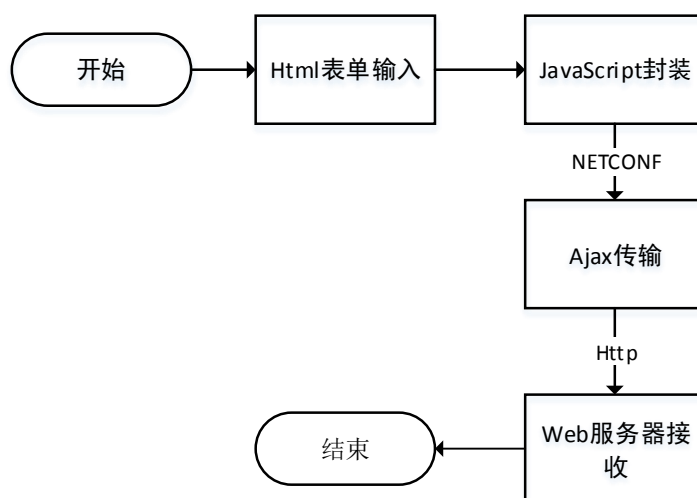


图 4.2 Web 浏览器请求发送图

在 JavaScript 进行自动生成 NETCONF 格式报文这个过程中，我们先在页面将用户配置的数据以对象形式进行以键值对的形式存储。然后再交给 NETCONF 包装部分，通过 for-in 操作对数据对象键--值拆分放入 NETCONF 的内容层标签和标签值中。例如：

参数对象为：

```
var object = {
    "Table": "Interfaces",
    "Row": "Interface",
    "Column": {"IfName", "PortIndex", "Ipv4Address", "Ipv4Mask"}
};
```

通过 JavaScript 的 DOM 选择器选择 DOM^[15]对象并取出对象的输入值，放入合适的对象 Column 内。结果就是：

```
object = {
    "Table": "Interfaces",
    "Row": "Interface",
    "Column": {"IfName": "GE1/0/1", "PortIndex": "254", "Ipv4Address": "10.153.17.188", "Ipv4Mask": "255.255.255.0"}
};
```

下一步就是将这个对象进行解析，并将键值对放入 XML 格式的 NETCONF 上去。具体的代码如下。

```
var opt ,aXML;
for(var sIndex in object)
```



```

{
    if(sIndex != "Column")
        aXML.push('<'+sIndex+'>');
    else
    {
        for(var sSubIndex in object.Column)
        {
            aXML.push('<'+ sSubIndex + '>'+ object.Column[sSubIndex]+'</'+
                sSubIndex+'>');
        }
    }
}
for(var s in object)
{
    if(s != "Column") aXML.push('</'+s+'>');
}
opt.paras={ };
opt.paras.xml = '<rpc message-id="211" xmlns="urn:ietf:params:xml:ns:
NETCONF:base:1.0"><get><filter type="subtree"><top xmlns=" http://
example.NETCONF.com/NETCONF/data:1.0 ">'+aXML.join("")+'</top>
</filter></get></rpc>';

```

这样就把对象 object 内的数据放入到了 opt.paras 里面，而这是依靠的 Json 技术将对象进行存储，键名为 XML，值为 NETCONF 格式字符串。接下来就是通过 Ajax 进行传输了。

```

var opt={
    dataType: "json",
    type:"POST",
    url:"http://example.com",
    data:option?option.paras:null,
    success:onSuccess,
    complete:onComplete
};
$.ajax(opt);

```

这样系统前端就通过 Ajax 把报文传过去了。以上一系列的代码步骤就是用户

配置完成后前端代码块所做的处理。接下来要设计实现的是 Web 服务器。

4.1.2 Web 服务器

Web 服务器是 Web 的核心模块，里面包含了呈现给用户的前端页面代码（主要由 html+css+JavaScript 组成）^[14]，还有服务器端代码（如 PHP）。Web 服务器的作用主要是用来接收客户端发送的 HTTP 请求并进行调度到相应管理逻辑模块如下图 4.3 所示的 NETCONF 模块，最后还有返回信息给客户端。

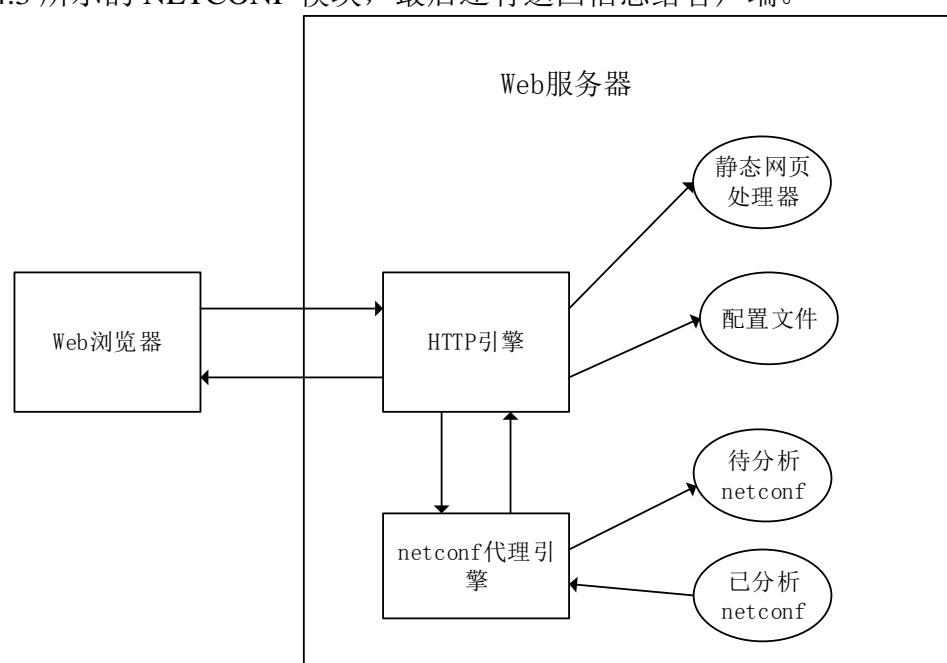


图 4.3 Web 服务器工作模块

在上图中，Web 服务器接受 Web 浏览器发送过来的包含有 netconf 数据的 http 报文。然后进行如下几个步骤。

- 1) Web 服务器内的 HTTP 引擎解析 http 报文，并将解析后的 formdata 内 XML 内容传给 netconf 代理引擎模块。
- 2) NETCONF 代理接收 XML 内容进行解析处理并返回响应 XML 数据。
- 3) HTTP 引擎接收 netconf 代理器返回的 XML 数据，放入 http 报文的 response 中通过 http 返回给浏览器。

4.2 NETCONF 代理处理器实现

本小节重点对 NETCONF 代理器工作原理以及实现机制进行深入分析，这个是本系统的核心模块，也是本篇论文最重要的设计模块部分。

4.2.1 NETCONF 代理总体结构

NETCONF 代理运行在被管理的网络设备如路由器上，主要负责解析 NETCONF 请求报文和生成 NETCONF 响应信息，并修改和过滤网络设备的管理信

息,之后将修改过滤之后的信息发送给Web服务器。下面主要介绍NETCONF代理的结构实现,如下图4.4所示,该模块包含三个层次,下面分别针对管理对象层,操作管理层,RPC消息管理层进行分析研究及其技术实现。

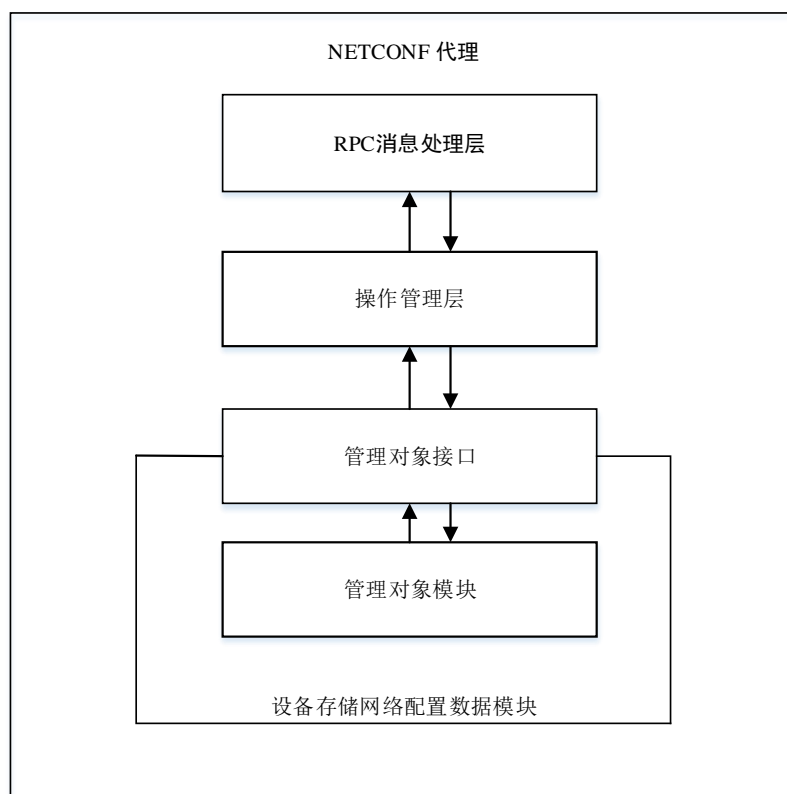


图 4.4 NETCONF 代理三层次

管理对象层与NETCONF模型中的内容层相呼应,它主要负责存储处理网络设备的被管理信息。NETCONF的管理对象层将请求报文中内容层的被操作对象信息通过管理对象接口与设备内的被管理对象模块对应,然后完成对设备配置的实际管理和操作。在进行配置对象管理的过程中必须遵循NETCONF数据模型的数据标准,否则NETCONF代理将无法对设备管理对象模块进行配置操作。并且会生成错误标签作为返回报文。这里的管理对象接口是负责与设备的被管理对象模块进行连接处理的。

