**分类号**  **学号 M201576139**

**学校代码10487 密级**

****

**硕士学位论文**

**基于ODL控制器的一种SDN系统的设计与实现**

|  |  |  |
| --- | --- | --- |
| **学位申请人** | **：** | **樊 开** |
| **学科专业** | **：** | **软件工程** |
| **指导教师** | **：** | **黄立群** |
| **答辩日期** | **：** | **2017.12.15** |

**A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree for the Master of Engineering**

**Design and Implementation of an SDN System**

**Based on ODL Controller**

**Candidate : Fan Kai**

**Major : Software Engineering**

**Supervisor : Prof.Huang Liqun**

Huazhong University of Science and Technology

Wuhan 430074, P. R. China

December, 2017

**独创性声明**

本人声明所呈交的学位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，除文中已经标明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□， 在 年解密后适用本授权书。

本论文属于

不保密√。

（请在以上方框内打“√”）

学位论文作者签名： 指导教师签名：

日期： 年 月 日 日期： 年 月 日

# 摘 要

一直以来，广域网的主要指责就是维持网络中的各个网络节点的通信，各个数据中心之间、总部和分支之间的业务流量等都需要经过广域网进行传输。广域网和业务之间属于两个分离的系统，其中，广域网主要作为业务系统的传输通道，被动的承载业务流量。但随着移动互联网、云计算等技术的日新月异的发展，这些技术不仅给人们的生活带来了极大的便利，同时也极大地提高了对传统网络体系的要求。在现有的网络体系下，网络资源分配大多通过路由器或交换机进行独立的配置来实现，其耗时多、操作复杂、易出错。作为未来网络的发展方向，SDN（Software Defined Network，软件定义网络）技术越来越受到人们的重视，其商业化进程也正在逐步的加快。SDN的主要思想是将网络的控制和数据平面进行分离，以应用驱动网络，实现网络主动适应用户业务和流量变化。

本项目主要是以SDN开源控制器OpenDaylight为基础，通过屏蔽底层不同设备的协议之间的差别向上层应用提供统一的API接口。通过SNMP、NetConf、OpenFlow等各类协议来对设备进行管理、编排等操作，通过Java、Maven、MongoDB、SVN等IT技术达到项目的开发，实现一款商用的SDN系统。

通过本项目，可以实现转、控分离，重构网络架构，增加网络的扩展性和适应性。同时，其集中化的控制可以充分的调度现网流量，达到流量路径的优化从而提高业务的服务质量。

**关键字：**网络 软件定义网络 流量 业务 控制

# Abstract

For a long time, the WAN is mainly responsible for the interconnection among network nodes, such as between data centers, and between a headquarters and a branch. The WAN and the service are two separate systems, where the WAN mainly serves as a transmission channel of a service system and passively carries the weight of service traffic. However, with the rapid development of technologies such as mobile Internet and cloud computing, these technologies not only bring great convenience to people's lives, but also raise the requirements for the traditional network system. In an existing network system, network resource allocation is mostly implemented by using a router or switch that is independently configured, which is rather time-consuming, complex, and error-prone. As a future development direction of the network, an SDN (Software Defined Network) technology has received more and more attention of human beings, whose commercialization is also gradually accelerated. The core idea of the SDN is to separate the control aspect of the network from the data aspect of the network and use the application to drive the network, so that the network can proactively adapt to user service and traffic changes.

This project is mainly based on the SDN open source controller OpenDaylight, which provides a unified API interface to upper-layer applications by shielding the differences between protocols of different devices at the bottom. Through the SNMP, NetConf, OpenFlow and other types of protocols to manage the devices, through Java, Maven, MongoDB, SVN and other IT technologies to achieve project development and get a commercial SDN system.

Through this item, it can realize the separation of forwarding and control, reconstruct the network architecture and increase the scalability and adaptability of the network. At the same time, the centralized control helps to integrate the resources of the whole network, and optimize the distribution of the traffic, so as to ensure the service quality of the service.

**Keywords:** Network SDN Traffic Service Control

[摘 要 I](#_Toc500424145)

[Abstract II](#_Toc500424146)

[1 绪论 1](#_Toc500424147)

[1.1 课题研究背景 1](#_Toc500424148)

[1.2 研究目的及意义 2](#_Toc500424149)

[1.3 国内外研究现状 6](#_Toc500424150)

[1.4 本文主要研究内容 7](#_Toc500424151)

[2 关键技术分析 8](#_Toc500424152)

[2.1 OpenDaylight控制器 8](#_Toc500424153)

[2.2 Maven 9](#_Toc500424154)

[2.3 MongoDB数据库 10](#_Toc500424155)

[2.4 HTML5 13](#_Toc500424156)

[2.5 本章小结 13](#_Toc500424157)

[3 ADWAN系统需求分析与设计 15](#_Toc500424158)

[3.1 ADWAN系统的需求分析 15](#_Toc500424159)

[3.2 ADWAN系统的系统设计 21](#_Toc500424160)

[3.3 历史信息数据库需求分析及设计 33](#_Toc500424161)

[3.4 本章小结 33](#_Toc500424162)

[4 ADWAN系统的实现及测试 35](#_Toc500424163)

[4.1 ADWAN的开发及运行环境 35](#_Toc500424164)

[4.2 ADWAN功能模块的实现 36](#_Toc500424165)

[4.3 历史信息数据库实现 50](#_Toc500424166)

[4.4 本章小结 52](#_Toc500424167)

[5 总结与展望 53](#_Toc500424168)

[5.1 全文总结 53](#_Toc500424169)

[5.2 展望 54](#_Toc500424170)

[致谢 55](#_Toc500424171)

[参考文献 56](#_Toc500424172)

# 绪论

近几年随着计算机技术的快速发展，大数据、云计算、移动互联网等业务的蓬勃发展[1]，这些服务在对人们生活提供了极大的便利的同时，也极大的增加了现有的计算机网络的负担，对目前的计算机网络的性能提出了更高的要求。基于这样的状况，软件定义网络（Software Defined Network，简称SDN）技术愈发被人们重视研究了起来，其热度持续升温[2]。

## 课题研究背景

一直以来，广域网的主要指责就是维持网络中的各个网络节点的通信，各个数据中心之间、总部和分支之间的业务流量等都需要经过广域网进行传输。其和业务应用属于两个独立的系统，基本上没有联系，更多的是作为业务系统的传输通道，实现业务流量的被动“承载”，但随着云计算、移动互联网等应用模式的发展和流量模型的改变，很多时候需要网络能主动“适应”业务流量，做到应用随需而变，但是由于目前的网络在管理上主要是面向设备而非业务的管理，视角上更多的是基于节点而非全局的视角，因此，产生了很多无法解决的问题：

1. 业务发布较慢，上线花费时间较长[2]：
   1. 现网中的设备繁多且地理位置分散，在部署业务时需要进行逐台操作，人工配置，配置难度大，操作繁琐；
   2. 现网中跑的各类业务数量巨大，其规则复杂，在进行手动设备配置时容易出错并且花费时间周期长。
2. 流量难以调去，灵活性很差：
   1. 由于目前的网络中没有统一的集中式管理且各设备都是基于本地的路由规则进行选路操作，这样选出来的路径往往并不是最优解，导致其带宽利用率低；
   2. 目前设备本地的选路策略其局限性很大，对工作人员的专业要求高且其不能动态的调整路由以适应网络状态的变化。
3. IT维护人员的运维体验很差：
   1. 网络管理手段有限，基本上是以手工为主，对工作人员的技术水平要求较高；
   2. 界面对用户不友好，不能可视化的呈现流量和业务的分布状态。同时在网络发生故障时不能快速的定位故障并恢复，具有很大的运维难度。
4. 网络过于封闭，不能适应业务对网络的动态需求变化：
   1. 网络设备种类繁多且系统封闭，很难快速的部署业务与定制不同的业务需求。
   2. 网络和应用不能有效联动，云计算应用的体验很差。

图1-1为传统广域网所面临的问题图示。

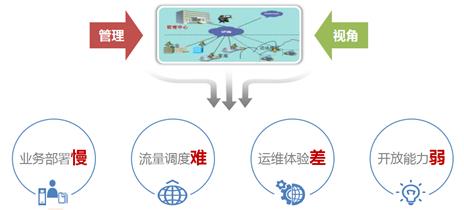


图1-1 传统广域网问题

## 研究目的及意义

### 云计算对广域网的需求

随着IT技术的发展，云计算已经日益在各行业进行了大规模部署。许多企业的传统数据中心正在向云计算数据中心过渡。然而在这个过渡过程中用户的需求体验是不会改变的。用户往往希望他们能像访问本地应用一样的速度、安全的访问云应用。这样就使的现网的需求发生了很大的变化。主要体现在以下几点：

1. 原先在本地运行的本地应用过渡到云端后，其运行的网络环境发生了很大变化。原先本地应用在本地局域网中运行，独占带宽资源。现在在广域网中运行，就会和其他广域网中的应用共享带宽，这就对广域网提出了更高的带宽、可靠性、承载力要求。
2. 在本地应用与服务器云端进行通信时，需要通过广域网进行。这种情况下，广域网内的各类业务繁多、其通过的流量也日趋复杂。这就需要针对不同的用户进行差异化定制以实现不同用户的不同应用的精细化调度，从而可以确保用户感受到的服务体验。
3. 随着云计算数据中心的日益发展，其部署的数量也越来越多。这些数据中心之间的通信交流也日益频繁，这就需要专门的DCI网络来保障服务质量，同时也要求DCI网络能够实时的动态计算、适应云计算应用。

目前现网的广域网网络的架构，难以扩展，不够开放，很难适应新时代的云计算业务的的需要，因此广域网重构和变革势在必行。

图1-2为云计算应用对现行广域网的需求示意图。



图1-2 云计算对广域网的需求

### 运营商转型对广域网的需求

随着时代的不断发展，许多IT技术都得到了蓬勃发展，例如数据中心、云计算、移动互联网等技术。现在用户不仅仅要求满足基本的上网业务，还要求其体验丰富。但是现在的网络提供商所提供的网络是不能满足这种要求的，还有许多疑难问题需要解决，例如：

1. 业务规模提高，收入不增。网络提供商发现虽然部署的业务越来越多，流量也是增长迅速，但其收入却没有显著增加。与流量规模的剪刀差愈来愈大。
2. 越来越高的网络成本：为了满足日益增长的流量、业务等要求，需要不断的购买和部署新的网络设备来提高网络承载能力，同时这些设备也配套专属的设备维护人员、机房等周边。运营成本不断攀升。
3. 业务部署时间长，交付耗时：现网中的网络设备厂商很多，型号繁多。同时这些不同厂商、不同型号的网络设备没有统一的标准的开发接口，导致难以自动化部署相关配置。这就造成了上线一个业务时需要针对不同的设备进行不同的配置、调试等，涉及到相关各类问题，使得整个部署、上线周期长。
4. 竞争压力大：现行的网络拓扑结构复杂、网络封闭不够开放，在面对虚拟运营商和互联网厂商的竞争时，有较大的压力。

为了解决上述的这些问题，运营商普遍认为业务转型是其主要的方法。然而在进行这个方案时，还需要面临下面这几个主要的问题：

1. 通过利用SDN技术来重构、优化现有的网络架构[4]。利用SDN技术整合全网资源，通过软件调度以驱动网络资源从而达到网络的服务型转变，产生新的价值。
2. 发挥整个网络的最大价值，深入挖掘并利用现网中的资源为用户提供趋近完美的、差异化的服务。
3. 通过开放统一的标准化接口以实现运维工作的简化与自动化。释放网络管理人员的人力，降低运维成本。

