基于SDN下的应用信息的获取与呈现的设计与实现

# 摘要

[摘要 2](#_Toc498678858)

[1. 绪论 1](#_Toc498678859)

[1.1. 研究背景 1](#_Toc498678860)

[1.2. 研究目的和意义 2](#_Toc498678861)

[1.3. 国内外相关研究情况 2](#_Toc498678862)

[1.4. 本文的主要研究内容 2](#_Toc498678863)

[2. ADWAN应用信息分析相关技术分析 2](#_Toc498678864)

[2.1. ADWAN（Application-driven Wide Area Network） 2](#_Toc498678865)

[2.2. SDN控制器：ODL（OpenDaylight） 3](#_Toc498678866)

[2.3. 报文统计技术 6](#_Toc498678867)

[2.4. 数据存储 11](#_Toc498678868)

[2.5. 应用信息报表的呈现 14](#_Toc498678869)

[2.6. 本章小结 14](#_Toc498678870)

[3. ADWAN系统分析与设计 15](#_Toc498678871)

[3.1. ADWAN系统设计 15](#_Toc498678872)

[3.2. 基础管理 17](#_Toc498678873)

[3.3. 设备管理 18](#_Toc498678874)

[3.4. 拓扑管理 20](#_Toc498678875)

[3.5. 流采集 22](#_Toc498678876)

[3.6. 运维管理 24](#_Toc498678877)

[3.7. 应用管理 25](#_Toc498678878)

[3.8. 告警管理 26](#_Toc498678879)

[3.9. MongoDB中间层 27](#_Toc498678880)

[3.10. 流量调度 28](#_Toc498678881)

[3.11. 用户界面 28](#_Toc498678882)

[3.12. 本章小结 28](#_Toc498678883)

[4. ADWAN系统实现 29](#_Toc498678884)

[4.1. ADWAN的运行环境 29](#_Toc498678885)

[4.2. 基础管理 30](#_Toc498678886)

[4.3. 设备管理 37](#_Toc498678887)

[4.4. 拓扑管理 40](#_Toc498678888)

[4.5. 流采集 42](#_Toc498678889)

[4.6. 运维管理 45](#_Toc498678890)

[4.7. 应用管理 47](#_Toc498678891)

[4.8. 告警管理 50](#_Toc498678892)

[4.9. MongoDB中间层 52](#_Toc498678893)

[4.10. 流量调度 53](#_Toc498678894)

[4.11. 用户界面 54](#_Toc498678895)

[5. SDN总结与展望 55](#_Toc498678896)

[5.1. 全文总结 55](#_Toc498678897)

[5.2. 展望 55](#_Toc498678898)

[致谢 55](#_Toc498678899)

[参考文献 55](#_Toc498678900)

# 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兴起，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，SDN（Software Defined Network，软件定义网络）技术愈发被人们重视研究了起来，其热度持续升温。

## 研究背景

长期以来，广域网主要负责各网络节点的互联互通，比如总部和分支之间，分支和分支之间，数据中心之间等，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：

* 业务部署慢，上线周期长：
  + 广域网设备分散，业务开通时需要逐台部署，手工配置，部署工作量很大；
  + 广域网业务众多，配置复杂，手工配置容易出错，开通周期长。
* 流量调度难，缺乏灵活性：
  + 由于缺乏整网视角，设备各自基于路由进行选路，选出来的最短路径而非最优路径，带宽利用率低；
  + 传统的策略路由和流量工程，局限性大，配置复杂，无法动态适应网络状态和应用需求的变化。
* IT维护人员的运维体验很差：
  + 网络管理手段有限，手工为主，对IT维护人员的技能要求较高；
  + 流量和业务无可视化呈现，造成故障无法快速识别和定位，运维难度大。
* 网络开放能力弱，无法适应业务对网络的要求：
  + 设备复杂，网络封闭，可编程能力弱，无法满足业务快速部署和灵活定制需求；
  + 网络和应用静态绑定，无法有效联动，难以提高云计算应用体验。

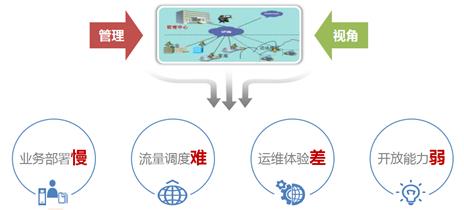


图1-1-1 传统广域网问题

## 研究目的和意义

### 云计算对广域网的需求

随着云计算的快速发展和大规模部署，企业IT已经从传统的数据中心向云计算数据中心转型，在这个过程中，用户对应用的体验需求是不会变的，用户希望访问云应用，能像访问本地应用一样快，一样安全，但流量模型发生了根本改变，对广域网的需求也发生了很大改变，主要体现在以下几点：