操作管理层与NETCONF模型中的操作层相呼应,主要用来接收来自RPC消息处理层的信息(操作参数),并将其解析生成被管理对象的配置操作指令。前提也是该被管理对象模块支持这种操作指令,如果不支持就会返回错误报文。同时操作管理层还负责将管理对象层传递过来的配置操作结果返回给RPC消息层。

RPC消息层则与NETCONF模型中的RPC层相呼应,主要负责解析HTTP代理处理器传递来的NETCONF报文请求信息,然后对RPC报文的结构进行验证,当验证达到相应要求时,就会传送给RPC层,然后RPC层响应接收到的报文,并把响应信息

发送给服务器端。

NETCONF采用独立的模块实现管理系统，带来了简单透明有条理的架构，这样在后期维护扩展上可以有大大的裨益，比如当添加新的管理信息时，NETCONF代理的RPC消息层，操作管理层模块程序是固定不变的，不需要改变，只需要修改管理对象层就可以了，即增加对应的管理对象模块程序。

4.2.2 NETCONF 代理工作流程

NETCONF代理的工作流程如图4.5所示。NETCONF接收NETCONF请求消息后，进行关键字段的解析，当关键字解析完毕之后，NETCONF代理就会根据解析出来的操作类型进行配置处理，当配置正确并允许操作执行时，就会进行<rpc-reply>封装处理，否则就要进行<error-errmsg>消息元素的封装和写入<rpc-error>的操作，直到信息配置正确为。最后以封装成功为前提，进行NETCONF响应信息处理生成。

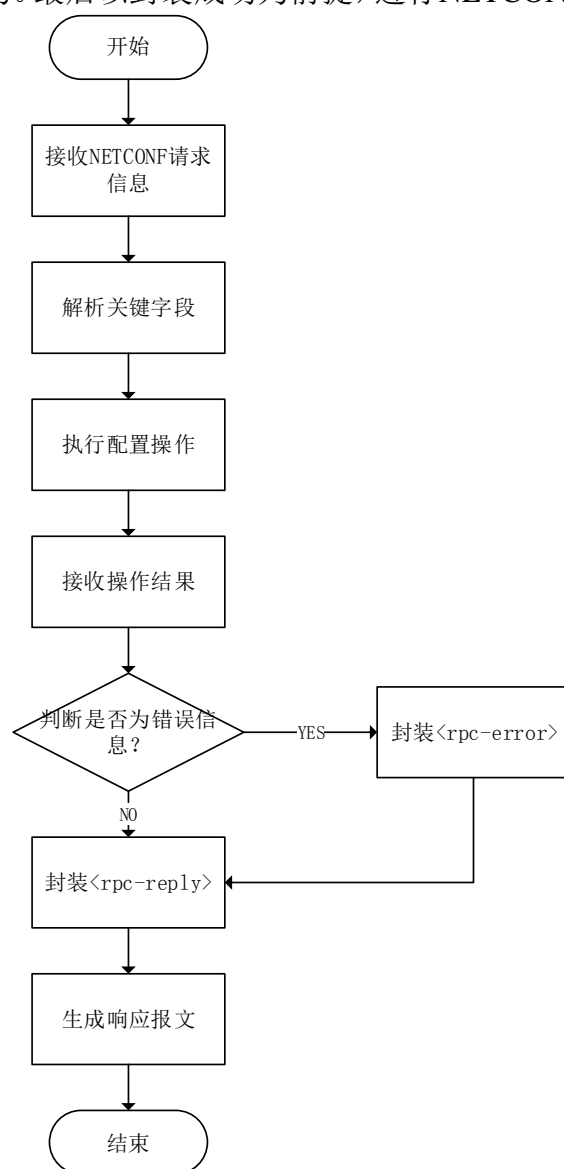


图 4.5 NETCONF 代理层工作流程图

具体的 NETCONF 请求以及响应报文信息的代码于附录中展示。下面是一个典型的 NETCONF 请求报文以及相应的响应报文信息。

请求消息：

```
<!--消息编号-->
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:NETCONF:base:1.0" >
  <get>
    <!--操作对象-->
  </get>
</rpc>
```

响应信息：

```
<!--消息编号-->
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:NETCONF:base:1.0" >
  <top>
    <!--操作结果-->
  </top>
</rpc-reply>
```

如上述表示，这是发送正确的报文，它取得响应报文也是对应在<rpc-reply>中。其中 NETCONF 文档会对请求报文进行封装，且该 NETCONF 文档必须以(rpc)为根节点。如上面请求消息所示，包括信息编号和配置获取操作以及对应的操作对象，分别由<message-id>,<get>,<top>标签来表示。针对请求报文进行解析之后会得到相应的关键字段指导之后的操作配置过程。首先 NETCONF 代理要进行的是获取配置的操作，使用关键字段<get>来实现，具体过程如下面代码所示,当配置操作被正确执行之后就应该得到相应的正确的结果。

```
<Ifmgr>
  <Ports>
    <Port>
      <PortIndex>253</PortIndex>
      <Name>Ethenet1/0/1</Name>
      <IfIndex>253</IfIndex>
    </Port>
    <Port>
      <PortIndex>254</PortIndex>
      <Name>Ethenet1/0/2</Name>
```

```

        <IfIndex>254</IfIndex>
    </Port>
</Ports>
</Ifmgr>

```

然后将返回对象数据放入<rpc-reply>中,即生成了上面响应的数据消息。否则会由于超时而导致错误的出现,如下面所示:

```
<rpc-reply message-id="101"><error errmsg="Request timed out"/></rpc-reply>
```

4.2.3 NETCONF 代理解析设计

根据以上对 NETCONF 代理器的总体架构和工作流程设计,我们可以应用 C++ 程序来实现一个 RPC 类。这个类表示 NETCONF 代理器完成接收处理消息的程序主体功能。其声明如下:

```

class RPC
{
public:
    RPC();
    virtual ~RPC();
    void Excute(char* paras, char* XMLRes);
    char* editNetconf(string& netmsg); //解析 NETCONF 报文
    string GetOP; //获取操作类型
private:
    string msgid; //消息编号
    string op; //操作类型
};

```

其中 msgid 为请求 NETCONF 报文 ID,一对请求和响应报文的 ID 相同;op 表示 NETCONF 请求的操作类型,op 可以表示 get、edit-config 等;函数 editNetconf 负责读入并解析 NETCONF 请求报文;函数 Excute 负责根据函数 editNetconf 解析的结果对设备进行处理并返回 NETCONF 响应报文。

本文采用的 NETCONF 解析处理器是 TinyXML,这款 XML 解析器是一款轻量级的 XML 解析生成处理软件。根据 DOM 模型标准对传入的 NETCONF 文件转换成树形的数据结构文件,利用 DOM 元素处理机制进行 XML 文件的自动生成和解析。用户在使用 TinyXML 时首先调用 tinyXML.h 头文件。

4.2.4 NETCONF 数据模型

在设计不同管理对象的 NETCONF 文档时,设计者必须知道 XML 文档的规范和限制,比如在运用 NETCONF 报文配置某一个对象数据配置时,设计者就必须知道对应的唯一的索引项,不然无法准确的查找到。同时设计者还必须知道内容层每

列数据的数据类型以及范围限制，这样才能对数据进行规范，防止设备报错。这个时候我们就需要一种NETCONF模型，目前有很多种数据模型，如XSD,Yang等等。本文运用的是前者也就是XSD数据模型。

XSD 是指 XML 结构定义 (XML Schemas Definition)，它是 DTD 的替代品。XML Schema 语言也就是 XSD^[27]。XML Schema 描述了 XML 文档的结构，在设备的数据库中存放的设备配置信息结构与 XSD 模型的结构一致^[28]。通过 XSD 我们可以验证 XML 请求文档是否符合配置结构要求。XML Schema 本身是一个 XML 文档，它符合 XML 语法结构，可以用通用的 XML 解析器解析它。