但是从目前的情况来看，总体上运营商的现网网络很难满足业务转型的要求。目前木槿需要对数据中心的网络进行以SDN思想和技术进行改造，也需要利用SDN技术对城域网和骨干网等网络进行变革和重构。图1-3为运营商对广域网的需求图示。



图1-3 运营商转型对广域网的需求

### 广域网的变革已成为必然

目前广域网迫切需要进行一次革命来应对运营商转型、传统广域网和云计算等遇到的需要解决的各类挑战和问题。图1-4所示是我们认为新一代广域网应具有的能力示意图。

1. 可以通过编自由、灵活的定制各类服务的能力：可以在用户业务的需求上基于代码实现的方式定制各类能力；
2. 可以调度全局资源的能力：可以在基于全网视角的情况下对全网的流量进行调度优化；
3. 开放的网络构建能力：通过开放的网络构建能力可以极大的提高整个网络的部署耗费时间；
4. 降低运维部署的压力：通过利用开放的网络接口来发开一些功能强大的运维应用，可以极大的降低系统的运维、部署压力，提高效率；
5. 图形界面可视的网络状态呈现：通过定制网络状态资源的可视化界面，便捷的用户管理操作。

图1-4为新一代广域网示意图。

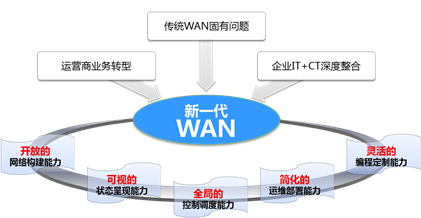


图1-4 新一代广域网示意图

## 国内外研究现状

SDN技术的起源来自于2006年美国斯坦福大学的Clean Slate项目组启动的关于网络安全与管理项目Ethane[6]。Ethane项目致力于使网络管理员通过一个集中式的控制器定义基于网络流的安全控制策略[7]，并将策略下发到控制器下的所属的各种网络设备中，从而实现对整个网络通信的安全控制[8]。在这个项目的研究基础上，研究人员通过对Ethane的进一步归纳总结[9]，设计了利用标准的、统一的各类网络设备的接口开发的集中式的将网络设备的数据转发和路由控制分离的控制器模型来对设备进行管理、配置，以达到优化网络资源的分配，促进网络的变革和发展。

经过多年的发展和进步，在2011年，开放网络基金会（open networking fundation，ONF）成立[10]，该基金会致力于推动和促成SDN的标准化，希望基于OpenFlow协议的SDN技术成为业界网络的新标准[11]。其成员涵盖了世界上主要的互联网和通信设备公司，以及部分运营商。其中有华为、ITU、中兴、IETF、腾讯、ETSI等组织。在设备研发领域，惠普、Juniper、IBM等厂商均已推出支持OpenFlow协议的SDN硬件。微软和谷歌公司已经采用OpenFlow技术实现了全球数据中心的互联，将链路的使用率提升到接近100%，并且利用网络拓扑和通信路径的动态调整技术提高了网络的可用性、容错力。在国内，中国通信标准化协会也在重点研究包括SDN在内的未来网络发展的核心技术，着手于SDN应用场景、系统架构、技术规范等标准的指定。2014年，中国电信北京分公司与华为公司共同宣布完成了全球首个运营商的SDN系统的部署[12]，这是首例成功使IDC网络应用了SDN技术。SDN技术，目前正在国内外迅速的发展。

## 本文主要研究内容

随着SDN技术的发展，SDN概念已经由最初的以OpenFlow为代表的狭义SDN概念演变为以软件定义网络为核心的广义SDN概念[13]，即现在SDN更多代表着构建网络的一种架构、一种思想，在本项目中主要研究的内容为以下几点：

OpenDaylight（简称ODL）控制器：OpenDaylight是一款开源的SDN控制器。在本项目中ODL作为整个项目的基石，其作用不言而喻。本项目主要研究ODL框架原理、架构以及底层协议的各类插件的使用了基于ODL框架项目的编码规范等。

Maven：Maven是一款Apache 以Java语言开发的一款开源的项目管理工具。在本项目的开发过程中，Maven作为其默认的项目管理工具，主要承担着第三方Jar包管理、编译、发布等功能。本项目中主要研究了Maven的依赖管理、编译等各方面的使用内容。

MongoDB数据库：本项目中的一些历史信息相关的功能中，其主要使用MongoDB作为第三方存储数据库。因此本项目中研究了MongoDB数据库的各类特点，其存数引擎、存储形式以及优化存储等相关内容。

SDN技术：通过研究SDN技术与思想，结合ODL、Java、Maven等IT技术来实现一款商用的SDN系统的开发与发布。

# 关键技术分析

## OpenDaylight控制器

OpenDaylight控制器（简称ODL）是一款全新的、开源的SDN控制器[14]。它是一款具有南向协议层、控制层、北向接口层这三层架构的SDN控制器。南向协议层主要是支持OpenFlow、LISP[15]等多种不同协议的模块。控制层则是一层具备完整的网络基础服务功能的模块。北向接口层负责对外提供REST[16] API以供外部的网络应用程序调用。同时它还具有集群工作模式，可部署在大规模的网络中。

目前SDN技术主要分为三层[17]：最下面一层是数据转发层，该层主要提供可以根据配置的路由规则转发报文的物理连接网络。而中间层则是控制层，该层主要是通过接口提供网络的编程能力，可以根据用户需求提供网络基础服务，是一个软件层。通过软件层可以提升网络的效率。最顶层是应用层[17]，主要是管理、优化和使用网络。

ODL控制器处于SDN技术模型的中间层，即控制层。它具备五个基础功能，分别是拓扑管理、交换机管理、路径转发管理、主机管理、网络资源切片管理[17]。ODL控制器是使用的OSGi技术[18]作为其技术架构，通过OSGi技术可以允许在ODL控制器运行时进行功能模块的热插拔。

ODL在设计架构上，从下到上分共有四层，他们分别是南向协议层、服务抽象层、控制层，以及北向接口层[17]。

在南向协议层中主要包含各类协议插件，例如BGP-LS[19]、SNMP[20]、SSH等，其主要功能是负责与现网中的设备进交互通信、下发报文、监听端口等。

服务抽象层是整个ODL架构设计的核心，它向上屏蔽了各类不同南向协议之间的不同差异，向北向接口提供统一的、标准的API接口。

控制层主要包括在OpenFlow等协议插件的基础上形成的6个基础服务：Switch Manager（交换机管理）、Statistics Manager（信息统计）、Topology Manage（拓扑管理）、Forwarding Rule Manager（转发规则管理）、ARP Handler（ARP报文处理）和Host Tracker（主机追踪），这六个模块通过互相协作向其他的功能模块提供接口。

北向接口层主要是对外提供Rest接口。外部的程序可以通过HTTP[21]的POST方式访问这些Rest接口，以获得响应的数据或者向控制器发出操作指令等。

图2-1为ODL控制器的架构示意图。

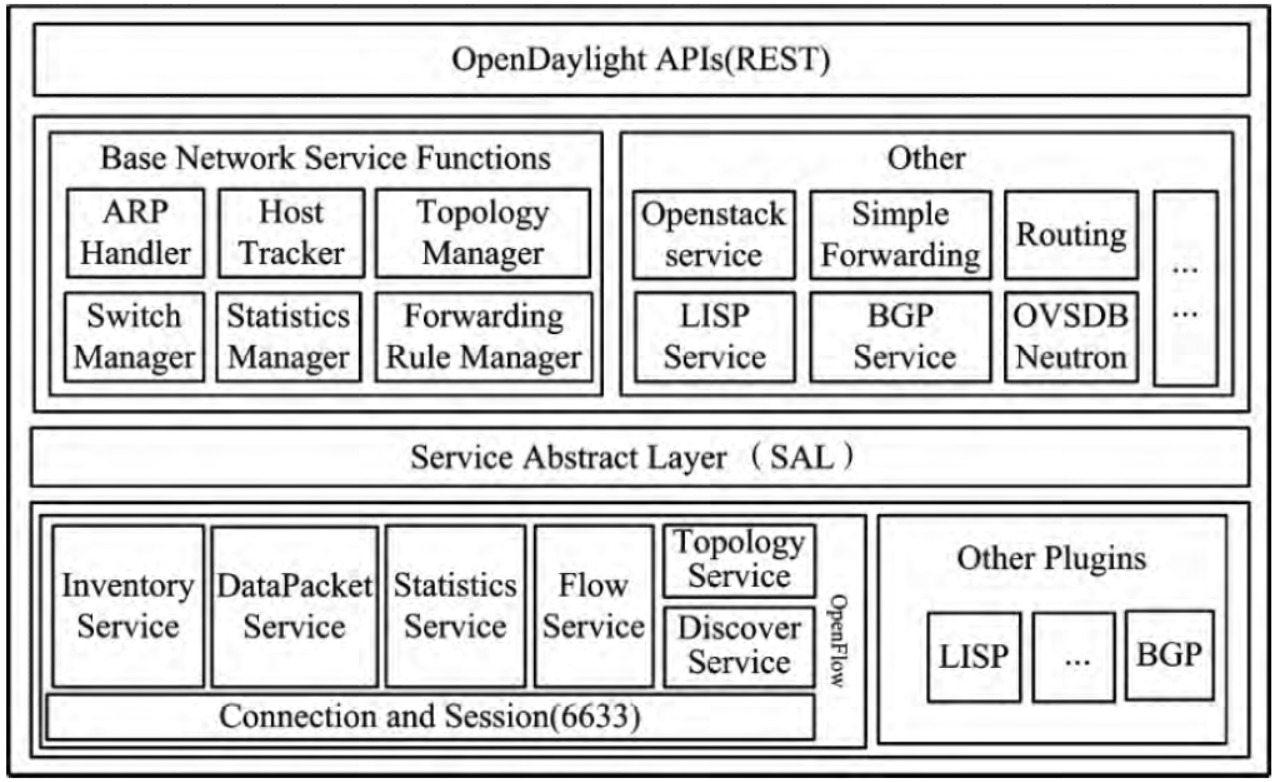


图2-1 ODL架构及功能图

## Maven

Maven是一款非常好的开源项目管理工具，项目对象模型（POM，Project Object Model）是Maven的主要思想理念[22]。Maven的主要功能就是在项目的各个环节中进行管理和控制。例如在项目构建（Build）中的清理、编译、测试、生成报告等环节[23]。通过调用Maven的命令都可以很方便的实现相应的功能，从而大幅减少了手动配置和构建项目工程的工作量。

POM项目对象模型：通过在项目根目录下的pom.xml文件进行一些相关的配置来实现项目对象模型的相关功能。通过pom.xml这个XML[24]文件里面定义了Maven项目的一些主要配置，这其中有项目的名称、项目的拥有者和该项目的一些相关依赖等内容。同时，还可以通过pom.xml进行指定一些配置来达到项目构建过程中的任意阶段的功能实现。

插件：绝大多数Maven的功能都是通过使用插件来实现的[23]。项目中的构建、测试等各个功能的实现配置都可以通过配置插件来实现覆盖。

项目周期：Maven项目的生命周期[25]主要如下表2-1所示。

表2-1 Maven的项目周期

|  |  |  |
| --- | --- | --- |
| 顺序 | 阶段 | 描述 |
|  | Process-resources | 源代码与依赖的处理 |
|  | Compile | 源代码编译 |
|  | Process-test-resources | 测试代码处理 |
|  | Test-compile | 测试代码的编译 |
|  | Test | 测试 |
|  | Package | 打包 |
|  | Install | 将包安装到本地库 |
|  | Deploy | 发布与部署 |