* 本地应用迁移到云端后，原来这些应用在本地运行，独享局域网带宽，现在变为云端运行，共享广域网带宽来进行数据交互，对广域网的带宽、承载能力、可扩展性、可靠性等都提出了更高的要求。
* 各地的本地应用都要通过广域网和云端进行通信，这个时候广域网的业务种类变得很多、流量也变得更加复杂，为了保证应用体验不受影响，需要面向不同用户实现差异化、精细化调度，保障各类用户的服务质量。
* 云计算数据中心的规模部署，数据中心之间的互联互通和数据同步越来越多，需要专门的DCI网络来承载和保障这些流量，同时要求DCI网络具备实时感知、统一编排、灵活调度，动态适应云计算应用的能力。

传统广域网架构复杂、扩展困难、封闭僵化，难以适应云计算的需求，需要广域网进行重构和变革。



图1-2-1 云计算对广域网的需求

### 运营商转型对广域网的需求

随着近些年数据中心、云计算、移动互联网的快速发展，用户除了基本上网需求外，对业务的丰富性和体验感要求越来越高，但目前运营商主要还是提供网络的互联互通，网络和业务基本上是割裂的，因此产生了很多突出的问题，如：

* 增量不增收： 业务流量呈现爆炸式的增长，但运营商发现收入并没有随着流量的增长而增长，反而和流量的剪刀差越来越明显。
* 网络成本高： 随着流量的增长需要不断扩容现有的网络，增加带宽，来满足用户的需求；新业务也要求购买和部署新的物理设备，而这些网络设备各自封闭的系统需要专门的人员来熟悉，技能难以复用，造成维护成本很高；专属的设备对机房、电源等的要求也有可能不同，也会导致运营成本的上升。
* 业务交付慢： 由于各设备系统封闭，缺乏标准的开放接口和自动化的部署工具，因此，每开通一个新业务，要求各设备之间配合设计、开发和调试，涉及到各个部门、不同厂商的协调，而且硬件设备需要挨个站点安装、升级和调试，整个周期很长。
* 竞争压力大：现有的网络结构复杂、设备封闭、缺乏灵活性，在面对虚拟运营商和互联网厂商的竞争时，压力会越来越大。

面对上述的这些问题，进行业务转型已经成为运营商的普遍共识，要进行业务转型，就需要解决以下几个主要问题：

* 网络架构的重构和优化，通过引入SDN技术和思想，搭建一个开放的网络平台，能够全局调控网络资源，以业务驱动网络，使网络从功能向服务转型，产生新的增值。
* 现网资源的利用和发挥，通过精细化的流量调度，为不同的用户提供差异化的网络服务，充分利用和挖掘现网资源，最大发挥整个网络的价值。
* 运维管理的简化和自动化，通过开放接口，开发一些丰富的运维应用，实现网络自动化、可视化，来简化运维。

总的来看，运营商现有网络难以适应业务转型的要求，除已经和正在进行SDN改造的数据中心网络外，也需要对城域、骨干等网络进行重构和变革。



图1-2-2 运营商转型对广域网的需求

### 广域网的变革已成为必然

为了应对传统广域网、云计算以及运营商转型中遇到的各种问题和挑战，广域网急需重构和变革，形成新一代的广域网，如下图所示，我们认为变革后的新一代广域网需要具备以下能力：

* 灵活的编程定制能力：可以基于用户业务进行灵活的定制编程；
* 全局的控制调度能力：管理系统具有整网视角，可以基于整网的流量进行调度和调整；
* 开放的网络构建能力：打破传统网络设备封闭的管理控制方式，快速的部署构建整个网络；
* 简化的运维部署能力：支持开放的接口，通过开放接口，开发一些丰富的运维应用，简化系统的运维和部署；
* 可视的状态呈现能力：实时监控资源变化，做到整个网络可视化，方便用户运维管理。