一个 XML Schema 会定义：XML 文件中的元素、XML 文件中的属性、元素下的子元素数与顺序、XML 文件元素以及属性的类型与默认输入。

XML Schema 的优点有：

- 1) XML Schema 基于 XML,没有专门的语法
- 2) XML 可以象其他 XML 文件一样解析和处理
- 3) XML Schema 支持一系列的数据类型(int、float、Boolean、date 等)
- 4) XML Schema 提供可扩充的数据模型。
- 5) XML Schema 支持综合命名空间
- 6) XML Schema 支持属性组。

下面有个实例来说明 XML 文档和 XSD 文档的关系，以及如何运用 XSD 模型来约束规范化 NETCONF 文档的开发。

```
<?xml version='1.0'?>
<bookstore XMLns = "schema.xsd">
  <book genre="autobiography" ISBN="1-861003-11-0">
    <title>The Autobiography of Benjamin Franklin</title>
    <author>
      <first-name>Benjamin</first-name>
      <last-name>Franklin</last-name>
    </author>
    <price>8.99</price>
  </book>
  <book genre="novel" ISBN="0-201-63361-2">
    <title>The Confidence Man</title>
    <author>
      <first-name>Herman</first-name>
```

```
        <last-name>Melville</last-name>
      </author>
      <price>11.99</price>
    </book>
    <book genre="philosophy" ISBN="1-861001-57-6">
      <title>The Gorgias</title>
      <author>
        <first-name>Sidas</first-name>
        <last-name>Plato</last-name>
      </author>
      <price>9.99</price>
    </book>
  </bookstore>
```