依赖：在Java项目中，处理各种Jar包之间的依赖关系一直都是一个繁琐的问题。比如我们打算使用Spring[26]包，就要把Spring包所依赖的包也引人进来，而这些被Spring所依赖的包也可能会依赖于其他的包。在使用Maven之前，我们需要手工的方式逐个向项目中导入所依赖的各类Jar包。而在Maven 之后，利用Maven我们可以很容易实现这个操作，我们所需要做的就是在pom.xml文件中配置我们对Spring的依赖就可以了。

## MongoDB数据库

MongoDB是一个旨在提供高性能、高可用性和自动扩展的开源文档型数据库（document database）[27]。MongoDB中的每一条记录称之为一个文档（document），这是由键、值对（field and value pairs）组成的数据结构[28]。它和JSON[29]数据很相似。Field的value可以是另一个document、可以是一个数组（arrays），可以是一个由document构成的arrays，还可以是一些基本数据类型。

图2-2为MongoDB数据库document的示意图。



图2-2 document示意

document具有以下几个优点：

1. Document，即对象，可以对应很多编程语言中的原生数据类型；
2. 通过嵌入式的document可以大幅度的减少关系型数据库利用表连接才可以获得的数据的消耗；
3. Document动态的数据结构可以流畅的支持数据的多态性。

MongoDB的几个关键特性：高性能（high performance）、强有力的查询语句（rich query language）、高可用性（high availability）、易水平扩展（horizontal scalability）、支持多种存储引擎（support for multiple storage engines）。下面分别对这几点做介绍。

### 高性能

MongoDB提供高性能的数据持久化操作。尤其是：

1. 因为有嵌入式数据模型因此MongoDB可以具有更少的数据库系统层上的I/O活动。
2. 通过索引，可以支持更快的查询。

### 功能强大的查询语言

MongoDB提供一个功能强大的查询语言，可以向用户提供更加便捷的据聚合查询、文本搜索和地理位置查询等强大功能。

### 高可用性

得益于MongoDB的副本集（Replica set）功能，使得MongoDB具有数据备份、故障自动转移等功能。

副本集是一组维护着一份同样的数据的MongoDB数据库服务器，可以提供数据的冗余备份和提高数据的可用性。

### 水平扩展

MongoDB的水平扩展性是作为它的核心功能提供的[30]：

通过分片（sharding）将数据分配至一组服务器集群上；

MongoDB自3.4版本后，便支持通过创建基于shard key的数据区域（zones of data）

### 支持多种存储引擎

MongoDB支持多种存储引擎[31]，目前官方版本自3.2版本及以后的默认使用存储引擎是WiredTiger存储引擎。此外在3.2版本之前的默认引擎为MMAPv1存储引擎。同时，MongoDB还提供了一种可插拔的存储引擎接口，只要是按照规范开发的第三方MongoDB存储引擎都可以接入。

1. **WiredTiger存储引擎**

WiredTiger存储引擎是MongoDB在3.0版本及后续版本中提供的新一代的64位存储引擎[32]。它具有以下几个特点：

1. Document Level Concurrency（文档级的并发控制）。WiredTiger引擎针对写操作使用了更加细粒度的并发控制，这种粒度是在document级别上的。因此在多客户端可以同时更新同一个集合下的不同文档。
2. Snapshots and Checkpoints（快照和检查点）。WiredTiger使用了多重版本并发控制（MVCC，MultiVersion Concurrency Control）技术[33]。WiredTiger引擎通过在一个操作开始签提供一个涉及数据的该时间点的快照以保障内存数据的一致性。
3. Journal（日志系统）。WiredTiger通过提前写入事物日志与checkpoints结合来保证数据的持久性。
4. Compression（数据压缩）。在WiredTiger引擎下，可以对数据库中的集合、索引进行压缩以减小数据的存储占用空间，但这样的操作会消耗额外的CPU计算资源。
5. Memory Use（内存占用）。MongoDB可以同时使用WiredTiger引擎的内部缓存和文件系统的缓存。并可指定其内存的使用量。
6. **MMAPv1存储引擎**

MMAPv1存储引擎是MongoDB最开始时使用的一个基于内存映射文件的存储引擎。其擅长处理大量的读、写操作的情景。

本项目采用的引擎为WiredTiger，MMAPv1不做过多的介绍。

## HTML5

相较于HTML4，HTML5 在标签元素上增加了新的特殊内容元素，比如article、footer、header [34]；新的表单控件，比如email、url、search[35]；新的页面元素，比如footer、figure。同时还废除了font、s、strike、tt、u、frameset、frame、noframes、plet、bgsound等元素[36]。在功能特性上HTML5 中加入了一些新特性，如语义化的标签、音频和视频的直接支持、矢量绘图、支持网页内容编辑、Web Socket[37]、地理定位、多线程处理以及离线存储等[38]。

HTML5 提供了画布功能，这个功能通过Canvas 标记元素来实现的[39]。利用该标签，可以使开发者利用JavaScript语言来进行一些诸如矢量图、栅格图的绘制。甚至一些非常复杂的图形动画等也可以表达出来。这些图形可以直接在浏览器上进行渲染显示[40]。

## 本章小结

在本项目中是一款使用了一些诸如Java 语言、Maven、新型的NoSQL 数据库[41]MongoBD、HTML5 等流行的IT技术，基于ODL（OpenDaylight）控制器上开发的商用的SDN产品。其通过对现有技术的组合利用，目的是帮助用户构建一个架构开放的、可以编程的、易于维护的新一代广域网整体解决方案。通过该方案可以有效的承载日益丰富的应用流量，实现网络的动态适应能力。

# ADWAN系统需求分析与设计

## ADWAN系统的需求分析

### ADWAN系统的总体架构需求分析

ADWAN系统是一款在全局视角的基础上通过多维观测并分析网络的实时状态信息，然后统一整合全网的资源来达到实现网络的层次化、可视化的。并且通过对全网资源的集中控制、调度、优化，以期实现用用户策略、应用等方面的内容驱动整个网络系统。它具有如下的需求：

1. 网络的全开放：在网络架构、接口方面进行全方位开放。使得网络的各个层次、组件、设备等完全解耦，从分的开放网络的扩展能力。从设备到控制器，ADAWAN提供多层次的、高抽象的接口，使网络更加便捷的可以被编程定制，更好的提供网络服务。
2. 使用情景的场景化：基于不同的使用场景，ADWAN提供了多个场景供用户选择。在不同的场景下，针对不同的业务需求定制APP应用，满足用户的多维度需求。
3. 全流程：为了简化运维难度，ADWAN应在全流程上重构网络。着眼于全局角度上的控制、管理网络。
4. 端到端：根据用户不用的业务需求，ADWAN应可以动态的进行安全部署、网络优化。向用户提供端到端的网络服务。
5. 可迁移：ADWAN应可以支持现网的网络平滑的迁移ADWAN方案上。

目前ADWAN总共分为：基础管理、设备管理、拓扑管理、流采集、运维管理、应用管理、告警管理、MongoDB中间层、流量调度以及用户界面等模块组成。共同组建了一个基于ODL框架下的SDN产品。下面分别介绍一下这些模块的需求分析。

图3-1为ADWAN的各功能模块示意图。

图3-1 ADWAN各功能模块示意图

### ADWAN系统各功能模块的需求分析

#### 基础管理

基础管理模块是整个ADWAN的基础模块，ADWAN的其他模块都依赖于基础管理模块。

基础管理的核心需求主要为：

1. 连接设备。ADWAN是一个SDN产品，所有的操作基本上都是与设备息息相关的。因此基础管理模块主要任务就是能够和不同厂商、不同型号的各类设备建立连接。其中与设备简历连接的方式多种多样，有OpenFlow发现、BGP-LS发现、SNMP等等协议方式。根据用户配置的不同的设备发现方式以及相关信息，基础管理应能够通过ODL框架与合法的设备建立连接并可正常保活设备。
2. 设备配置下发。基础管理模块作为底层模块，其应通过屏蔽底层不同设备厂商和型号、连接协议的差异，向上层模块提供统一的、标准的设备配置下发接口。
3. 网络场景的设置。ADWAN方案是基于场景化的思路的，因此应提供不同的网络场景供用户选择不同的部署方案。
4. License注册。ADWAN作为一款商业SDN软件，基础模块也负责着软件的License注册管理。
5. 当ADWAN控制器所在服务器内存或硬盘使用率超出阈值时，可以向告警管理模块发出告警，提醒用户注意。

#### 设备管理

众所周知，在互联网中的有许多通信设备厂商，同一家公司下的网络设备也有不同的型号，即使同厂商同型号的设备软件版本还有所不一致。不同的厂商、不同的型号、不同的软件版本的设备有诸多不同点、接口也有的不一致。通过基础管理管理模块，屏蔽了不同设备连接的差异性，然后设备管理模块即向其他模块提供统一的、标准被的设备管理相关的各类接口。

设备管理模块主要功能需求点为：

1. 对其他模块提供ADWAN系统下的所有设备或某个设备的状态信息。例如设备的厂商、型号、软件版本号、各接口状态信息、温度、内存使用等状态信息。获取这些信息需要依托于基础管理模块对设备下发命令。
2. 对其他模块提供ADWAN系统下的所有设备或某个设备的操作接口。例如Up/Down设备的某个接口、控制设备上下线、向设备下发某配置命令、新增设备、删除设备等操作接口。
3. 设备故障、ADWAN系统重启时，恢复设备配置等。
4. 当系统内某个或多个设备出现异常时，应发出相关信息，提醒用户注意。

#### 拓扑管理

ADWAN作为一款SDN产品，其中一个重要功能就是查看、管理网络拓扑中各设备、链路的状态信息。其中链路状态信息的查看和管理是拓扑管理模块的主要功能。

其中拓扑管理模块主要需求功能为：

1. 获取拓扑信息。获取ADWAN系统下的所有或某个拓扑的状态信息，包括拓扑下所含的节点、链路、名称等信息。
2. 拓扑管理。对网络拓扑进行增、删、改、查等管理操作。
3. 提供某条或所有链路的状态信息。例如链路的名称、源节点、目的节点、带宽、源接口、目的接口等信息。
4. 提供链路的管理接口。例如链路的上下线、设置链路的带宽、可分配带宽、源接口、目的接口、新增链路、删除链路等管理操作。
5. 当系统内某条链路出现异常时，可以向告警管理模块发出告警，提醒用户注意。

#### 流采集

流采集模块顾名思义，即是采集链路上流量信息为主要功能的模块。应用信息的获取与呈现项目中的流量采集功能就是由流采集模块提供支持。

其主要功能需求为：

1. 针对ADWAN系统内需要进行流量采集的链路下发流采集命令。
2. 实时根据设备侧上报上的流量信息更新link上的带宽信息。
3. 针对ADWAN系统内需要统计的应用信息下发流采集命令，采集应用信息状态。

#### 运维管理

运维管理，以流采集模块为基础，对外提供统一的流量信息、历史信息相关的接口。其中包括链路质量、带宽、应用带宽、链路历史信息、应用历史信息、设备运行状态、设备历史运行状态等。

运维管理核心需求点主要有一下几个方面：

1. 对外提供统一的设备、链路的信息（诸如设备的CPU、内存、链路的带宽等）状态接口；
2. 提供统一的设备、链路历史信息状态的查询接口；
3. 获取应用的相关的流量信息。

在应用信息的获取与呈现子项目中，其主要实现即在运维管理模块中实现。通过调用流采集接口获取设备侧报文信息并进行归并聚合，然后在调用运维管理的接口进行数据的存储及对外呈现的各接口的发布。