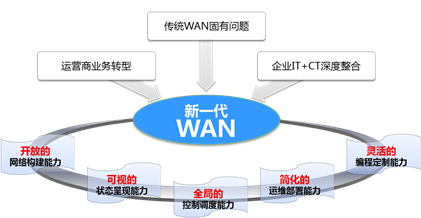


图1-2-3 新一代广域网示意图

## 国内外相关研究情况

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。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，并将策略应用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。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，设计了将传统的网络设备的数据转发和路由控制分离的模型，利用集中式的控制器通过标准化的接口对各类网络设备进行配置和管理，从而实现使网络资源的设计、管理和使用提供更多的可能性，更容易的推动网络的发展和革新。

经过多年的发展，开放网络基金会（open networking fundation，ONF）在2011年成立，该基金会致力于推动SDN的标准化，希望基于OpenFlow协议的SDN称为业界网络的新标准。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。包括华为、腾讯、中兴、ITU、IETF、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，华为和中国电信北京分公司共同宣布完成了全球首个运营商的SDN部署，成功将SDN技术应用到IDC网络。SDN技术，目前正在国内外迅速的发展。

## 本文的主要研究内容

随着SDN的发展，SDN已经由最初的以Openflow为代表的狭义SDN演变为以软件定义网络为核心的广义SDN，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：

软件定义网络：用软件定义的思想来设计新的网络架构，让网络能够主动适应用户业务和流量变化，而不是被动的承载流量。

应用驱动网络：以应用为源头，用自动化的方式驱动网络进行动态调整，快速满足实际的业务需求；

软件简化运维：通过软件应用参与网络的整个生命周期，实现自动化的业务部署、可视的网络呈现、灵活的流量调度，最终简化网络的管理运维。

# ADWAN应用信息分析相关技术分析

## ADWAN（Application-driven Wide Area Network）

ADWAN是H3C公司（新华三技术有限公司）的一款基于SDN框架ODL开发的新一代广域网解决方案。

ADWAN方案，基于全局视角，通过统一整合全网资源、多维观测网络状态、智能分析运行数据，使整个网络实现多层次、全方位的可视化，并根据用户策略和应用需求进行集中控制、全局调度、及实时调优，实现应用驱动的广域网服务，它具有以下特点：

* 全开放：首先是架构开放，网络的各个层次、不同组件是解耦的，做到网络可以很容易的扩展，也能很好的兼容多个厂商的设备；其次是接口开放，网络各组件之间通过开放、标准的接口来通信，从设备到控制器、APP提供多层次的、不同抽象度的API接口，赋予网络灵活的可编程可定义能力，能够让应用很容易的使用网络服务。
* 场景化：基于场景化的开发思路，提供面向业务的可定制APP应用，满足不同场景用户需求。
* 全流程：全流程重构广域网，整网视角管理和控制网络，简化运维管理。
* 端到端：业务驱动网络，基于业务应用的不同需求动态部署安全、WAN优化、CDN缓存等，提供端到端的网络服务。
* 可迁移：兼容传统网络，支持平滑迁移到SDN方案。

本项目既是基于ADWAN开发一个子模块功能，向用户提供应用信息查询的功能。通过符合ADWAN的编码规则，利用Maven、YANG模型使其嵌入进ADWAN中。

## SDN控制器：ODL（OpenDaylight）

### ODL概览

2013年，思科、微软、戴尔、惠普、IBM、英特尔、VMware等18个国际知名厂商联合建立了OpenDaylight项目。他们的共同目标是以透明、开放、公平、协作为原则建立一个供应商、客户、合作伙伴和开发人员可以共同使用的SDN开源平台，从而推动SDN的产品化和商业化。

ODL项目于2013年2月启动，2014年2月发布第一个正式版本，每隔8个月发布一个正式版本，截止目前为止已发布了氢（Hy）、氦（He）、锂（Li）、铍（Be）四个正式版本，能够为服务提供商和企业提供自动化服务交付、网络资源优化、云计算、NFV、以及区域网络的自动化、可见性和可控制相关的解决方案。

ODL控制器是一款基于Java开发的SDN控制器。其是SDN概念的一种具体实现，主要使用了以下工具：

**Maven**：ODL通过使用Maven来达到更加便捷的自动化构建项目的目的。

**OSGi**：是Java动态化模块化系统的一系列规范。ODL使用该OSGi框架以达到动态化的加载bundle和JAR包以及不同bundle之间的交换信息的目的。

**JAVA interfaces**：是同Java接口以达到事件监听、格式规范等目的。是一些bundle实现特定事件的回调函数的主要实现形式。

**REST APIs**：是ODL北向接口（northbound APIs）的接口形式。诸如拓扑管理、静态路由、主机跟踪等北向接口的接口形式。

ODL控制器通过向应用程序提供开放的基于web服务的北向接口来提供服务。

在南向（southbound）接口中，通过插件的形式提供诸如OpenFlow、BGP-LS等多种协议的支持。还有其他ODL捐赠者提供的一些特定的接口代码等模块。这些不同的模块功能都被动态的连接进了ODL的SAL（业务抽象层，Service Abstraction Layer）中。