上面是 XML 文档，每一内容数据项都是根据 XSD 模型的规范进行设计的，下面就是相应的 XSD 模型文档。

```
<!-- schema.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="schema.xsd"
  elementFormDefault="qualified"
  targetNamespace="schema.xsd">
  <xsd:element name="bookstore" type="bookstoreType"/>
  <xsd:complexType name="bookstoreType">
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="book" type="bookType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="bookType">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="author" type="authorName"/>
      <xsd:element name="price" type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="genre" type="xsd:string"/>
    <xsd:attribute name="publicationdate" type="xsd:string"/>
  </xsd:complexType>
</xsd:schema>
```



```
<xsd:attribute name="ISBN" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="authorName">
  <xsd:sequence>
    <xsd:element name="first-name" type="xsd:string"/>
    <xsd:element name="last-name" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

在这个XSD模型文档中，清晰的结构体系让用户和计算机都可以直接读懂，而且这种形式也符合XML的规范。有了XSD模型文档，我们就可以对NETCONF文档内容格式进行相应的定义以及规范，而且这也有利于减少后台接收的报文格式上的错误而导致设备效率变低。

4.3 DOM 转换器与 SNMP 处理器的实现

在上一节本文详细阐述了NETCONF代理的设计与实现，可以通过NETCONF代理来对NETCONF的请求进行处理并返回响应报文，这里有个前提是，当前设备支持NETCONF管理，拥有相关的NETCONF管理配置接口。但是以目前所知，还有很多企业级的网络设备没有这项新功能，仍然运用着SNMP代理来进行网络上的配置管理。所以，本系统也必须满足这一部分网络设备的需求，这样的环境下，本文就对NETCONF代理模块与设备模块之间的层设计添加两个模块：DOM转换模块以及SNMP代理模块。这两个模块一个是用来对NETCONF数据进行转换，转换为SNMP形式报文，另外一个是用来接收SNMP请求来进行相应的设备配置的管理并生成SNMP响应报文返回DOM转换器。

4.3.1 DOM 转换模块

DOM转换器的原理是应用基于DOM元素查找更换方案来进行数据格式间的转换，这样转换器就能实现NETCONF和SNMP数据格式的兼容性转换。如图4.6所示，DOM转换器以NETCONF代理器解析的NETCONF数据报文作为输入，以SNMP格式的数据形式作为输出传给SNMP处理器，其中中间的过程就是DOM转换器做的转换工作。同时，当SNMP代理传给DOM转换器时，SNMP格式的数据会在DOM转换器中转换成为NETCONF格式的数据对象，再将这对象传给NETCONF代理器进行NETCONF报文的生成。图4.6中包括NETCONF中的DOM元素和其元素值，表示出了DOM是如何实现转换的，他们分别用圆圈和方框来表示。DOM转换器的详细工作流程如下：

第一步，DOM转换器收到NETCONF代理提供的NETCONF配置数据，解析配置数据的各个元素关系并在内存中创建对应元素关系的DOM结构树，并在初始化时将DOM结构树上的元素节点的文本值进行清空处理。

第二步，获取节点SNMP.OID值^[30]，并获取对应的SNMP操作。获取节点SNMP.OID时，是通过NETCONF报文中的内容节点与DOM结构树的目标节点进行对应得到的此值。获取SNMP操作过程中主要的依据是来着NETCONF-SNMP协议中涉及到的操作原语的对应关系。最后将获取的SNMP对象节点、SNMP操作类型和SNMP代理IP地址发送给SNMP处理器。

第三步，准备接收返回的SNMP数据响应报文，并检测SNMP数据报文的响应结果。如果设备接收SNMP并成功执行，则进入第四步；否则停止DOM转换器内NETCONF向SNMP数据格式的转化，并将错误信息封装成NETCONF响应报文依据下面介绍的第五步来进行处理。

第四步，对于SNMP报文操作成功的情况，DOM根据返回的报文结果修改NETCONF内DOM元素节点的值。同时平行寻找是否存在其他操作对象，如果存在的话，跳到第二步；如果不存在的话，DOM转换器就会依据DOM结构树更新的结果值转换成NETCONF的格式，跳到第五步。

第五步，DOM转换器把NETCONF格式的响应结果报文返回给NETCONF代理。

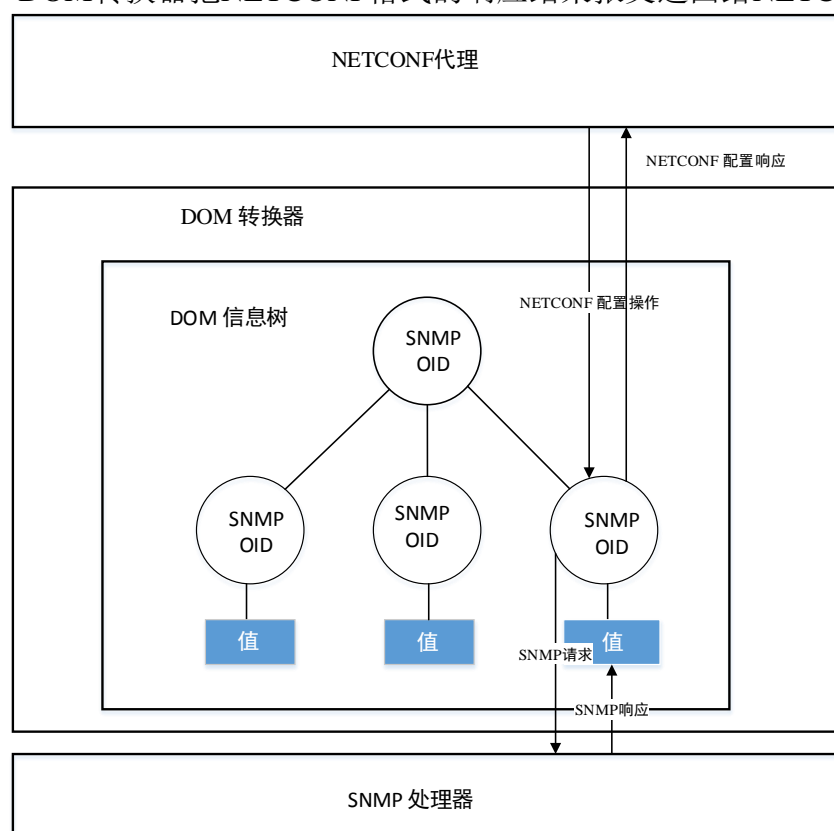


图 4.6 DOM 转换器工作原理图

4.3.2 SNMP 处理器

在典型的SNMP管理模式中，存在一个或多个管理计算机，称为管理者，管理者具有监视以及管理一组主机或一个计算机网络。在每个被管理的系统上都会装载它的SNMP管理代理软件，管理者通过向代理发送请求来获取或者配置被管理的计算机对象。

SNMP代理是网络设备中的一个应用模块，用于维护被管理设备的信息数据并响应管理者的请求，把管理数据汇报给发送请求的管理者。同时，当设备发生故障或者其他事件的时候，代理会主动发送Trap信息给管理者，通知设备当前的状态变化。通过SNMP访问的变量是有组织层次结构的，这些层次结构和其他元数据（例如类型和变量的说明）由管理信息库（MIB）进行管理。

一个SNMP管理的网络有三个主要组成部分：

- 1) 被管理设备；
- 2) 代理(运行在被管理设备的软件)；
- 3) 网络管理站（运行于管理设备上的软件）；

NMS、代理和被管理设备（MIB）之间的关系如图 4.7 所示。被管理的设备拥有个实现 SNMP 的接口，其允许单向网络节点（只读）或双向（读写）访问节点显示特定信息。管理设备会通过 NMS 来交换节点上特定的信息，这些特殊节点有时也被称为网络元素，被管理设备可以是任何类型的设备，包括路由器，接入服务器，交换机，网桥，集线器，IP 电话，IP 视频摄像机，计算机主机和打印机等等。

作为最主要的软件模块，代理具有管理信息的本地知识和接收NMS发送的操作请求并对MIB节点信息进行修改和配置的功能。

网络管理站（NMS），执行监视和控制管理设备的应用程序。网络管理站提供大量需要处理和存储的网络管理资源^[29]。被管理网络上可能存在多个网管系统进行着监控。

SNMP^[30]本身并没有完全去定义网络管理信息（变量）来描述和管理对象。SNMP是使用一种可扩展的设计，其中所说的那些可用的信息由管理信息库（MIB）来定义。MIB使用分层命名空间内的对象标识符（OID）来描述子系统内部的管理数据的结构。每个OID标识，可以通过SNMP读取或设置一个变量。

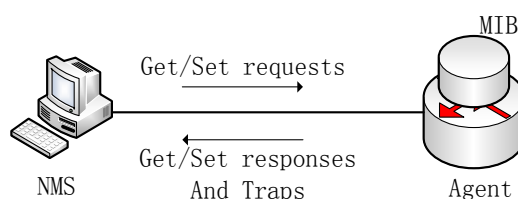


图 4.7NMS、Agent 和 MIB 关系图

MIB 是以树状结构进行存储的。结构树的节点代表被管理对象，在对被管理对象进行管理的过程中，MIB 通过树的节点路径进行识别，这个节点路径就是 OID。它是一种管理识别节点唯一的标志，需要注意的是 OID 是从根节点开始记录的。如图 4.8 所示，例如对 private 节点进行管理的时候，OID 为{1,3,6,1,4}。若要对 ip 进行管理，则查找的 OID 为{1,3,6,1,2,1,4}。

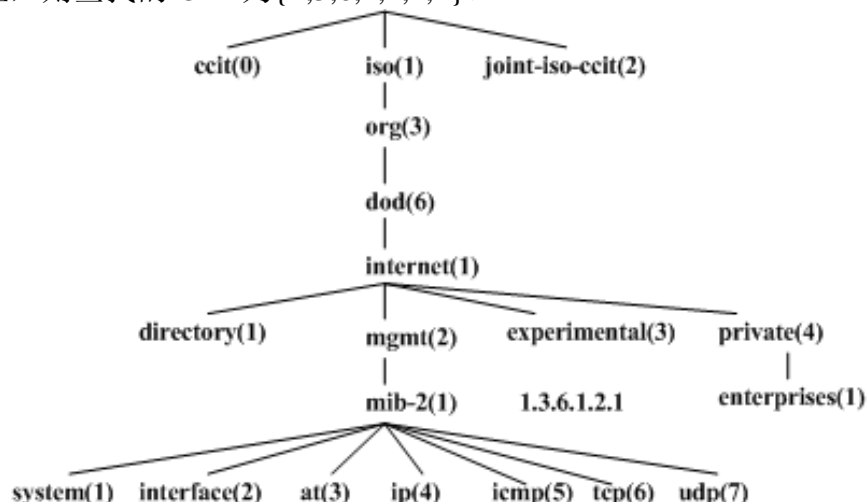


图 4.8 MIB 树结构示意图

一个视图的范围可以通过 MIB 结构子树的掩码和其子树 OID 来共同确定。其中用十六进制来表示掩码，而且该十六进制是需要转换才能知道所有的节点信息，即转换为二进制后用每个比特来表示节点。在二进制表示中，如有访问节点与树节点值相等则必须用 1 表示，否则要用 0 来表示。如下图 4.9 所示，子树的掩码为 0xDB，将其转换为二进制格式为 11011011，子树 OID 为 1,3,6,1,6,1,2,1，则所确定的子树视图范围就是 1,3,*1,6,*2,1(*表示任意的数字)。

子树OID	1	3	6	1	6	1	2	1
子树掩码	1	1	0	1	1	0	1	1

图 4.9 子树 OID 与子树掩码对应关系图

4.4 本章小结

本章我们对这套基于NETCONF的Web操作系统进行了全面的实现。根据上一章的总体架构设计我们分别对内部的各个模块进行了详细的设计与实现。同时对支持NETCONF和还未支持NETCONF的设备进行了兼容性的设计，完成了对市场各式路由设备的支持。在下一章的研究中，我们将用H3C的M9000新一代路由器进行实例测试分析。因为这个设备全面支持NETCONF功能，所以我们就不需要再

进行DOM转换成SNMP形式。

第五章 网络管理系统测试与分析

上一章详细阐述了这套系统的设计实现方案，并对各个功能模块进行了详细的分析和研究。目前对SNMP代理系统的测试分析已经有很多人做过，因此针对这个，本人就只对基于NETCONF的Web操作系统进行测试。下面就将对这套系统进行详细的测试分析。

5.1 运行环境的搭建

由于公司的路由器里面包含了一整套服务器配置，包括Web服务器，NETCONF代理，以及telnet远程登录的协议接口。因此在对这套系统进行测试的时候，就不需要另外添加一台PC作为服务器了。本人根据这个特性搭建了一套如图5.1所示的基于NETCONF的综合网络操作系统。在实习公司的实际情况下，网络环境主要由一台路由器，一台直连路由器的计算机，还有通过浏览器Web登陆服务器的客户端PC组成。

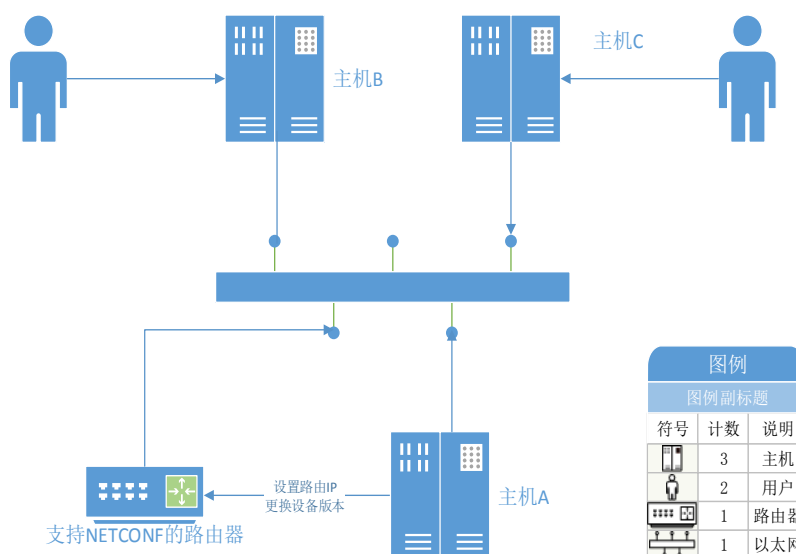


图 5.1 系统运行环境

在上图所示的运行环境中，各个主机以及路由器的具体配置如下。

- 1) 主机A、B、C：台式机HP 8380 Core i7-3770 3.4G/4G/500G/DVD/双网卡/19寸WLCD显示器。
- 2) 路由器：H3C SecPath M9000，双主控板/四业务板/主控引擎1+1冗余/支持多业务安全网关的多核全分布式架构。

在软件环境上前端开发环境是Win7+Linux，页面开发工具是Subline+svn。Web服务器选用的是Apache服务器，来执行HTTP报文收发服务和控制转发的功能服务。同时该路由器也包含NETCONF代理，用来对设备网络配置参数进行读取以及

NETCONF响应报文的自动生成。我们在NETCONF代理的选择上选择的是基于DOM解析的XML C++语言解析器CMarkup，而且主要是基于压缩文档对象模型(EDOM: Encapsulated Document Object Model)的，不仅具有DOM的优势和特点，而且达到了吸引力-简单的优势，仅仅通过一个XML对象。

上图的主机A是用来对路由器进行软件版本的更换和配置，以及IP地址的配置。同时主机B和C也可以通过远程登陆主机A来代替A做这些工作。

主机B跟C是通过浏览器访问路由器设备的Web服务。通过浏览器URL地址来访问系统，输入用户名和密码后进入主页面，然后就可以通过配置参数和选项来进行网络搭建和配置。

5.2 网络管理系统测试分析

本文的测试方案将对设备路由的IPv4模块进行测试。该测试是从两个方面来入手的。首先对系统进行get操作的测试，通过页面请求获取设备配置信息并显示到页面列表。并与SecureCRT远程登陆设备命令行获取的进行比较来验证get操作的准确性。

第二方面是对系统进行edit-config操作的测试，其中包括原语操作之上封装的action操作以及config操作。通过页面请求发送配置设置信息并对设备进行配置上的修改，然后通过SecureCRT远程登录设备获取设置完成后的设备配置信息。

本文的测试工具选用的是Chrome浏览器内置的开发者工具Developer Tools。该工具页面视图如下图5.3所示。这是一款包含HTTP请求响应报文获取，页面元素代码查询修改，控制台脚本调试等功能的开发测试工具。利用此工具可以很好的完成网络管理系统get/config操作的测试工作。

现在我们通过Web利用浏览器来测试路由设备IPv4管理模块的管理功能。IPv4管理页面如下图5.2所示。

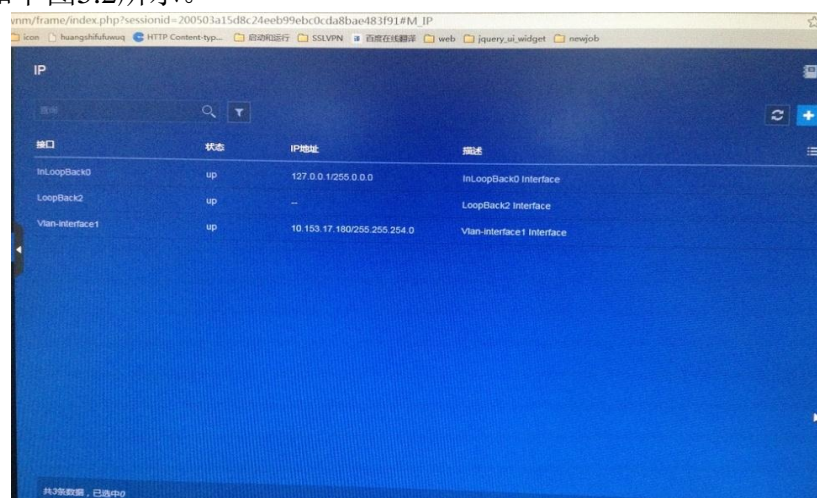


图 5.2 IPv4 管理模块 Web 页面

5.2.1 get 操作测试

在页面进入的时候，html与JavaScript会有初始化的过程，在这个初始化的过程中，我们调用NETCONF的get操作来获取我们所需要的接口ipv4信息。然后将这些信息放入页面的列表对象中显示出来。本系统通过Chrome浏览器内置的Developer Tool里面的Web查看请求报文头。在HTTPS请求报文的主体Form Data内有个XML键值对，这包含了发送给后台的NETCONF请求数据。

测试输入：