#### 应用管理

应用管理向应用信息的获取与呈现提供基础服务支持。流采集解析从设备侧发来的netstream报文时，会依据应用管理模块内的应用信息进行流信息的过滤、聚合等操作。

互联网中有许许多多的应用流量，诸如QQ聊天、优酷视频等等。这里我们定义一条应用信息为一个或若干个流的组合。应用管理模块的主要需求为以下几点：

1. 提供流、流组的增、删、改、查接口；
2. 设置应用调度策略、时间段等信息；
3. 控制网络范围。

#### 告警管理

告警管理顾名思义，就是指ADWAN系统内负责处理告警信息的模块。ADWAN作为一款SDN产品，其需要管理许多不同厂商、不同型号的网络设备。因此，基于设备的差异化，有时会出现一些设备连接不上的问题、或者设备内存使用过高、链路带宽占用率过高等等情况。这个时候告警管理模块就要发挥作用了。告警管理作为一个需要用户配置参数的模块，它会根据用户配置的参数来在适当的情况下向用户发出告警信息以提醒用户需注意的某些情况。

告警管理模块其主要需求为：

1. 向其他模块提供统一的告警接口；
2. 提供告警种类和告警相应的设置接口；
3. 向用户界面提示告警信息，引起用户注意等。

#### MongoDB中间层

ADWAN运行期间的设备、链路等状态信息存储在ODL控制器本身自带的DataStore内。DataStore是ODL控制器自身的数据存储容器。而关于ADWAN的历史信息、调度日志等数据则存储在第三方数据库MongoDB中。

作为使用的第三方数据库，MongoDB中间层主要需求为：

1. 可以连接本地或远程服务器上的MongoDB服务。
2. 当本地或者远程服务器上的MongoDB服务未启动时，可以拉起本地或远程服务器上的MongoDB服务。
3. 对其他模块提供统一的MongoDB数据库连接接口。
4. 当连接或启动MongoDB服务出现异常时，可以向告警管理模块发出告警，提醒用户注意。

#### 流量调度

作为一款商用的SDN产品，应用流量的调度功能是一项重要功能。流量调度的好坏决定着产品的质量。选路算法的优劣则决定这流量调度功能的好坏。

流量调度这里主要功能需求为：

1. 根据不同的网络场景和实时网络拓扑状态选择合适的调度策略。
2. 下发设备配置，定义应用报文转发路径。
3. 配置各类专线、VPN等业务。

#### 用户界面

ADWAN是基于ODL控制器实现的一个SDN系统。其通过Rest接口对外提供服务。但Rest接口展示的信息均为JSON或XML格式的数据，是一种非面向用户友好的数据。因此采用一种面向用户友好的显示方式，对于用户体验来讲至关重要！

本模块的主要功能需求为：

1. 转化用户操作，将用户的需求转化为ADWAN后台接口对应的入参；
2. 通过Ajax调用代理接口访问ADWAN的Rest接口，并获取后台响应的数据；
3. 通过AngularJS、jQuery、Bootstrap、Highcharts等前端框架，使响应数据转化为用户直观可见的图文报表的形式。

## ADWAN系统的系统设计

### ADWAN系统的总体系统设计

华三公司通过深入调研用户实际应用中存在的问题，利用其在广域网方面深度的技术和经验，通过SDN技术开发了新一代的具有架构开放、编程灵活，便于运维等特点的广域网整体解决方案，即ADWAN。通过ADWAN可以承载越来越多的应用流量，实现网络动态的调整、适应网络状态的能力。



图3-2 ADWAN方案架构

如上图3-2所示，ADWAN方案和其他场景下的SDN网络架构一样，都是一个分层、开放、灵活的网络架构，如上图所示，整个ADWAN方案分为网络设备、控制器+APP、管理编排三个层次：

1. 网络设备层：ADWAN不仅通过SNMP、NetConf等方式对其下属的网络设备进行管理和控制，而且还对优化了设备的转发层，提高了转发性能。
2. 控制器+APP层：ADWAN是基于开源的ODL控制器的，其支持各种APP的集成。可以通过定制的各类APP来满足不同场景下的各类需求，实现用户的差异化定制需求。
3. 管理编排层：通过对业务的流量、策略等方面的定义，利用APP提供的各类接口达到对整网的网络监控、可视化等功能。简化运维工作。

在基于ODL框架的基础上开发相应的业务逻辑模块具有很大优势与便捷性。可以是开发人员着重于聚焦业务逻辑问题，避免在底层协议、控制设备上面付出过多的精力。

目前ADWAN总共分为：基础管理、设备管理、拓扑管理、流采集、运维管理、应用管理、告警管理、MongoDB中间层、流量调度以及用户界面等模块组成。共同组建了一个基于ODL框架下的SDN产品。下面分别讨论一下这些模块的系统设计。

### ADWAN系统各功能模块的设计

#### 基础管理

基础管理模块作为ADWAN系统的基础模块，ADWAN的其他模块都依赖于基础管理模块。根绝基础管理的相应需求，设计如表3-1所示的ODL数据模型，并在具体实现中提供相应需求的具体接口。

表3-1 基础管理模块的主要ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 范围 | 默认值 |
| sceneName | 场景类型 | Integer | 1-5 | 1 |
| networkType | 网络类型 | Integer | 1-2 | 1 |
| licenseStatus | License状态 | Object | -- | -- |

#### 设备管理

设备管理主要负责设备相关的功能，根据设备参数情况设计如表3-2所示的ODL数据模型，并在具体实现中提供相应需求的具体接口。

表3-2 设备管理模块的主要ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 范围 | 默认值 |
| deviceId | 设备唯一标识 | UUID | -- | -- |
| deviceName | 设备名称 | String | 1-255（长度） | -- |
| manageIp | 设备管理IP | String | -- | -- |
| manageMac | 设备管理口Mac地址 | String | -- | -- |
| deviceType | 设备类型：  1：通用设备  2：虚设备 | Integer | 0-1 | 0 |
| description | 描述 | String | 1-255（长度） | -- |
| deviceStatus | 节点状态：  1：active  2：inactive | Integer | 1-2 | 2 |
| inactiveReason | 节点下线原因 | Integer | 0-65535 | -- |
| role | 通用设备角色 | Integer | 1-65535 | -- |
| scene | 设备所属场景 | Integer | 1-5 | -- |
| serialNum | 设备序列号 | String | 1-31（长度） | -- |
| serialNums | 设备序列号列表 | List<String> | -- | -- |
| sysObjectOid | 设备OID | String | -- | -- |
| softVersion | 软件版本号 | String | -- | -- |
| company | 厂商信息：  0：H3C  1：HP  2：Cisco  3：HuaWei  4：Juniper  65535：Unknown | Integer | 0-65535 | 65535 |
| model | 设备型号，如MSR810 | String | -- | -- |
| regionId | 设备所属区域信息 | UUID | -- | -- |
| tpIds | 接口索引列表 | List<UUID> | -- | -- |
| comeFrom | 设备来源 | Integer | 0-4 | -- |
| nodeOperStatus | 设备业务状态  1：up  2：down | Integer | 1-2 | -- |

#### 拓扑管理

拓扑管理主要负责网络拓扑相关的功能，其中主要拓扑的主要构成即为链路。根据链路参数情况设计如表3-3所示的ODL数据模型，并在具体实现中提供相应需求的具体接口。

表3-3 拓扑管理模块的主要ODL数据模型

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 描述 |
| topoId | UUID | 拓扑唯一标识 |
| topoName | String | 拓扑名称 |
| description | String | 拓扑描述 |
| nodes | List<UUID> | 该拓扑下的节点ID列表 |
| links | List<UUID> | 该拓扑下的链路ID列表 |

表3-4为ADWAN系统内链路的数据模型。

表3-4 链路的ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 范围 | 默认值 |
| topoId | 拓扑索引标识 | UUID | -- | -- |
| linkId | 链路唯一标识 | UUID | -- | -- |
| linkName | 链路名称 | String | 1-255（长度） | -- |
| owner | 链路所属的拓扑类型：  1：物理拓扑  2：业务拓扑 | Integer | 1-2 | 1 |
| linkType | 链路类型 | Integer | 1-65535 | 100 |
| srcNodeId | 源节点索引 | UUID | -- | -- |
| srcTpId | 源接口索引 | UUID | -- | -- |
| dstNodeId | 目的节点索引 | UUID | -- | -- |
| dstTpId | 目的接口索引 | UUID | -- | -- |
| linkStatus | 链路状态：  1：up  2：down | Integer | 1-2 | -- |
| metric | 链路metric值 | Long | 0-4294967295 | 0 |
| bandwidth | 链路带宽 | Long | 0-4000000000（kbps） | -- |
| reservable-Bandwidth | 可分配带宽 | Long | 0-4000000000（kbps） | -- |
| tunnelId | 链路对应的隧道索引 | UUID | -- | -- |

#### 流采集

流采集模块顾名思义，即是采集链路上流量信息为主要功能的模块。根据其需求特点设计如下所列表格所示的ODL数据模型。

表3-5为设备下发的ACL数据模型。

表3-5 ACL的ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 参数描述 | 参数类型 | 参数范围 | 参数默认值 |
| aclType | ACL匹配IP类型 | Enum | IPv4、IPv6 | IPv4 |
| groupCategory | ACL类别 | Enum | Basic、Invalid、Advanced | Advanced |
| rules | ACL下所绑定的rule列表 | List<Rule> | -- | -- |

表3-6为ACL内的Rule数据模型。

表3-6 Rule的ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 参数描述 | 参数类型 | 参数范围 | 参数默认值 |
| ruleId | Rule的唯一标识 | UUID | -- | -- |
| dscp | 优先级 | Integer | 0-16 | 0 |
| protocol | 协议类型 | Integer | 6（TCP）、  17（UDP）、  256（IP） | 256（IP） |
| type | 类型：1（permit）、2（deny） | Integer | 1-2 | 1 |
| src | 策略路由需要匹配的出接口索引 | IP-port | -- | -- |
| dest | 策略路由需要匹配的下一跳地址 | IP-port | -- | -- |

表3-7为Rule内的IP-Port数据模型

表3-7 IP-port的ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 参数描述 | 参数类型 | 参数范围 | 参数默认值 |
| ip | IP地址 | String | -- | -- |
| mask | 掩码 | String | -- | -- |
| port | 端口 | Integer | 0-65535 | -- |
| toPort | 截止端口 | Integer | -- | 65535 |
| portOp | 端口选择 | 1（lt，小于）、  2（eq，等于）、  3（gt，大于）、  4（neq，不等于）、  5（range，范围） | 1-5 | 1 |

#### 运维管理

运维管理，以流采集模块为基础，对外提供统一的流量信息、历史信息相关的接口。其中包括链路质量、带宽、应用带宽、链路历史信息、应用历史信息、设备运行状态、设备历史运行状态等。表3-8为应用信息的主要ODL数据模型。

表3-8 应用信息ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 范围 | 默认值 |
| flowGroupId | 应用组的唯一标识 | UUID | -- | -- |
| flowGroupName | 应用组的名称 | String | -- | -- |
| bandwidth | 应用组带宽 | Long | -- | -- |
| bandwidthPercentage | 应用组带宽占链路总带宽的百分比 | Integer | 0-100 | -- |

在应用信息的获取与呈现子项目中，可提供用户查询近一年的任意时间段的应用信息的详情。因此使用MongoDB存储数据时需要考虑到数据的存储方式以达到用户无论请求什么什么时间段的数据都可以迅速的作出响应。这里主要应用到数据分层、分集合技术对数据进行存储同时采用了MongoDB副本集技术对数据进行备份。