SAL向北向接口提供服务。不管控制器底层使用的是什么协议、是什么网络设备，SAL都会清楚如何满足从北向传来的请求服务。对于控制器而言，它需要知道设备的功能、可达性等信息来控制其控制域中的设备。这些信息在拓扑管理（Topology Manager）模块中被存储和管理。

### ODL组织架构

图2-1-1 为ODL架构。

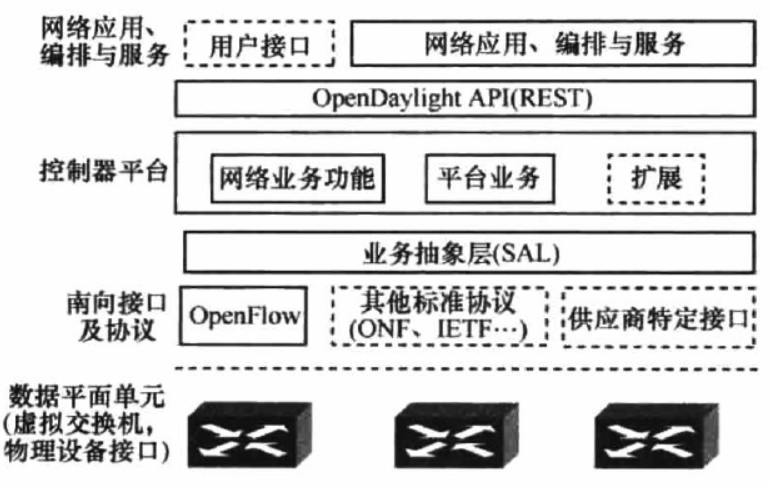


图2-2-1 ODL架构分析

可以看出，ODL有以下几大特点：

* 南向接口不仅支持OpenFlow，还支持SNMP、PCEP、BGP-LS等其他标准化协议，甚至允许出现私有化接口。
* 引入了SAL（业务抽象层），这使得上下层模块之间的调用可以相互隔离，屏蔽多种南向协议的差异，为上层的功能模块提供一致性服务。
* 北向提供开发可扩展的API，用户根据具体需要通过调用函数或者REST接口开发应用。
* ODL具有模块化、可扩展的控制器核心。采用OSGi（开放服务网关，Open Service Gateway Initiative）体系结构，解决功能组件之间的隔离问题，实现代码与功能的灵活加载，并可支持运行业务或应用的安装、更新、删除等插拔操作。
* 使用YANG工具可直接生成业务管理的“骨架”，开发者只需专注于具体业务，根据业务驱动模型工具来设计接口，实现业务功能。
* ODL拥有一个开源的分布式网络平台，该平台不仅能实现数据的存储、查找和监听，更重要的是它使得ODL支持控制器集群。

## 报文统计技术

本项目的主要功能：应用信息的获取与呈现，其第一步即为分析从设备上上报的流量信息。

目前行业内主流的报文统计协议有netstream、netflow和sFlow等技术。

Netstream技术是由华三公司提出的一种基于网络流信息的统计和发布技术。是一种利用网络设备（主要是交换机和路由器）进行网络流量统计、分析、打包的技术。因此在华为设备、华三设备中广泛使用了netstream技术。

Netflow技术是由思科公司在1996年设计出的，最开始时主要用于加速数据流的交换兵同步计量和统计正在高速转发中的IP数据流。随着技术的不断发展和迭代，netflow主要功能变为了对网络设备（主要是交换机和路由器）中的IP信息流进行测量和统计。思科设备主要使用的是netflow技术。

sFlow技术是InMon、Foundry和HP共同提出的一种网络流量监控技术。它可以提供完成整的数据链路层到传输层、甚至全网络覆盖范围内的流量统计信息。

由于笔者实习阶段是在华三公司，且参与的SDN项目开发都是基于华三设备，因此应用信息的获取与呈现这里使用的为netstream技术。因此下面着重介绍一下netstream技术。