```
<rpc message-id="101"
  XMLNs="urn:ietf:params:XML:ns:NETCONF:base:1.0" >
  <get >
    <filter type="subtree">
      <top XMLNs="http://example.NETCONF.com/NETCONF/data:1.0" >
        <Ifmgr>
          <Interfaces>
            <Interface>
              <IfIndex></IfIndex><!--接口的索引-->
              <Name></Name>
              <PortIndex></PortIndex>
              <ifTypeExt></ifTypeExt><!--接口的类型-->
              <OperStatus></OperStatus>
              <ConfigSpeed></ConfigSpeed>
              <InetAddressIPv4><InetAddressIPv4><!--获取接口 Ipv4-->
              <InetAddressIPv4Mask></InetAddressIPv4Mask>
            </Interface>
          </Interfaces>
          <InterfaceCapabilities><!--获取接口能力集-->
            <Interface>
              <IfIndex/>
              <Configurable/>
              <MinMTU/>
              <MaxMTU/>
              <Removable/><!--获取接口是否可删除-->
              <MaxCreateSubNum/><!--获取可建子接口个数-->
            </Interface>
```

```
</InterfaceCapabilities>
</Ifmgr>
</top>
</filter>
</get >
</rpc>
```

在这个测试输入的NETCONF表中，本系统可以通过Interfaces表下发的IfIndex索引来找到相应的接口，由于IfIndex未指明具体索引，因此本操作将获取所有的接口信息。同时系统前端下发InterfaceCapabilities表来获取所有接口的能力集，有了能力集我们就可以知道接口是否可配，创建子接口的个数，MTU范围以及是否可删除这些属性了。如下图5.3所示，前端向后台发出get请求获取路由设备所有接口数据信息。

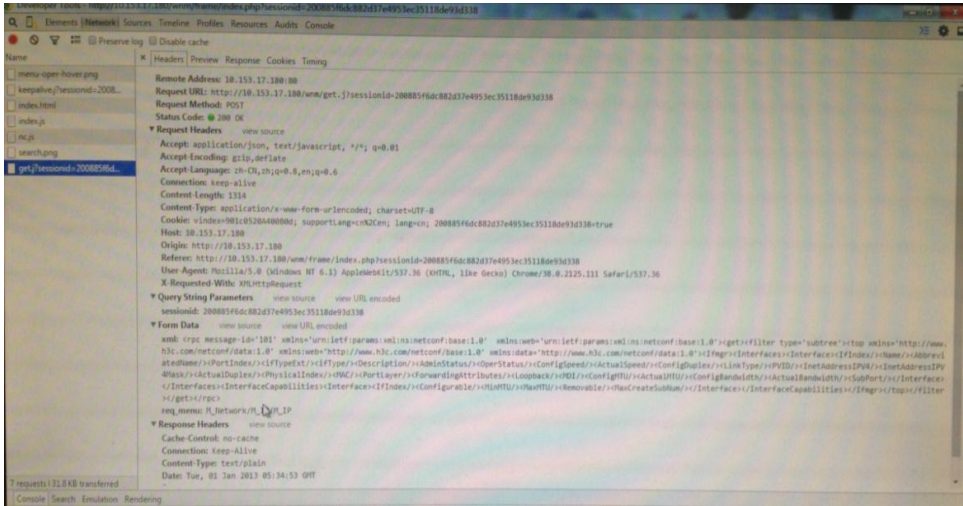


图 5.3 请求报文 NETCONF 数据

测试用例表如下表5.1所示。

表5.1 Ipv4页面get操作测试

测试类型	功能测试
测试目的	用户通过Web在ip页面上完成设备的get操作
测试方法	因果图法
前置条件	1. B/S网络搭建正确； 2. 浏览器正确访问设备服务器页面；
执行步骤	1. 主机A配置路由设备ip地址，开通http/https服务； 2. 客户端浏览器URL访问设备Web站点页面； 3. 用户进入Ipv4模块页面；

	4. 观察初始化页面列表显示的Vlan-Interface1与Vlan-Interface5虚接口Ipv4信息;
预期结果	Web页面获取的Vlan-Interface1和Vlan-Interface5虚接口Ipv4信息与设备实际Ipv4信息相同。

测试页面结果:

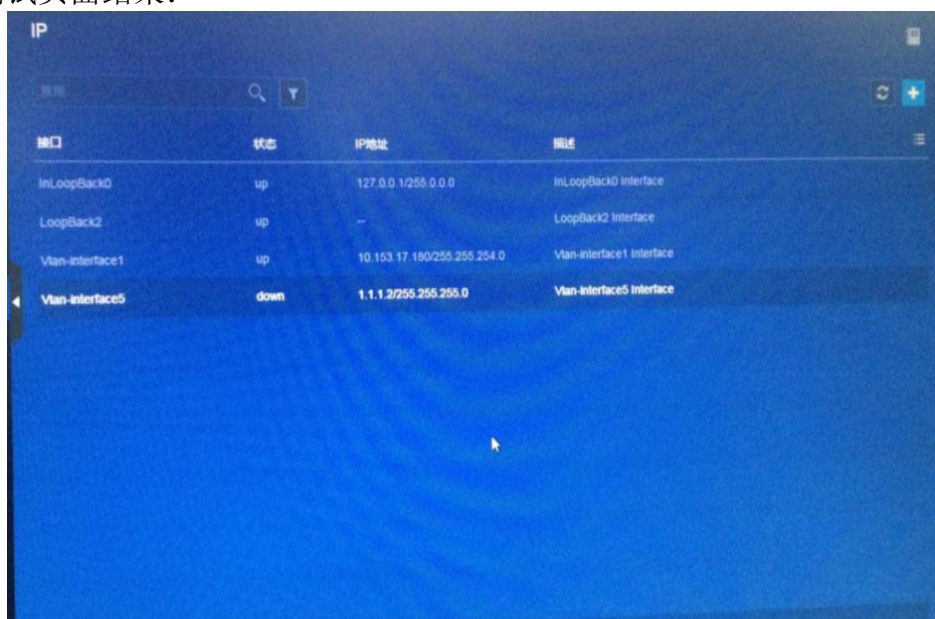


图 5.4 Ipv4 管理页面列表显示

上图 5.4 显示了从设备返回的接口信息,这里返回的数据是通过 json 格式封装的,这是因为 JavaScript 处理解析 json 报文更加简便,不像处理 NETCONF 格式还要进行 DOM 解析。因此我们设备服务器在返回数据的过程中安装了一个 json 代理处理器,它的作用是将响应数据格式转换成 json 格式。数据对象以 json 对象形式返回之后会根据页面需求筛选虚接口的 Ipv4 地址。这是根据接口的 ifTypeExt 属性来过滤的。此时接口列表显示了内部环回口,环回子接口和两个 vlan 虚接口。

这次测试中,我们主要查看 vlan 虚接口的 Ipv4 返回信息,页面中两个接口 get 操作结果如下表 5.2 所示。

表5.2 Ipv4页面get操作测试结果

接口	状态	Ipv4 地址	Ipv4 掩码
Vlan-interface1	up	10.153.17.180	255.255.254.0
Vlan-interface5	down	1.1.1.2	255.255.255.0

测试结果验证:

为了更加准确的表明从设备获取的 vlan 虚接口信息没有错误, 我们用 SecureCRT 这款远程登录软件来通过 Telnet 登录设备系统。然后利用 CLI 命令获取 vlan 虚接口的接口 ip 信息来与页面上的 vlan 虚接口 ip 信息进行对比。