#### 应用管理

应用管理向应用信息的获取与呈现的提供基础服务支持。流采集解析从设备侧发来的netstream报文时，会依据应用管理模块内的应用信息进行流信息的过滤、聚合等操作。根据本模块的需求特点设计如表3-9为应用管理中应用的ODL数据模型。

表3-9 应用的ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 范围 | 默认值 |
| appId | 应用唯一标识 | UUID |  |  |
| appName | 应用名称 | String | 0-255（长度） |  |
| flows | 应用匹配的流 | List<Flow> |  |  |
| comment | 备注 | String |  |  |

表3-10为应用管理中应用的flow的ODL数据模型。

表3-10 flow的ODL数据模型

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 范围 | 默认值 |
| protocol | 协议类型 | String | ip、tcp、udp |  |
| srcAddr | 源IP | String | 网段，或精确的IP。必须具有掩码 |  |
| srcPort | 源端口范围 | String | eq/lt/gt/neq/range + 端口号 |  |
| dstAddr | 目的IP | String | 网段，或精确的IP。必须具有掩码 |  |
| dstPort | 目的端口范围 | String | eq/lt/gt/neq/range + 端口号 |  |
| matchMode | 匹配模式 | String | permit/deny |  |

#### 告警管理

告警管理顾名思义，就是指ADWAN系统内负责处理告警信息的模块。ADWAN作为一款SDN产品，其需要管理许多不同厂商、不同型号的设备。因此，基于设备的差异化，有时会出现一些设备连接不上的问题、或者设备内存使用过高、链路带宽占用率过高等等情况。这个时候告警管理模块就要发挥作用了。告警管理作为一个需要用户配置参数的模块，它会根据用户配置的参数来在适当的情况下向用户发出告警信息以提醒用户需注意的某些情况。

根据ADWAN系统内的实际情况，设计告警管理共有4大类告警信息，12类细分告警信息。下表为各类告警信息的类型编号及描述。

表3-11为四大类告警信息描述。

表3-11 告警管理模块的四大类告警信息

|  |  |  |
| --- | --- | --- |
| 告警分类 | 编号 | 告警描述 |
| 链路告警 | 1 | 链路状态相关的告警分类。 |
| 设备告警 | 2 | 设备状态相关的告警分类。 |
| MongoDB告警 | 3 | MongoDB数据库服务启动、连接时的告警分类。 |
| 控制器告警 | 4 | 控制器所在服务器状态的告警分类。 |

表3-12为告警管理中的12类具体告警信息。

表3-12 告警管理模块的12类细分告警

|  |  |  |
| --- | --- | --- |
| 告警类型 | 告警编号 | 告警描述 |
| 链路下线 |  | 链路告警，链路不正常下线时告警。 |
| 延时 |  | 链路告警，链路延时超过阈值时告警。 |
| 抖动 |  | 链路告警，链路抖动超过阈值时告警。 |
| 丢包率 |  | 链路告警，链路丢包率超过阈值时告警。 |
| CPU利用率 |  | 设备告警，设备CPU使用率超过阈值时告警。 |
| 温度 |  | 设备告警，设备温度超过阈值时告警。 |
| 剩余内存 |  | 设备告警，设备剩余内存超过阈值时告警。 |
| 设备下线 |  | 设备告警，设备不正常下线时告警。 |
| MongoDB服务启动 |  | MongoDB告警，ADWAN试图启动服务器上MongoDB时失败告警。 |
| MongoDB服务连接 |  | MongoDB告警，ADWAN试图连接服务器上MongoDB时失败或丢失MongoDB连接告警。 |
| 控制器内存告警 |  | 控制器内存告警，控制器所在服务器内存使用率超过阈值时告警。 |
| 控制器硬盘告警 |  | 控制器硬盘告警，控制器所在服务器各硬盘分区使用率超过阈值时告警。 |

#### MongoDB中间层

ADWAN运行期间的设备、链路等状态信息存储在ODL本身自带的DataStore内。DataStore是ODL自身的数据存储容器。而关于ADWAN的历史信息、调度日志等数据则存储在第三方数据库MongoDB中。

在MongoDB数据库中间层中，设计为用户通过配置MongoDB的配置文件来进行MongoDB的连接。在本模块中没有建立ODL的数据模型。

#### 流量调度

作为一款商用的SDN产品，应用流量的调度功能是一项重要功能。流量调度的好坏决定着产品的质量。而流量调度功能的好坏取决于选路算法的优劣。根据用户需要目前主要有四种调度算法，分别是：

1. 基于带宽利用率调度，双链路上行。
2. 基于延时调度，双链路上行。
3. 基于丢包调度，双链路上行。
4. 基于时间段调度，双链路上行。

#### 用户界面

ADWAN是基于ODL控制器实现的一个SDN系统。其通过Rest接口对外提供服务。但Rest接口展示的信息均为JSON或XML格式的数据，是一种非面向用户友好的数据。因此采用一种面向用户友好的显示方式，对于用户体验来讲至关重要！

目前的实现是B/S架构，即通过浏览器展现页面的形式供用户操作、查看ADWAN的数据。

目前WEB界面主要是通过node.js作为WEB服务的后台，通过架设代理访问ADWAN的Rest接口实现的。这样，在页面上可直接通过Ajax访问代理接口请求数据，然后前端使用最新的html5技术绘制图表显示。主要使用的框架有AngularJS、TWaver、Hightcharts等框架。

## 历史信息数据库需求分析及设计

在上述ADWAN的功能模块架构设计中，其中各个模块牵涉到的ODL数据模型均为ODL框架本身自带的数据模块。而一些相关的历史信息如应用信息的获取与呈现功能则牵扯到第三方数据库MongoDB的存储。根据该功能的需求特点和实际数据情况环境，采用了以下技术方案对其进行数据库设计。

1. 分层设计

采用数据分层技术，通过对原始数据的提取、聚合操作使数据库中的数据模型以时间段为单位进行层级划分，可以有效的提高在长时间段的数据返回的效率。

1. 分集合设计

MongoDB是一款NoSQL型数据库。MongoDB其没有传统关系型数据库中表的概念。其与表相对的对象称为集合。在考虑到迅速返回用户结果，提高插入、查找数据的效率，这里采用了以季度为间隔的分集合数据存储形式。

表3-13为应用信息的获取与呈现子项目中主要数据存储字段

表3-13应用信息的获取与呈现中的主要数据存储字段

|  |  |  |
| --- | --- | --- |
| 数据字段 | 描述 | 类型 |
| Timestamp | 时间戳。 | Long |
| LinkId | 对应的链路UUID。 | String |
| FlowStr | 流量的标识。通过对源IP、目的IP、源端口、目的端口、协 议类型进行字符串拼接，达到记录一条流量信息的目的。 | String |
| Bandwidth | 带宽大小 | Long |

## 本章小结

ADWAN作为一款基于ODL框架的SDN产品，其基础功能需求即为和设备的交互，在基础功能满足之上实现其核心的调度功能。同时丰富周边的支持功能，例如查看网络拓扑、现网流量分布等。通过对ADWAN各类功能的拆解使其形成模块化的开发，这样不仅极大的提高了开发人员的开发效率，也促进了各功能的解耦和，便于产品的功能扩展。这些不同的模块互相协作，综合作用最终实现了新一代的基于SDN思想构建的广域网解决方案。同时应用信息的获取与呈现作为ADWAN对外提供的一个功能，其拥有自己的第三方数据存储，并针对性的对其数据存储进行优化，以达到尽可能的迅速响应操作。

# ADWAN系统的实现及测试

经过前面几章对ADWAN的架构分析，我们可以认识到ADWAN的总体由基础管理、设备管理、拓扑管理、流采集、运维管理、应用管理、告警管理、MongoDB中间层、流量调度以及用户界面等模块组成。它们通过相互结合，共同实现了新一代的基于SDN技术、思想的广域网解决方案。

本章节主要论述各模块的具体实现。由于篇幅原因和笔者实际参与度的问题，论述的主要焦点为应用信息的获取与呈现功能的具体实现。这其中包括设备应用流量的采集与解析、应用信息的数据存储方式、WEB页面的显示逻辑和绘图方式等。

## ADWAN的开发及运行环境

### ADWAN的开发环境

ADWAN属于运行在服务器上的SDN产品，用户通过浏览器访问控制器达到使用的目的。其是在基于开源SDN控制器ODL进行的二次开发。其使用IntelliJ IDEA作为其IDE，JDK8作为开发环境，使用了Maven作为项目管理工具，使用SVN作为版本控制工具，Jenkins作为其自动编译、构建工具。开发工作机的操作系统为Ubuntu 14.04。而在WEB端，Chrome作为默认的开发浏览器。

### ADWAN的运行环境

服务器的配置应满足以下表4-1所示的最低配置。

表4-1 ADWAN系统运行的最低配置

|  |  |  |
| --- | --- | --- |
| 属性 | 参数 | |
| 服务器配置 | 处理器 | 主频≥2.6GHz，CPU核心数≥8 |
| 内存 | 64G以上 |
| 硬盘 | 512G以上 |
| 网口 | 2个以上的10GE网口 |
| 操作系统 | CentOS 6.5或Ubuntu 14.04 |
|  | Java运行环境 | Java8及以上 |
| 客户端配置 | 处理器 | 主频≥2Ghz，CPU核心数≥2 |
| 内存 | 2G以上 |
| 操作系统 | Windows或Linux |
| 浏览器 | Chrome 45以上 |

## ADWAN功能模块的实现

### 基础管理

基础管理作为ADWAN系统的基础模块，其主要核心功能即是建立与各类设备的连接，同时向其他模块提供该设备配置下发的接口。

这里和设备建连接的方式主要有SNMP、BGP-LS、OpenFlow三种方式，这里主要使用了ODL框架的提供协议插件与设备进行连接。通过用户在设备上、控制器上配置SNMP、BGP-LS和OpenFlow相关的参数来达到使控制器发现并连接设备的目的。

同时，由于配置下发牵扯到不同类型的设备，根据设备不同的厂商与型号，配置内容有所区别。这里主要是采用了配置XML文件与设备型号、版本进行匹配的操作来达到正确识别命令的下发参数、内容，然后通过NetConf协议向设备下发配置报文并获取设备响应报文。

通过XML配置文件，控制器向不同的厂商、型号设备下发配置命令。不同的配置命令具有不同的语句参数，通过XML文件和接口入参的键值对匹配，将不同的配置命令的NetConf报文直接配置在XML文件中，这样在下发某个命令的配置时，便可以直接寻找到该命令的XML文件，然后根据文件内容下发配置。

图4-1为某个XML命令文件的示意图。



图4-1 获取VCPE设备的虚拟机状态信息的命令

在其他模块向设备下发配置时，基础管理会根据已经配置好的XML文件寻找对应的配置命令参数与语句，并对传来的入参对应填到命令中去。利用NetConf协议向设备下发配置命令。接收设备响应的NetConf报文。

同时向设备下发配置时，还采用了一种小版本识别技术。此技术主要是针对解决同型号的设备不同软件版本号具有不同命令格式、或者支持的命令不一致时这样的问题的。具体来讲就是当其他模块向设备下发配置时，基础管理会先寻找到该款型的设备软件版本配置文件，如果有此文件则根据文件上的命令、路径寻找对应的XML命令文件进行下发配置。如果没有，则按照此款型的通用XML文件下发命令。

图4-2为小版本识别的XML配置图。



图4-2 小版本XML配置

综合来讲，利用XML配置文件加小版本识别，基础管理可以屏蔽不同设备的差异向上次其他模块提供统一的下发配置接口。

同时由于向设备下发配置需要进行网络IO，因此下发配置一般为异步操作，并且提供了一个专门的具有4个固定线程的线程池来进行下发配置操作。

基础管理除了连接设备、提供设备下发配置的接口外，还提供了设备的保活功能。现网中网络环境复杂多变，设备的保活功能即是通过定时向设备发送Hello报文来确定设备是否在线保持连接。这样当一个设备保活失败时，可以迅速的响应到用户层面。