### Netstream技术

Netstream提供报文统计功能，它根据IPv4报文的目的IP地址、目的端口号、源IP地址、源端口号、协议号、ToS（Type of Service，服务类型）和输入接口或输出接口来区分不同的流信息并针对不同的流信息进行专门的数据统计功能。

Netstream技术主要有三部分组成：NDE、NSC和NDA：

NDE（Netstream Data Exporter，网络流量采样）：主要功能是对网络流进行采集与发送采集到的流量信息。提取符合条件的流并进行统计，然后将统计到流信息发送至网络流量采集设备。一般网络设备如交换机、路由器等充当NDE的角色。

NSC（Netstream Collector，网络流量采集）：一般为运行在电脑上一个应用程序，负责收集和存储从NDE发送过来的netstream流信息。存储的netstream流信息可以作为网络流量分析程序的数据来源供其进行流量解析。NSC可以从多个NDE设备上接收数据并对数据进行统一的过滤、聚合和存储，供NDA进行网络流量解析。

NDA（Netstream Data Analyzer，网络流量分析）：即网络流量分析程序，它从NSC存储的流量信息中获取数据，然后对数据进行解析、归并、加工处理、生成报表，为其他业务提供支持。

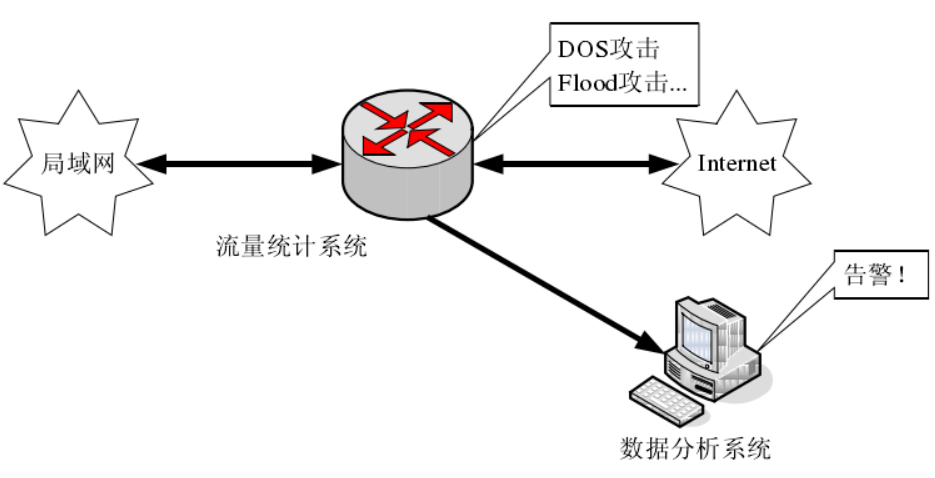


图2-3-1 netstream技术的典型应用

Netstream网络流老化是NSC输出流统计信息的前提。当设备上netstream缓存区达到符合输出条件时，NDE才会将流信息发送至NSC让其进行进一步的数据分析操作。

流老化主要有以下五种方式：

* 按活跃时间老化：当一条流的活跃时间超过所配置的时间限制时，流信息将会被老化。
* 按不活跃时间老化：当一条流的活跃时间超过所配置的时间限制时，流信息将会被老化。
* 按TCP连接FIN和RST报文触发老化：对于TCP连接，当有FIN或者RST报文发送时，说明一次会话的结束。因此，当一条已经存在TCP协议netstream流中流过一条有FIN或者RST标志的报文时，会立即该流老化。
* 按统计字节超过限制时老化：Netstream缓存区中的流需要记录流过的报文字节数，当字节数超过配置的大小时，就会造成流溢出，所以系统在检测到某条流的字节数超过限制时，为了避免发生计数错误，会立即对该流进行老化。
* 强制老化：管理员可以通过手工输入命令的方式对流进行老化处理。该功能主要是对未满足老化条件的流进行老化，比如未到老化时间但又需要最新的老化流来统计信息，或者netstream业务出现故障，需要对缓存区中的流进行全部老化处理。

当设备上的netstream缓冲区中达到流老化的条件时，NDE便会向NSC发送流信息，方便NSC对网络流量进行过滤、存储操作。

其中netstream流输出，分为三种方式：

* 原始流输出：设备上的所有的流信息都会被统计，然后在到达流老化的条件后，netstream缓冲区中的流全部输出到NSC。这样的输出方式可以详细的知道每条流的统计信息，但也会显著的消耗大量的计算资源和网络资源，同时在许多情况下，网络管理员并不一定需要这么全面详细的流统计信息。
* 聚合流输出：在原始流输出的基础上，以特定的方式对流进行分类和聚合。NDE向NSC发送经过聚合后的流统计信息。这种方式可以很大程度上减少网络设备的内存消耗、计算消耗和网络带宽的消耗。