```
<h3c-180>display ip interface vlan-interface
```

```
Vlan-interface1 current state: up
```

```
Line protocol current state:up
```

```
Internet Address is 10.153.17.180/23 primary
```

```
Vlan-interface5 current state: down
```

```
Line protocol current state: down
```

```
Internet Address is 1.1.1.2/24 primary
```

验证结果表明 Web 页面显示结果符合设备接口信息结果。测试验证配置如下图 5.5 所示。

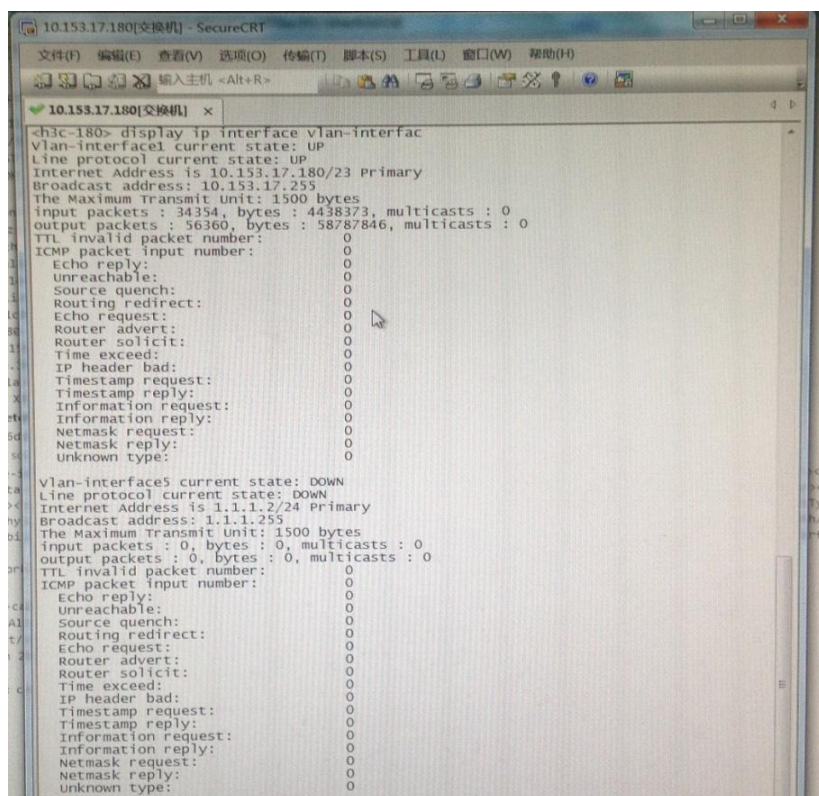


图 5.5 设备 Vlan 虚接口 ip 信息

5.2.2 edit-config 操作测试分析

本节我们将对 IPv4 管理页面的 Loopback6 口进行 IPv4 的设置。首先, 由于设备没有这个虚接口, 我们必须先进行创建。

测试输入:


```
<rpc message-id="101"
  XMLNs="urn:ietf:params:XML:ns:NETCONF:base:1.0" >
  <action>
    <top XMLNs="http://example.NETCONF.com/NETCONF/data:1.0" >
      <Ifmgr>
        <LogicInterfaces>
          <Interface>
            <IfTypeExt>16</IfTypeExt><!--该口的类型-->
            <Number>6</Number><!--子接口的编号-->
          </Interface>
        </LogicInterfaces>
      </Ifmgr>
    </top>
  </action>
</rpc>
```

在这个NETCONF请求报文中，使用edit-config之上封装的action动作操作，这个操作主要是根据内容层的信息做出判断，如果内部包含remove标签就执行删除操作，如果没有这类动作标签就执行默认操作，在这里是执行create操作。报文内容层中的IfTypeExt代表Loopback口的接口类型，Number表示待创建子接口的编号。这样预期生成的结果应该是创建Loopback6口。测试输入的新建弹出框界面如图5.6所示。

图 5.6 Loopback6 接口创建窗口

测试用例表如下表5.3所示。

表5.3 Ipv4页面新建Loopback6操作测试

测试类型	功能测试
------	------

测试目的	用户通过Web在ip页面上完成创建Loopback6子接口操作
测试方法	因果图法
前置条件	<ol style="list-style-type: none"> 1. B/S网络搭建正确; 2. 浏览器正确访问设备服务器页面;
执行步骤	<ol style="list-style-type: none"> 1. 主机A配置路由设备ip地址, 开通http/https服务; 2. 客户端浏览器URL访问设备Web站点页面; 3. 用户进入Ipv4页面; 4. 用户点击新建Loopback口弹出框; 5. 观察设备Loopback6接口创建信息;
预期结果	Web页面创建Loopback6子接口及接口Ipv4信息, 设备执行该操作完成配置。

向后台发送的请求报文如下图5.7所示。

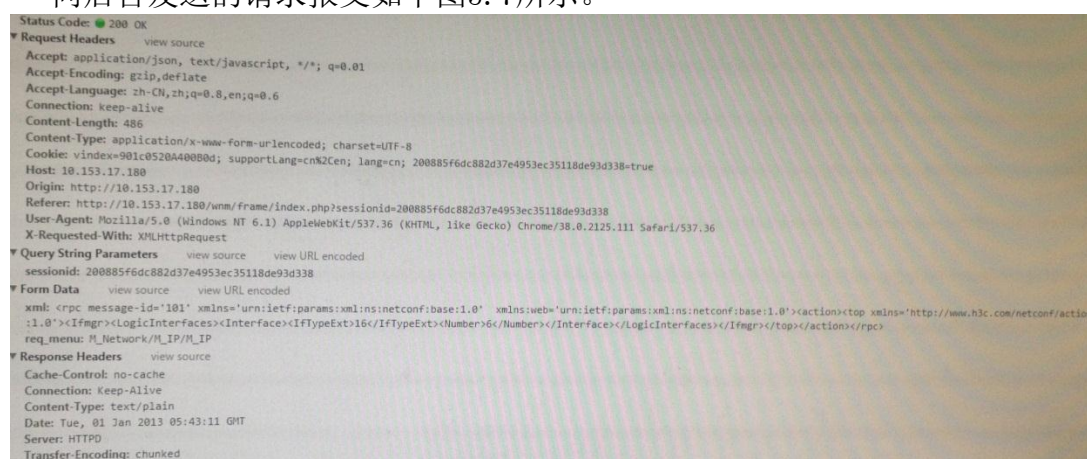


图 5.7 Loopback6 接口创建请求报文

测试结果:

后台返回成功会在Response内显示ok, 如图5.8所示。

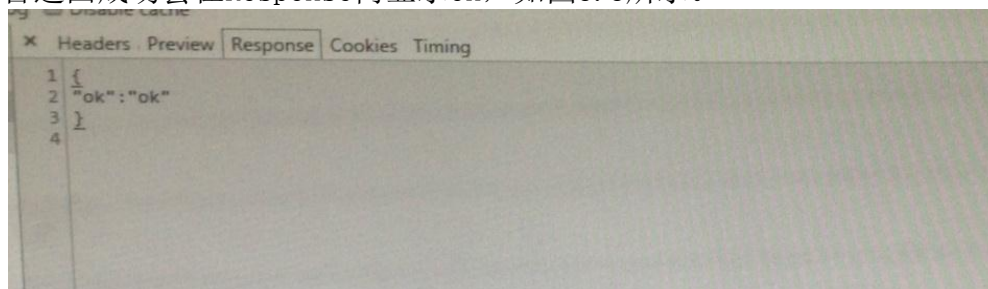


图 5.8 创建接口响应报文

测试结果验证:

为了验证页面新建子接口操作的准确性,我们在页面完成操作后在SecureCRT上输入display ip interface Loopback6检查子接口ip信息是否已经创建成功。