同时用户场景的设置也是在本模块中实现的。目前ADWAN主要支持三类场景，分别是行业纵向网、骨干网/DCI以及分支/专线，他们分别对应不同的调度策略。

图4-3为ADWAN的纵向网调度示意图。

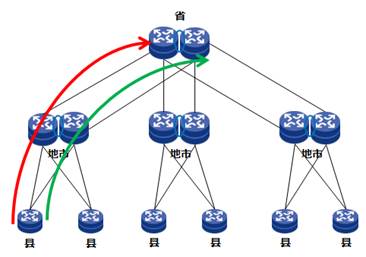


图4-3 场景一：ADWAN纵向网调度

图4-4为ADWAN骨干网/DCI调度的示意图。

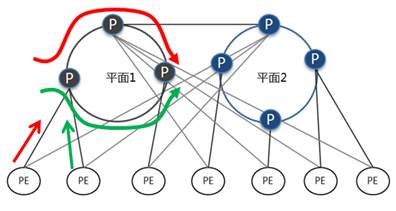


图4-4 场景二：ADWAN骨干网/DCI调度

图4-5为ADWAN分支/专线调度的示意图。

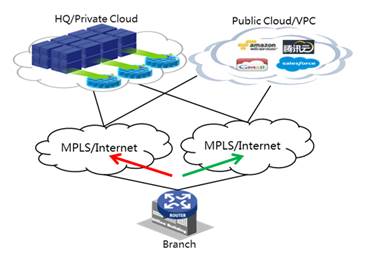


图4-5 场景三：ADWAN分支/专线调度

表4-2是ADWAN基础管理主要对外提供的主要接口的信息。

表4-2 基础管理模块的主要接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| setNetworkScene | 用户设置组网场景。 |
| bgpConfig | 设置BGP的基本配置，修改后BGP邻居需要重新建立。  主要配置BGP-LS的AS号和BGP的ID值。 |
| bgplsSetPeers | 设置BGP-LS多域邻居。  在设备上配置BGP-LS相关的配置可以使控制器自动发现一些设备极其拓扑结构。通过BGP-LS自动发现设备，可以避免重复性手动添加设备、链路，在设备数量大时效果尤为明显。 |
| addNetConf | 增加一条NetConf配置信息。 |
| addSnmp | 增加一条SNMP配置信息。  通过配置的snmp信息，ADWAN与设备进行连接。 |
| distributeConfigAsync | 下发配置。通过指定设备的标识和配置命令名称，基础管理模块会对应的自动在XML文件中寻找相应的命令格式然后填充对应的命令参数，通过NetConf协议下发配置。在应用信息的获取与呈现中，主要就是通过此接口向设备下发netstream配置命令。 |

在基础管理模块中，主要使用到的技术有多线程、线程池、ssh通信、Http中的Post提交表单等技术。

### 设备管理

设备管理模块主要是向其他模块提供设备信息获取、设备管理的接口API以及设备配置恢复、设备告警相关的内容。

设备管理模块通过获取基础管理模块中的设备信息并进行相应的组织、格式化数据然后存储在内部变量，并定期持久化到ADWAN自带的DataStore中。同时，其会记录设备的操作日志并备份设备上的配置命令，以用来在重启控制器时进行设备配置的回滚。表4-3是ADWAN基础管理主要对外提供的主要接口的信息。

表4-3 设备管理模块的主要接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| addDeviceManual | 通过手动的方式向ADWAN系统中增加一个设备。 |
| updateDevice | 根据请求参数中的设备的唯一标识更新该设备的一些状态信息。诸如设备厂商、版本等信息。 |
| deleteDevice | 根据请求参数中的设备UUID删除ADWAN系统中的该设备。 |
| getDevice | 根据请求参数的不同，获取ADWAN中所有设备的信息或通过设备唯一标识获取该设备的信息。 |
| getTerminalPoints | 获取ADWAN中所有设备的所有接口信息或者通过请求参数中的设备的UUID获取该设备的接口信息。 |
| getTerminalPointById | 通过接口唯一标识获取该接口的状态信息。 |

### 拓扑管理

拓扑管理主要功能是呈现出现网中实际的网络拓扑图，并可以实现对拓扑中的链路、专线进行增、删、改、查等管理操作。

表4-4为拓扑管理模块主要实现的接口描述。

表4-4 拓扑管理模块的主要接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| getTopo | 通过拓扑信息以及拓扑内的设备ID列表和链路ID列表。 |
| addLinkManual | 通过手动的方式增加一条链路。 |
| updateLink | 通过链路的唯一标识更新链路的一些状态信息。 |
| deleteLink | 通过链路的唯一标识删除ADWAN中的某一条链路。 |
| getLink | 获取ADWAN中所有链路的信息或通过链路唯一标识获取该链路的信息。 |

### 流采集

流采集是ADWAN中的一个基础模块，其主要功能通过向设备下发流采集命令采集设备上流经的流量信息，流采集模块向其他模块提供接口供其他模块使用。当其他模块注册接口或链路时，流采集便会向对应的接口下发流采集命令并将收集到的应用信息存储在其本地缓存中供其他模块读取。

在应用信息的获取与呈现中，流采集通过向网关节点下发netstream配置命令，设备会向指定的IP地址发送netstream的UDP报文。通过解析设备侧上报的netstream报文并与应用管理里的应用进行匹配、聚合，可以得出在这次报文内的各个应用的流量信息内容。之后即可调用MongoDB数据库中间层的接口将获取到的原始信息存储进数据库中。

通过利用流量信息的六元组来达到匹配对应的应用。如前文所述，一个应用可以包括一个或若干flow组成。在ADWAN中，一个应用被表示为流组，而flow称之为流。流里面根据源IP、源端口、目的IP、目的端口、协议、匹配类型六元组定义的。表4-5为应用字段说明。

表4-5 应用字段示意

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 范围 | 默认值 |
| appId | 应用唯一标识 | UUID | -- | -- |
| appName | 应用名称 | String | 0-255（长度） | -- |
| flows | 应用匹配的流 | List<Flow> | -- | -- |
| comment | 备注 | String | -- | -- |

表4-6为应用中flow字段说明。

表4-6 flow字段示意

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 描述 | 类型 | 范围 | 默认值 |
| protocol | 协议类型 | String | ip、tcp、udp | ip |
| srcAddr | 源IP | String | 网段，或精确的IP。必须具有掩码 | -- |
| srcPort | 源端口范围 | String | eq/lt/gt/neq/range + 端口号 | -- |
| dstAddr | 目的IP | String | 网段，或精确的IP。必须具有掩码 | -- |
| dstPort | 目的端口范围 | String | eq/lt/gt/neq/range + 端口号 | -- |
| matchMode | 匹配模式 | String | permit/deny | permit |

在解析出netstream报文后，会根据解析出的流组里面的流规则逐一进行匹配。符合则统计，不符合则抛弃并进行下一个报文。最终统计出该时间段内的应用带宽信息并调用数据库接口存储在MongoDB中。这里需要注意的是，匹配IP时需要考虑掩码，匹配端口时需要考虑端口可以是等于、大于、小于甚至时一个范围的情况，因此匹配flow的方法需要单独进行注意。

表4-7为流采集模块的主要实现接口。

表4-7 流采集主要实现接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| registerFgInPhysicallIf | 注册应用接口。注册后，流采集模块会在设备指定物理接口出或入方向对指定应用的流量进netstream采集，并返回一个采集id用于唯一标记这次注册，后续获取该应用的流量时，通过该采集Id可以获取到在该物理口采集的应用流量，并将采集到的流组实例的统计信息进行存储，供其他业务使用。 |
| registerLinkBandwidth | 注册link接口，只有注册过的link，流采集才会去获取该link的流量信息。 |

### 运维管理

运维管理作为提供统一的流量信息、历史信息相关的接口的模块。其主要是通过对外提供Rest接口，组织相关参数逻辑，调用其他底层模块的接口处理业务。

表4-8为运维管理主要的实现接口。

表4-8 运维管理模块的主要实现接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| getLinkQuality | 获取link质量信息，用户通过此接口查询链路质量信息。例如延时、抖动、丢包率等。 |
| getLinkBandwidth | 获取链路/隧道上的实时带宽（总流量）：用户通过此接口查询链路/隧道上的实时带宽（总流量）。 |
| getLinkFlowgroupBandwidth | 获取指定link上的应用组带宽信息。 |
| getHistoryLinkFg | 获取指定link上的应用组带宽历史信息：用户通过此接口查询在指定时间段内，经过指定链路的应用组带宽历史信息，历史信息的最小粒度为1分钟，开始时间的默认值为结束时间前一个小时，结束时间的默认值为当前时间，时间格式为“年-月-日 小时-分钟-秒 时区”，如2016-04-19 16:10:00 GMT+8。返回值的时间所在时区为输入时间时区，startTime和endTime的时区必须相同。返回值的粒度是根据输入时间段大小来确定的。 |
| clearHistoryData | 一键清除链路、设备、应用的历史信息。根据入参的不同，清除不同类型的信息。 |
| getDeviceRunStatus | 获取某台设备运行状态统计信息，内存使用率、cpu利用率、温度等。 |

### 应用管理

应用管理模块主要负责ADWAN系统内定义应用相关的内容。如上文所述的流组、流等均在应用管理内进行定义、操作、管理、调度策略等。同时包括向设备下发根据应用生成的ACL。表4-9是应用管理的主要实现接口。

表4-9 应用管理模块的主要接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| addFlowgroup | 增加一条流组信息 |
| deleteFlowgroup | 删除一条流组。 |
| addFlow | 新增一条流信息。 |
| addNetworkScope | 设置一个网络范围信息。 |

### 告警管理

告警管理作为ADWAN的功能模块，其主要作用是向其他模块提供发送告警的接口并显示到用户界面上。其通过每一条的告警的唯一标识、告警的各项字段参数等确定告警的显示状态。表4-10是主要告警参数。

表4-10 告警属性参数

|  |  |  |
| --- | --- | --- |
| 参数 | 描述 | 类型 |
| alarmId | 告警标识 | String |
| timestamp | 告警时间 | Long |
| level | 告警等级、严重程度  1：提示  2：一般  3：严重 | Integer |
| status | 告警处理状态  1：未处理  2：已处理 | Integer |
| recoverTime | 告警恢复时间 | Long |
| isRecover | 告警是否恢复 | Boolean |
| source | 告警来源 | Integer |
| type | 告警类型 | Integer |
| content | 告警内容 | String |
| appendix | 附加参数，如deviceId等 | String |

表4-11为告警管理主要实现的接口。

表4-11 告警管理主要实现的接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| getAlarms | 获取告警信息。可以根据告警各字段进行查询符合条件的告警信息，并且该接口同时支持分页查询。 |

### MongoDB中间层

MongoDB作为ADWAN的第三方数据库，其主要用来储存ADWAN系统运行过程中的历史信息、日志等内容。在ADWAN中有许多模块的功能都使用到了MongoDB存储数据。因此MongoDB中间层的主要功能就是对外拉起、连接MongoDB服务，对内向其他模块提供统一的MongoDB连接。