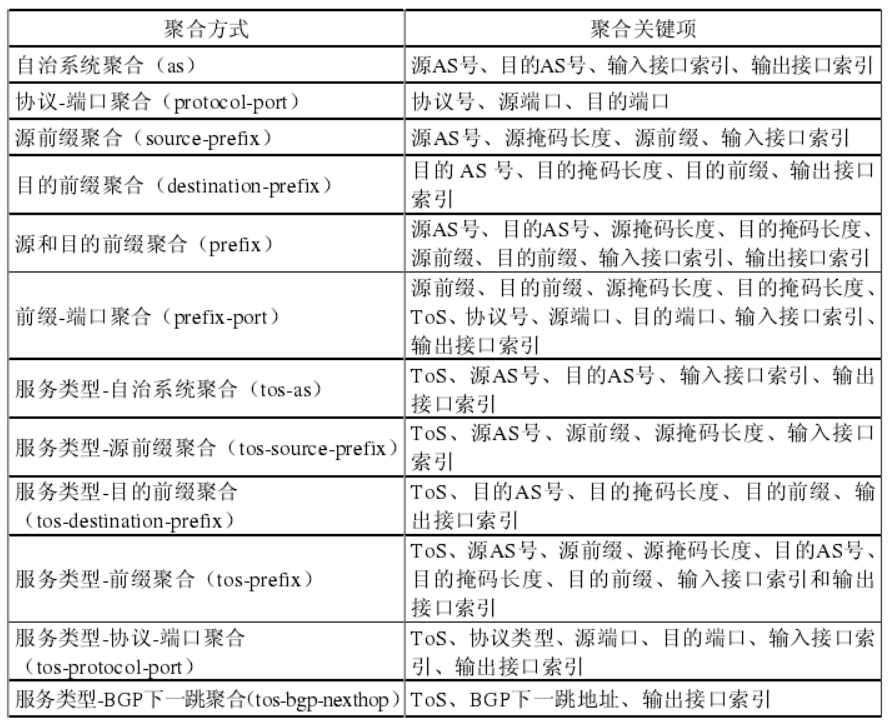


表2-3-1 聚合流输出的12种方式

设备根据选择的聚合方式和聚合关键项，可以将符合聚合条件的多条流聚合为一条聚合流，对应一条聚合记录。这12个聚合方式各自相互独立，互不影响，可以同时配置。

* 灵活流输出：是一种可以自定义形成流条件的流输出方式。这些条件可以是七元组中的任意组合。通过这种方式进行的流统计，可以得到网管人员最想要的流统计信息，并且也会在一定程度上减少网络带宽的消耗。

在NDE向NSC发送流信息报文时，会将流信息以用户指定的报文格式进行封装并以UDP的形式发送至NSC。NSC在接收到网络设备发来的报文后，以对应格式进行解析。下面是netstream的报文信息格式。

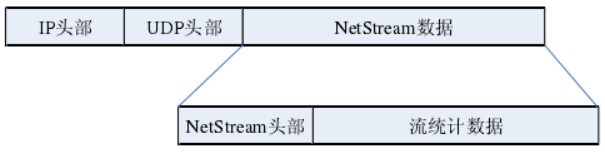


图2-3-2 Netstream报文格式

Netstream输出报文目前主要有V5、V8、V9几种版本：

|  |  |  |
| --- | --- | --- |
| 版本 | 优点 | 缺点 |
| V5 | * 输出的字段比较丰富，可以把聚合前的流记录的所有字段都输出给NSC * NDE设备负荷较小 | * 报文格式固定且不可扩展 * 数据量大，NSC无法长期保存，NSC和NDA分析压力大 |
| V8 | * 按一定规则进行聚合，数据量相对较小 * 承载内容略为简单，适合特定分析 * 可以增加新的聚合方式 | * 报文格式固定且不可扩展 * NDE设备完成聚合工作，负荷较重 * 如果增加新的聚合方式，需要NDE和NSC同时升级版本 |
| V9 | 基于模版   * 允许单独输出需要的域统计信息，减少了输出流的数据量进而减少了NDE、NSC可能的内存以及带宽消耗 * 不需要改变输出报文格式即可在输出记录中增加新的域，因为是以模版的形式输出所以即使NSC无法理解新增的域的真正语意，它仍然可以解释流记录 * 输出方式灵活，既可输出聚合前的流记录，又可输出聚合后的流记录 |  |