```
<h3c-180>display ip interface Loopback6
```

```
Loopback6 current state: up
```

```
Internet Address is 10.10.10.11/24 Primary
```

测试配置结果如图5.9所示。

```
<h3c-180> display ip interface LoopBack6
LoopBack6 current state: up
Line protocol current state: UP(spoofing)
Internet Address is 10.10.10.11/24 Primary
Broadcast address: 10.10.10.255
The Maximum Transmit Unit: 1536 bytes
input packets : 0, bytes : 0, multicasts : 0
output packets : 0, bytes : 0, multicasts : 0
TTL invalid packet number: 0
ICMP packet input number: 0
  Echo reply: 0
  Unreachable: 0
  Source quench: 0
  Routing redirect: 0
  Echo request: 0
  Router advert: 0
  Router solicit: 0
  Time exceed: 0
  IP header bad: 0
  Timestamp request: 0
  Timestamp reply: 0
  Information request: 0
  Information reply: 0
  Netmask request: 0
  Netmask reply: 0
  unknown type: 0
```

图 5.9 设备 Loopback6 接口信息

成功创建完Loopback6虚接口之后,我们将对此接口进行config操作。首先进入如图5.10所示的编辑页面。我们完成配置项的修改后点击提交,JavaScript就将这个接口要设置修改的属性封装成NETCONF形式,通过http发送给路由器,并对设备进行edit操作。

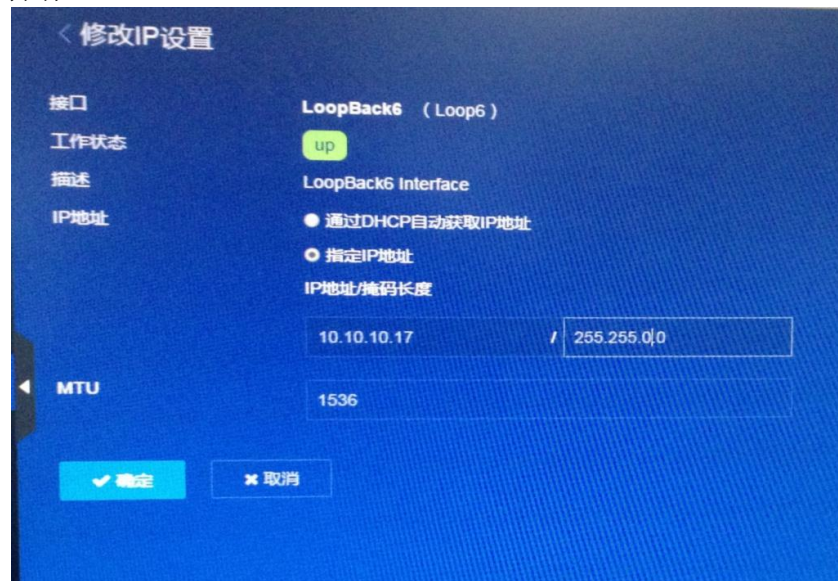


图 5.10IPv4 管理编辑页面

测试输入:

```
<rpc message-id="101" XMLns="urn:ietf:params:XML:ns:NETCONF:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <top XMLns= "http://www.hp.com/NETCONF/config:1.0" Web :
operation = "merge" >
        <IPV4ADDRESS><!--特性名-->
          <Ipv4Addresses><!--表名-->
            <Ipv4Address><!--行名-->
              <IfIndex>911</ IfIndex><!--列索引-->
              <Ipv4Address>10.10.10.17</Ipv4Address>
              <Ipv4Mask>255.255.0.0</Ipv4Mask>
              <AddressOrigin>1</AddressOrigin>
            </Ipv4Address>
          </Ipv4Addresses>
        </IPV4ADDRESS>
      </top>
    </config>
  </edit-config>
</rpc>
```

在这个NETCONF请求报文中，操作类型选用的是设置类型edit-config，主要是对设备进行配置的设置修改。operation='merge'表示如果接口已存在则在该接口上进行配置修改，如果接口不存在则会试图创建新接口，若创建成功则设置接口的配置，若不成功则返回失败。这种operation属性是实习公司对NETCONF原生操作的扩展和改进，这样的结果就是更加简化了报文的封装和解析，从而大大减轻了工作人员和代理引擎的工作负担。下发这样一组操作类型为配置新接口的NETCONF报文，后台会搜索内容层的索引即IfIndex，并根据索引遍历设备已有接口参数，找到接口的后就会将索引为911的Loopback6子接口的Ipv4地址由10.10.10.11变为10.10.10.17。同时Ipv4掩码地址也从255.255.255.0修改为255.255.0.0。设置页面请求发送如下图5.11所示。

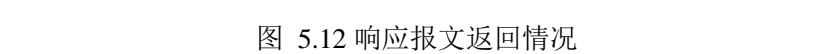


表5.4 Ipv4页面设置Loopback6接口配置操作测试

--	--

测试结果:

请求完成后系统可以通过Chrome浏览器内置的Developer Tool里面的Web response查看该次请求返回报文结果。如果返回ok则配置成功。如图5.12所示。



测试结果验证:

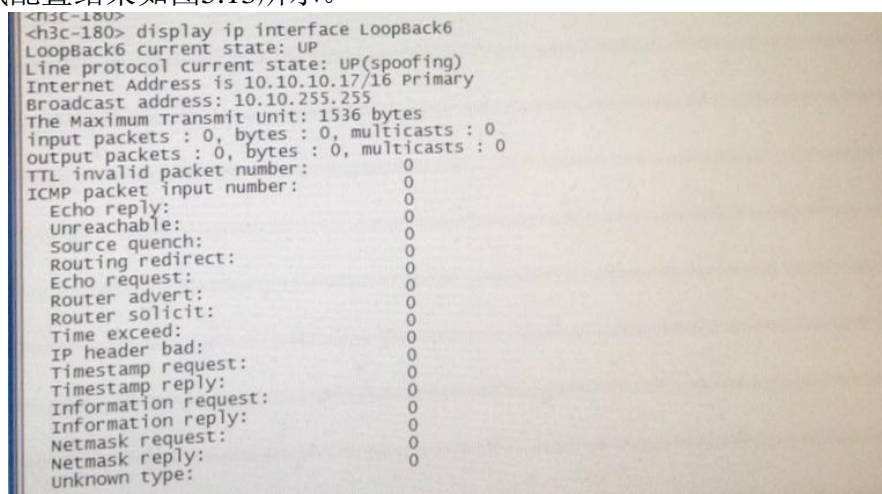
为了验证页面设置Loopback6子接口Ipv4地址操作的准确性, 本文在页面完成操作后在SecureCRT上输入display ip interface Loopback6检查子接口ip信息是否已经修改成功。