拉起、连接MongoDB是通过配置文件进行参数配置的。表4-12是MongoDB中间层的配置参数描述。

表4-12 MongoDB中间层的配置字段描述

|  |  |
| --- | --- |
| 参数名称 | 描述 |
| isStartByAdwan | 是否由ADWAN拉起MongoDB服务 |
| MongoDBPath | MongoDB所在目录 |
| dbPath | MongoDB数据文件目录 |
| Port | MongoDB服务监听端口 |
| Host | MongoDB服务所在的服务器IP |
| isStartAuth | MongoDB服务是否开启密码验证 |
| User | MongoDB用户名 |
| Passwd | MongoDB密码 |
| hostUser | 当配置的MongoDB是远程服务器上的服务时的远程服务器的用户名 |
| hostPasswd | 当配置的MongoDB是远程服务器上的服务时的远程服务器的密码 |

图4-6为mongo.properties配置示意图。

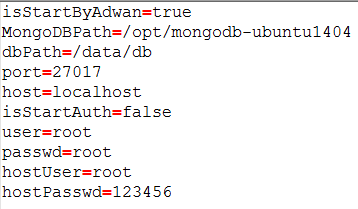


图4-6 mongo.properties配置示例

当需要拉起远程服务器上MongoDB服务时，需要用户配置hostUser、hostPasswd以用来通过SSH登陆到远程服务器上执行拉起命令，这里采用第三方开源的SSH库jSch实现的SSH连接。

在前文中介绍过MongoDB的特性中有关其内存使用的介绍，由于大多数情况下，MongoDB服务和ADWAN控制器处于同一个服务器上，因此在我们拉起MongoDB服务时，会专门的选择WiredTiger数据存储引擎，并配置其内存使用量：“--storageEngine wiredTiger --wiredTigerCacheSizeGB 1”。

表4-13是MongoDB中间层的主要实现接口。

表4-13 MongoDB中间层模块的主要接口

|  |  |
| --- | --- |
| 接口名称 | 描述 |
| getMongoDBClient | 返回一个MongoDBClient。其中这个MongoDBClient为一个单例模式。 |

### 流量调度

ADWAN作为一款SDN产品，其流量调度功能是其重要功能。在ADWAN中，目前主要有一下四种调度策略：

1. 基于带宽利用率调度，双链路上行。在链路带宽满足需求的情况下，该链路上业务流量正常行驶。当该链路带宽达到设定的阈值时，ADWAN便会触发选路操作，将该链路上的一些业务进行分流调节到其他链路上。
2. 基于延时调度，双链路上行。在链路延时满足需求的情况下，该链路上业务流量正常行驶。当该链路延时达到设定的阈值时，ADWAN便会触发选路操作，将该链路上的一些业务进行分流调节到其他链路上。
3. 基于丢包调度，双链路上行。在链路丢包率满足需求的情况下，该链路上业务流量正常行驶。当该链路丢包率达到设定的阈值时，ADWAN便会触发选路操作，将该链路上的一些业务进行分流调节到其他链路上。
4. 基于时间段调度，双链路上行。在设定的时间段范围内有限保障高优先级的业务。

通过各类不同的调度策略和业务组合以满足用户的不同需求。例如像一些低延迟的游戏业务，可以选用基于延时的调度策略来向用户提供更好的服务。

### 用户界面

基于ODL框架的ADWAN，其后台Rest接口对于普通用户来讲并不友好。因此用户界面主要解决的问题就是通过一种便捷的、直观的方式向用户展示数据并传递用的操作。

这里用户界面主要使用了node.js搭建了一个web服务器，并实现了一个代理接口解决Ajax异步请求的跨域问题。前端通过html5、JavaScript、jQuery、Ajax、AngularJS、Hightcharts等技术做页面展示和数据传输。

图4-7为ADWAN前台拓扑页面的展示图。

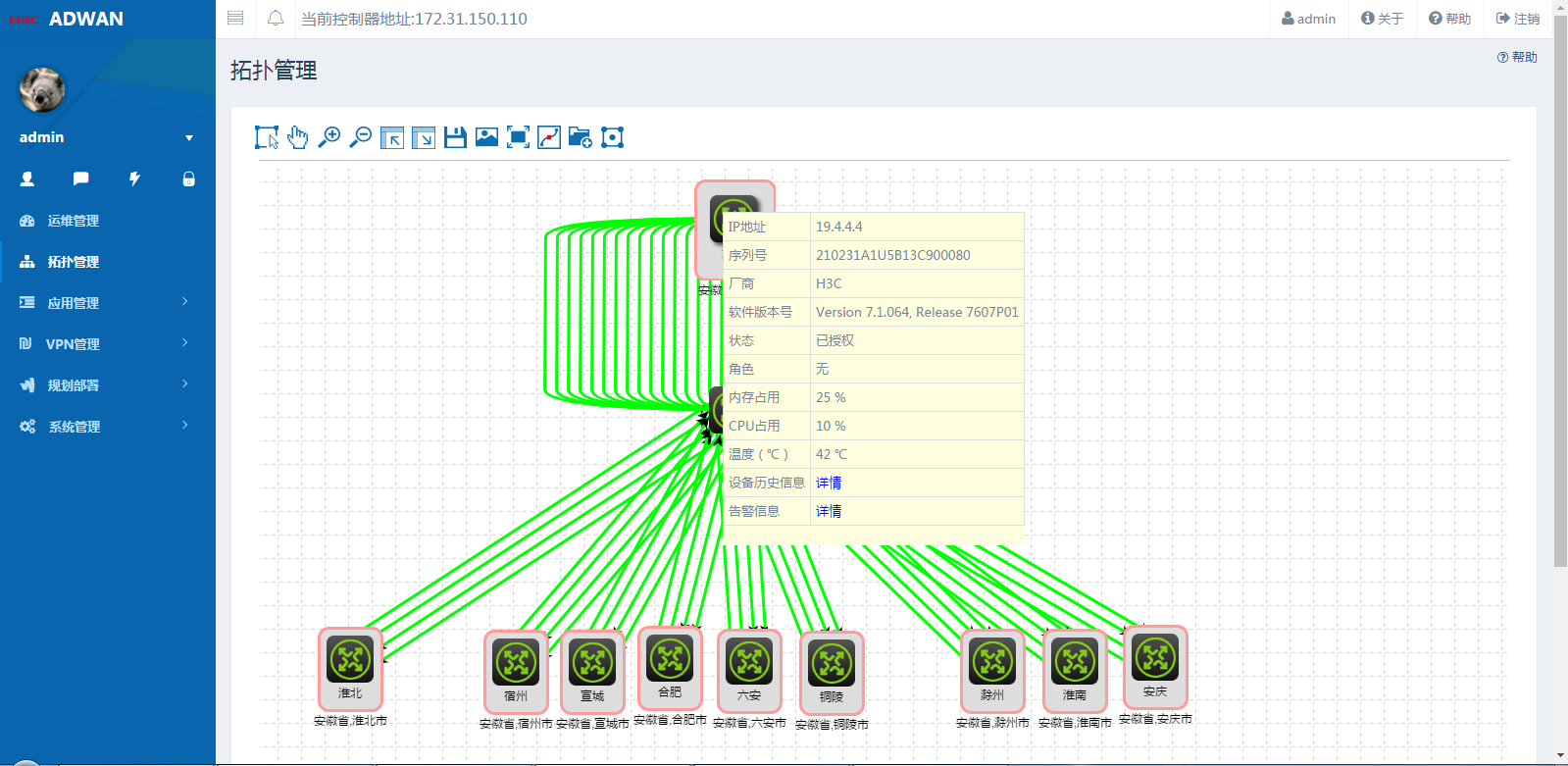


图4-7 拓扑页面

图4-8为ADWAN前台链路历史信息页面展示图。

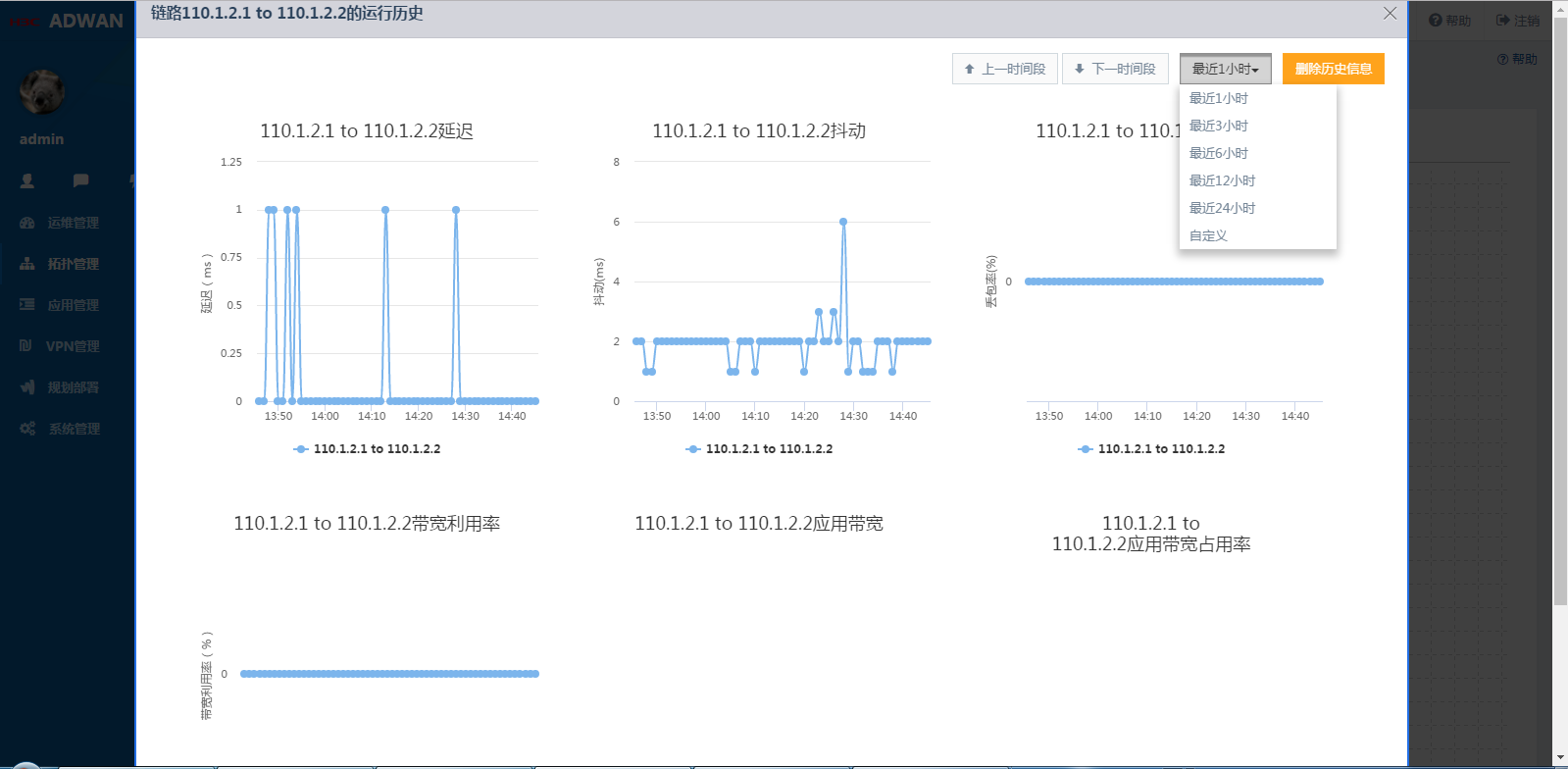


图4-8 link上的历史信息

## 历史信息数据库实现

在应用信息的获取与呈现子项目中，历史数据的存储采用了第三方数据库MongoDB存储。为了提升查询、插入的效率，采用了分层、分集合的数据存储设计。

例如在获取指定link上的应用组带宽历史信息时，用户通过接口查询在指定时间段内，经过指定链路的应用组带宽历史信息，历史信息的最小粒度为1分钟，开始时间的默认值为结束时间前一个小时，结束时间的默认值为当前时间，时间格式为“年-月-日 小时-分钟-秒 时区”，如2016-04-19 16:10:00 GMT+8。返回值的时间所在时区为输入时间时区，startTime和endTime的时区必须相同。返回值的粒度是根据输入时间段大小来确定的，表4-14为请求参数时间段与响应数据的时间粒度对应表。