## 数据存储

通过netstream技术，控制器可以得到某台设备上的具体的应用流量信息。针对设备上报的netstream报文进行分析。通过分析我们可以获取到实时的流量信息。我们的系统支持历史应用信息的查询，因此还必须将实时的这种流量信息存储起来以便用户以后进行查询。

数据存储，主要是有文件存储和数据库存储两种形式。本文中使用的为数据库存储。

目前业内主流的数据库主要有两类，一类是关系型数据库，另一类是NoSQL型数据库。它们各有特点。

传统的关系型数据库是建立在关系模型基础上的数据库，借助于集合代数等数据概念和方法来处理数据库中的数据。通过SQL语句可以实现复杂查询以及事务支持。常见的MySQL、Oracle、SQLite等都属于关系型数据库。

NoSQL型数据库泛指非关系型数据库。主要分为四大类：键值对存储数据库、列存储数据库、文档型数据库和图形数据库。其数据模型相对于关系型数据库来讲比较简单，性能比较高，易于水平扩展。常见的有Redis、HBase、MongoDB等。

本文中的应用信息模型相对简单，没有复杂的耦合性。最终选择使用的数据库为MongoDB，文档型数据库。MongoDB是一个基于分布式文件存储的数据库，它支持的数据结构是一种类似于json形的bson数据格式。具有高性能、可扩展、易部署、易使用的特点。将netstream上报的报文解析后存储在对应的数据库集合内，以供用户查询历史应用信息的情况。

### MongoDB

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）。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。它和JSON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。一个或多个文档构成一个集合（collection），一个或多个集合构成一个数据库。



图2-4-1 document示意

document具有以下几个优点：

* document（即对象，object）可以对应很多编程语言中的原生数据类型；
* 通过嵌入式的document可以大幅度的减少关系型数据库表连接的消耗；
* Document动态的数据结构可以流畅的的支持数据的多态性。

MongoDB的关键特性：高性能（high performance）、功能强大的查询语言（rich query language）、高可用性（high availability）、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：

* 对嵌入式数据模型的支持减少了数据库系统层上的I/O活动。
* 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使数据聚合查询、文本搜索和地理位置查询像读写操作那样便捷。

**高可用性**

MongoDB具有备份功能，这称为副本集（replica set），它可以提供自动故障转移和数据的冗余备份功能。

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：

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，例如WiredTiger存储引擎和MMAPv1存储引擎。另外，MongoDB还提供了一种可插拔的存储引擎接口，允许第三方为MongoDB开发存储引擎。

### 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。它具有以下几个特点：

* Doucment Level Concurrency，文档级的并发控制。WiredTiger引擎针对写操作使用了文档级的并发控制。因此在多客户端可以同时更新同一个集合下的不同文档。
* 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。在一个操作开始之前，WiredTiger会提供一个涉及数据的时间点快照。快照呈现了内存数据的一致性。
* 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。
* Compression，数据压缩。WiredTiger提供集合、索引的压缩功能。此功能可以减小数据的存储占用空间，但会消耗额外的CPU计算资源。
* 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。

### MMAPv1存储引擎

MMAPv1存储引擎是MongoDB最开始时使用的一个基于映射文件的存储引擎。其擅长处理大量的插入、读取、更新的操作。

本项目采用的引擎为WiredTiger，MMAPv1不做过多的介绍。

## 应用信息报表的呈现

设备的具体应用信息存储在数据库中，其原始数据是不易于用户直接查看，晦涩难懂。因此需要对其进行专门的处理来提供给用户一个简单、高效的查询接口。同时，通过此接口返回的数据应进行响应的处理，使其在页面上以图文报表的形式显示，方便用户直观的查看信息。

这里主要使用的ADWAN（Application-driven Wide Area Network, H3C新一代基于ODL开发的广域网解决方案产品）的Rest接口的设置，定义输入参数、输出参数形式以及前台web页面的具体显示。