```
<h3c-180> display ip interface Loopback6
```

```
Loopback6 current state: up
```

```
Internet Address is 10.10.10.17/16 Primary
```

测试配置结果如图5.13所示。



```
<h3c-180> display ip interface Loopback6
LoopBack6 current state: UP
Line protocol current state: UP(spoofing)
Internet Address is 10.10.10.17/16 Primary
Broadcast address: 10.10.255.255
The Maximum Transmit Unit: 1536 bytes
input packets : 0, bytes : 0, multicasts : 0
output packets : 0, bytes : 0, multicasts : 0
TTL invalid packet number: 0
ICMP packet input number:
  Echo reply: 0
  Unreachable: 0
  Source quench: 0
  Routing redirect: 0
  Echo request: 0
  Router advert: 0
  Router solicit: 0
  Time exceed: 0
  IP header bad: 0
  Timestamp request: 0
  Timestamp reply: 0
  Information request: 0
  Information reply: 0
  Netmask request: 0
  Netmask reply: 0
  unknown type: 0
```

图 5.13 设备 Loopback6 接口信息

以上的测试成功的说明了此系统功能上符合基于NETCONF与Web这样新一代网络管理系统的思想, 并成功的实现了这一技术。

5.3 测试分析小结

本章通过公司路由产品实际环境的检测, 对这套基于NETCONF与Web的网络管理系统的IPv4管理模块进行读取接口IP信息以及设置修改某接口IP地址信息的操作。经过功能上的测试以及与命令行结果的对比显示, 系统的运行结果与实际配置相同, 这套系统可以很好的完成复杂的配置操作以及大量数据的处理。

第六章 结束语

本文通过对NETCONF技术与Web通信技术的深入研究,结合当前市场对路由器网络配置管理系统领域Web化的发展和新需求,提出了以NETCONF数据格式报文为传输报文,以Web模式通信为基础,实现B/S模式的网络架构。同时该系统搭建于各类网络设备中,实现Web与CLI两种网络管理方式共同管理的新模式。

6.1 论文工作总结

纵观全文,本文主要完成了对基于NETCONF与Web的网络管理系统的需求分析、重点技术分析、总体系统设计和功能模块设计实现。并且完成了对这套系统的读取测试。具体包括如下几个方面。

1) 本文从用户交互性和用户体验角度出发,提出了以Ajax这种异步传输技术来对NETCONF数据报文进行传输的方案。利用Ajax机制将NETCONF数据报文包装在http请求主体里,然后通过http协议进行传输。

2) 设计研究了NETCONF代理。NETCONF代理生成器主要负责把Web服务器传送过来的NETCONF请求数据进行解析,从而提取出数据中的操作类型以及操作对象,并通过与设备对象提供的NETCONF操作接口进行参数的配置操作。如果该设备不支持NETCONF接口,则利用DOM转换器将其转换成SNMP格式报文。

3) 对于不支持NETCONF接口的设备设计了DOM转换器,它将从NETCONF代理获取的数据对象转换成相应的SNMP格式数据对象形式,并将转换的格式传送给相应的SNMP处理。

4) 对SNMP管理技术进行了设计研究。它将DOM转换器转换的SNMP格式对象进行设备的网络配置管理。并接收设备返回的响应报文。最后将返回数据传输给DOM转换器处理。

5) 本文还利用M9000路由器设备搭建运行环境,对该Web操作系统进行了get/edit操作的测试分析,结果符合设计要求。

6.2 后续工作展望

由于时间等条件的限制,本文所做的工作还存在一些需要继续改进和探讨的地方,在以下的几个方面进行更进一步的研究和完善:

1) 在完善研究数据模型XSD基础上,我们可以继续研究另一种数据模型YANG。并且进一步来说可以开发YANG的转换工具。目标是实现MIB、XSD与YANG数据模型语言的相互转换。

2) 在实现NETCONF网络管理配置的基础上, 我们可以进一步设计和实现NETCONF/SNMP转换网关的网络管理系统。这样可以管理支持SNMP进行管理网络的设备, 从而实现大型企业代理模式与嵌入式模式系统一体化。

3) 扩展NETCONF操作层的操作语言。在目前八种操作原语的基础上, 进行新的操作动作研究, 并研究嵌套式操作, 如merge操作内对某表在进行remove操作。

参考文献

- [1] 徐慧,肖德宝,陈历淼. 基于NETCONF和Web服务的综合网络管理技术研究与应用[J] 计算机应用研究, 2008,8(4):2537-2540.
- [2] 刘会芬.基于NETCONF的网络管理者的研究与实现[D],武汉:华中师范大学论文,2008.
- [3] KLIET, STRAUB F. Integrating SNMP agents with XML-based management systems[J]. IEEE Communications Magazine, 2004, 42(7) :76- 83.
- [4] 王聪.NETCONF数据模型与NETCONF/SNMP转换网关[D],湖南:中南大学,2009.
- [5] R.Enns. NETCONF Configuration Protocol[EB]. [2006] <http://www.ietf.org/rfc/rfc4741.txt>
- [6] E. Lear .Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP) [EB][2006]. <http://www.ietf.org/rfc/rfc4744.txt>.
- [7] E. Lear .An Evaluation Framework for Data Modeling Languages in Network Management Domain[EB][2006]. [http:// www.ietf.org/internet-drafts/draft-xiao-evaluate-dml-01.txt](http://www.ietf.org/internet-drafts/draft-xiao-evaluate-dml-01.txt).
- [8] 魏梅娟.下一代网络管理系统中基于XML管理者的研究与实现[D],武汉:华中师范大学, 2007.
- [9] 刘进.SNMP/XML网关的研究[D],武汉:华中师范大学,2007.
- [10] R. Fielding. Hypertext Transfer Protocol -- HTTP/1.1[EB]. [1999] [http://www.w3.org/Protocols /rfc2616/rfc2616.txt](http://www.w3.org/Protocols/rfc2616/rfc2616.txt)
- [11] Ryan Asleson . Foundations Of Ajax[M].金灵译.北京:人民邮电出版社,2006:50-100.
- [12] 叶锋.基于Web和XML的网络管理机制研究与实现[D],武汉:武汉科技大学,2004.
- [13] W.Richard Stevens.TCP/IP详解卷1:协议[M].北京:机械工业出版社.2002:70-292.
- [14] Nicolas C.Zakas. Javascript 高级程序设计(第三版)[M]. 北京:人民邮电出版社.2012:60-200.
- [15] W3C.Document Object Model(DOM)Level1 Specification[EB]. [2004].[http://www.w3.org /TR/REC-DOM-LEVEL-1/](http://www.w3.org/TR/REC-DOM-LEVEL-1/).
- [16] 章勋,杨家海,王继龙. 基于XML技术的网络配置管理系统[J].计算机工程,2008,34 (03):127-129.
- [17] Sandeep Adwankar. NETCONF Data Model[EB]. [2004].[http://tools.ietf.org/id/draft-adwankar -NETCONF-datamodel-01.txt](http://tools.ietf.org/id/draft-adwankar-NETCONF-datamodel-01.txt).
- [18] Tail-f Systems Instant NETCONF Agent[EB]. [2009] [http://www.tail-f.com/products/instant NETCONF-agent/](http://www.tail-f.com/products/instant-NETCONF-agent/).
- [19] Sandeep Adwankar, Sharon Chisholm. Using XML Schema to define NETCONF Content [EB] .[2007].<http://tools.ietf.org/id/draft-chisholm-NETCONF-model-06.txt>.
- [20] T. Goddard. Using NETCONF Over the Simple Object Access Protocol (SOAP)[EB]. [2006].

<http://www.ietf.org/rfc/rfc4743.txt>.

- [21] R. Enns, Ed.NETCONF Configuration Protocol[EB].[2006]. <http://www.rfc-editor.org/rfc/rfc4741.txt>
- [22] 刘萍,肖德宝. 基于XML的网络管理模型研究[J].计算机工程与应用, 2004,40(21):153-156.
- [23] Richard Froom,Balaji Sivasubramanian,Erum Frahim. CCNP SWITCH(642.-813)[M] 北京:人民邮电出版社,2011:160-180.
- [24] 赵志辖.基于策略和NETCONF管理的统一网关模型的研究与设计[D]. 武汉: 华中师范大学,2008.
- [25] 张沪寅, 吴黎兵, 吕慧等. 计算机网络管理实用教程[M]. 武昌: 武汉大学出版社. 2005: 15-98.
- [26] William Stallings. SNMP and SNMPv2: the infrastructure for network management[J]. CommunicationsMagazine.1998,36(3):110-115.
- [27] Torsten Klie ,Frank Straup.Integrating SNMP Agents with XML-Based Management Systems [J], IEEE Communication Magazine,Special Issue on XML-based Managementof Networks and Services in IEEE Communications Magazine,2004,30(8): 60-80.
- [28]STRAUBF, KLIET. Towards XML oriented Internet management[C]// GOLDSZM IDTGS, SCHONWALDER J. Proc of IF IP / IEEEInternational Symposium on Integrated Network Management. Dordrecht: Kluwer, 2003: 505-518.
- [29] 曾明,李建军. 网络工程与网络治理[M].北京:电子工业出版社,2003:139-180.
- [30] 刘学超.基于Web/XML的网络管理的研究与实现[D], 武汉: 华中师范大学, 2006:70-98.

致谢

三年的硕士生活，最后即将在这篇致谢词中划下句号。至此，在西电这个美丽的校园，我已经从懵懂的大一新生长大成即将踏入社会的研三毕业生。在每一步的成长路程中，我要感谢身边每一个帮助我的人。对我的培养，更离不开在我身边给过我帮助的每一个人。

首先感谢我的导师刘西洋教授。在将近半年的时间里，刘老师在毕业论文的学习和研究方法上都给我很多的指导，令我受益匪浅。他视野开阔、学识渊博、治学严谨、科研扎实、工作负责、有始有终，刘老师的这些优点令我敬重和佩服。无论是课题研究方向的指导，还是学术论文的撰写，刘老师都给我提出了很多建设性的意见。本论文是在刘老师的精心指导下完成的，因此在论文完成之际，我向导师表示衷心的感谢。

其次感谢黄东晓和张钢老师，在论文的选题和材料准备中，他们给予了我悉心的指导和帮助。没有两位老师的帮助，我就会缺少很多论文的相关参考资料，也不会有自身软件技术的快速成长。在论文完成之际，我向两位老师表示衷心的感谢。

感谢软件学院的所有老师在我研究生学习中的悉心教导及传授给我宝贵的知识，感谢牛琼琼导师对我的论文规范指导，让我受益匪浅。

同时要感谢身边默默支持帮助的同学，他们是胡小燕、李玮、李波、丁颖等。在整个研究生生涯，他们在研究中给了我很多有意义的建议和启发，在生活中大家和睦相处互相帮助，让我收获颇丰。

感谢我的父母、家人，他们默默地支持我，让我在意料之外的困境中坚持下来，获得前进的动力。

作者简介

1.基本情况

男，浙江舟山人，19892 年 10 月出生，西安电子科技大学软件学院软件工程领域 2012 级硕士研究生。

2.教育背景

2008.09～2012.07 就读于西安电子科技大学电子工程学院电子信息工程专业，获工学学士学位

2012.09～西安电子科技大学软件学院软件工程领域硕士研究生

3.攻读硕士学位期间的研究成果

3.1 实习项目经验

Comware V7 Web 项目，2013 年 7 月至 2014 年 7 月，独立完成接口管理、配置管理、IPv4、IPv6、VLAN 模块的 Web 页面研发。