表4-14 请求参数时间段与响应数据的时间粒度对应表

|  |  |
| --- | --- |
| 输入时间段长度 | 输出粒度大小 |
| 0-3小时 | 1分钟 |
| 3-24小时 | 10分钟 |
| 1-7天 | 6小时 |
| 7-30天 | 24小时 |
| 大于30天 | 7天 |

根据对应表的关系，在ADWAN运行的时候系统内部会有一个专门的定时器，该定时器会在特定的时间间隔自下而上聚合每一层级的数据。

同时在插入数据时，ADWAN还会检测当前时间，针对不同的季度，将数据插入不同的季度集合内，并定期删除超过四个季度以外的数据集合，以达到控制磁盘使用量和提高插入、查询的效率。

在数据库的实现中，主要利用了Java中的定时器、线程操作、函数式编程以及MongoDB的聚合功能。图4-9为分层模型示意图。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7天（7个24小时聚合） | | | | |
| 24小时（4个6小时聚合） | | | | … |
| 6小时（36个10分钟聚合） | | | … | … |
| 10分钟（10个1分钟聚合） | | … | … | … |
| 1分钟 | … | … | … | … |

图4-9 分层模型示意图

## 本章小结

作为基于ODL框架的一款SDN产品，ADWAN通过模块化、标准化等手段实现了高内聚、低耦合的软件设计思想。其通过各模块互相协作，共同实现了整合全网资源、多维观测网络状态、智能调度应用流量等功能，使整个网络实现了多层次、全方位的可视化，成功的实现了应用驱动广域网服务的目标。

# 总结与展望

本章主要对整个项目进行总结，并探讨一下ADWAN未来的发展前景。阐述了目前ADWAN研究的基本内容、方法以及取得的成果，并对现方案中存在的缺陷和需要改进、完善的地方进行探讨。

## 全文总结

ADWAN是一个具有分层、开放、灵活等特点的标准的SDN系统。如图5-1所示，ADWAN的三层次网络架构。

图5-1为ADWAN的方案架构图。



图5-1 ADWAN架构

其通过基于ODL控制器，ADWAN屏蔽了底层的不同协议、不同设备的差异，向上层应用提供了统一的、标准的Rest接口。利用BGP-LS、NetConf、SNMP等多中协议和设备进行交互，通过集中式的控制器方案架构来对网络设备进行集中管理、调度，实现了应用驱动的广域网服务，使网络具备灵活编程、动态感知及自动编排的能力，可以快速的满足不断变化的应用需求。

## 展望

虽然ADWAN相对传统网络来讲具有很多的优势，是未来网络发展的方向，但其目前依然存在一系列需要解决、完善的问题。

1. 计算资源不足，压力大，ADWAN系统的稳定性需要继续提高。作为集中式的SDN控制器，ADWAN在极端条件下需要为每一条流都进行定制化的优化策略，在流量复杂的情况下其对CPU的运算压力可想而知。并且这种压力会随着控制器控制的网络设备的增多而呈几何级的上升。因此如何解决、优化软件的设计复杂度、运算效率的问题，是未来ADWAN需要面临的一项挑战。
2. 目前ADWAN对华三公司的设备支持性最好。然而现网中的网络设备千千万万，在未来的开发中，ADWAN还需要加大对厂商的设备的兼容性开发，使ADWAN真正称为一款通用的SDN产品。
3. 网络设备集中控制的安全性隐患。目前ADWAN采取的时集中式的控制架构，这就存在着当控制器服务器瘫痪时，造成的整个网络瘫痪的风险。未来的开发中，ADWAN需要在服务器集群、数据备份等方面加大开发力度。

通过上面问题的分析考察，ADWAN的发展还需要面临一些问题，但是不可否认的是ADWAN目前已经实现了基本的SDN功能，可以并且已经在一些实际环境中进行了商用。相信以后通过技术的不断演进，ADWAN的功能会更加完善、稳定，在未来各家公司商用的SDN产品中大放异彩。

# 致谢

随着硕士学位论文的即将完稿，我的校园生活也即将踏入尾声，回首往昔，历历在目，在华科攻读硕士研究生的这段时光里，我得到了许多老师、同学们的帮助，非常高兴在这里成长。

首先需要感谢的是我的导师黄立群教授。黄老师是一名教学严谨的老师，在学生的培养上能够针对不同的学生特点指定不同的培养方针，因材施教。在黄老师的培养下，我们每个学生都得到了很大的成长。同时黄老师还是一个宽于待人老师。他给予了我们宽松的学习环境，让我们根据自己的兴趣特点选择不同的专业方向进行研究发展，这让我和同窗的研究生生涯中受益匪浅。在论文完稿之际，向我的导师致以最诚挚的敬意。

其次需要感谢软件学院的各位领导和老师们。正是在学院的领导和老师的帮助下，我们可以顺利的完成学业，在这里学习到了很多软件工程相关的知识，这在我找实习、找工作的过程中打下了良好的基础。

还要感谢我的父母。在这十几年的求学生涯中，无论经历了什么，他们都是我身后坚强的后盾，给予我最厚重的爱，正是由于他们的无私的支持和奉献，我才能如愿完成今日的学业。谢谢！

最后感谢各位同学的陪伴，感谢学校提供的平台。

樊 开

2017年12月11日

# 参考文献

1. 邓书华, 卢泽斌, 罗成程,等. SDN研究简述[J]. 计算机应用研究, 2014(11):3208-3213.
2. 郑毅, 华一强, 何晓峰. SDN的特征、发展现状及趋势[J]. 电信科学, 2013, 29(9):102-107.
3. Tu R, Wang X, Yang Y. Energy-saving model for SDN data centers[J]. Journal of Supercomputing, 2014, 70(3):1477-1495.
4. Tourrilhes J, Sharma P, Banerjee S, et al. SDN and OpenFlow Evolution: A Standards Perspective[J]. Computer, 2014, 47(11):22-29.
5. Kotronis V, Gamperli A, Dimitropoulos X. Routing Centralization Across Domains via SDN: A Model and Emulation Framework for BGP Evolution[J]. Computer Networks, 2015, 92(P2):227-239.
6. 张朝昆, 崔勇, 唐翯祎,等. 软件定义网络(SDN)研究进展[J]. 软件学报, 2015, 26(1):62-81.
7. 孙明海. 文明冲突与行业新秩序的重建[J]. 通信管理与技术, 2015(1).
8. 刘旭, 朱浩, 马可. 移动网“云”平台发展前瞻[J]. 电信网技术, 2014(12):26-31.
9. 王信龙. SDN和NFV标准进展[J]. 现代电信科技, 2013(5):49-55.
10. 田长星. SDN理论及其在金融业应用前景分析[J]. 金融科技时代, 2014(7):65-67.
11. Mckeown N, Anderson T, Balakrishnan H, et al. OpenFlow:enabling innovation in campus networks[J]. Acm Sigcomm Computer Communication Review, 2008, 38(2):69-74.
12. 黄海峰. SDN成IDC云化“催化剂”[J]. 通信世界, 2014(12):36-36.
13. 饶少阳, 陈运清, 冯明. 基于SDN的云数据中心[J]. 电信科学, 2014(8):33-41.
14. Ortiz S. Software-Defined Networking: On the Verge of a Breakthrough?[J]. Computer, 2013, 46(7):10-12.
15. Steele G L, Gabriel R P. The evolution of Lisp[J]. Nature Reviews Drug Discovery, 1993, 1(11):845-6.
16. Cesare Pautasso. RESTful Web service composition with BPEL for REST[J]. Data & Knowledge Engineering, 2009, 68(9):851-866.
17. 许名广, 刘亚萍, 邓文平. 网络控制器OpenDaylight的研究与分析[J]. 计算机科学, 2015, 42(s1).
18. Tavares A L C, Valente M T. A gentle introduction to OSGi[J]. Acm Sigsoft Software Engineering Notes, 2008, 33(5):1-5.
19. Mahajan R, Wetherall D, Anderson T. Understanding BGP misconfiguration[J]. Acm Sigcomm Computer Communication Review, 2002, 32(4):3-16.
20. Case J, Mccloghrie K, Rose M, et al. Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)[J]. Journal of the Air & Waste Management Association, 1996, 58(8):1057-1069.
21. Fielding R T, Kaiser G. Apache, the HTTP server project[J]. IEEE Internet Computing, 1997, 1(4):88-90.
22. Goldman M, Katz S. MAVEN: modular aspect verification[C]// International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems. Springer-Verlag, 2007:308-322.
23. 江日念, 林霞, 乔德新. Maven在Java项目中的引入及应用[J]. 电脑知识与技术, 2013(21):4842-4847.
24. Cohen, Edith, Kaplan, et al. Labeling dynamic XML trees[J]. Siam Journal on Computing, 2002, 39(5):271-281.
25. 刘兴宇. 冒烟自动化测试系统的设计与实现[D]. 北京交通大学, 2014.
26. 薄奇, 许林英. Spring框架中IoC的实现[J]. 微处理机, 2008, 29(1):147-149.
27. Jiang W, Zhang L, Liao X, et al. A novel clustered MongoDB-based storage system for unstructured data with high availability[J]. Computing, 2014, 96(6):455-478.
28. Banker K. MongoDB in Action[M]. Manning Publications Co. 2011.
29. Izquierdo J L C, Cabot J. JSONDiscoverer : Visualizing the schema lurking behind JSON documents[J]. Knowledge-Based Systems, 2016, 103:52-55.
30. Michael Stonebraker, Rick Cattell. 10 Rules for Scalable Performance in Simple Operation' Datastores[J]. Communications of the Acm, 2011, 54(6):72-80.
31. Plugge E, Membrey P, Hawkins T. The Definitive Guide to MongoDB[J]. 2016.
32. Schmidt J. Quelloffene NoSQL-Datenbank MongoDB 3.0 öffnet sich neuen Speicher-Engines[J]. 2015.
33. Bernstein P A, Goodman N. Multiversion concurrency control—theory and algorithms[J]. Acm Transactions on Database Systems, 1983, 8(4):465-483.
34. 郑子伟, 陈奋. HTML5离线应用开发技术研究[J]. 厦门城市职业学院学报, 2015(1):71-74.
35. 刘华星, 杨庚. HTML5——下一代Web开发标准研究[J]. 计算机技术与发展, 2011, 21(8):54-58.
36. 曹杰, 袁媛. 云计算环境下的浏览器[J]. 信息与电脑:理论版, 2011(11):139-140.
37. Langmann, R, Meyer, Laurid. Architecture of a web-oriented automation system[J]. IEEE, 2013:1-8.
38. 尤胜. HTML5的移动互联应用初探[J]. 电子世界, 2014(4):2-3.
39. Anthes G. HTML5 leads a web revolution[J]. Communications of the Acm, 2012, 55(7):16-17.
40. 吴磊, 张福庆. 基于HTML canvas的WebGIS客户端技术研究[J]. 地理信息世界, 2009, 7(3):78-82.
41. Stonebraker M. SQL databases v. NoSQL databases[J]. Communications of the Acm, 2010, 53(4):10-11.