前台页面通过Ajax请求，使用html5绘图eChart框架显示。应给予用户最直观易懂的操作与信息。

## 本章小结

应用信息的获取与呈现，是基于SDN产品ADWAN的基础上进行开发子模块。SDN产品ADWAN是华三公司的一款基于ODL框架的新一代广域网整体解决方案，目的是为了帮助用户构建一个就架构开放、灵活编程、易于运维的广域网来承载日益丰富应用流量，最终实现应用按需驱动、网络动态适应应用的目的。而应用信息的获取与呈现也是为了这个目的而产生的模块。通过ODL框架的规范、Rest接口的定义，YANG模型的定义以及前台Web页面的请求与显示作用以达到最终目的。

# ADWAN系统分析与设计

应用信息的获取与呈现是基于ODL框架的ADWAN项目的一个子模块，其主要功能是向用户提供近一年以内任意时段内的网络流量情况，以供用户进行其他相关的数据分析、处理等操作。下面主要讲述ADWAN项目以及子项目应用信息的获取与呈现的系统分析与设计。

## ADWAN系统设计

H3C凭借长期在广域网领域的技术积累和丰富经验，深入调研用户实际应用中存在的痛点和需求，采用SDN思想及其相关技术开发了新一代广域网整体解决方案，这个方案帮助用户构建了一个架构开放、灵活编程、易于运维的广域网，来承载日益丰富的应用流量，最终实现应用按需驱动网络、网络动态适应应用，即ADWAN（Application-driven Wide Area Network）。

ADWAN系统应基于全局视角，通过统一整合全网资源、多维观测网络状态、智能分析运行数据，使整个网络实现多层次、全方位的可视化，并根据用户策略和应用需求进行集中控制、全局调度、及实时调优，实现应用驱动的广域网服务，它应具有以下特点：

* 全开放：首先是架构开发，网络的各个层次、不同组件的解耦的，做到网络可以很容易的扩展，也能很好的兼容多个厂商设备；其次是接口开放，网络各组件之间通过开放、标准的接口通信，从设备到控制器，APP提供多层次的、不同抽象度的API接口，赋予网络灵活的可编程可定义能力，能够让应用很容易的使用网络服务。
* 场景化：基于场景化的开发思路，提供面向业务的可定制APP应用，满足不同场景的用户需求。
* 全流程：全流程重构广域网，整网视角管理和控制我那个落，简化运维管理。
* 端到端：业务驱动网络，基于业务应用的不同需求动态部署安全、WAN优化、CDN缓存等，提供端到端的网络服务。
* 可迁移：兼容传统网络，支持平滑迁移到SDN方案。

### 方案架构



图3-1-1 ADWAN方案架构

如上图所示，ADWAN方案和其他场景下的SDN网络架构一样，也是一个分层、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：

* 网络设备层：网络设备接收SDN控制器的控制和管理，除了传统的SNMP、NETCONF、命令行等方式外，支持SDN架构下的BGP-LS、BGP Flowspec、PCEP、Openflow等协议，和控制器进行通信；同时在转发层面进行优化，支持Segment Routing、Openflow硬件转发，提供高性能的转发平面。
* 控制器+APP层：整个方案基于开源的ODL平台，支撑各种APP集成；根据广域网不同的场景，比如DCI网络、骨干网、分支接入等，开发定制化、场景化的APP，满足用户在不同场景下的网络需求；南向通过标准南向接口协议和设备互通；北向面向用户提供定制化的API接口，实现和编排系统集成，满足用户差异化的业务需求。
* 管理编排层：通过调用APP提供的API接口，实现业务流组定义（Qos优先级、五元组、VPN等）和业务的策略定义和管理编排，全网的实时监控、可视化呈现、及故障排查等，进而增强网络的可视化，简化网络的运维管理。

在基于ADWAN项目的基础上编码应用信息的获取与呈现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。下面分别介绍一些这些模块的系统设计及功能。

## 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。其主要需求如下：

基础管理的核心需求主要为：

* 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。
* 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。
* 网络场景的设置。ADWAN作为一款SDN产品，其部署的网络场景多种多样。因此ADWAN方案是基于场景化的思路，有针对的开发了不同的APP应用来满足运营商、企业/行业以及互联网等不同类型用户的需求。
* 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。
* 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：

* 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。
* 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。
* 设备故障、ADWAN系统重启时，恢复设备配置等。
* 当系统内某个设备出现异常时，可以向告警管理模块发出告警，提醒用户注意。

表3-3-1 设备管理数据库各字段及含义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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

ADWAN作为一款SDN产品，其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。其中链路信息的查看和管理即为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：

* 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。
* 拓扑管理。对网络拓扑进行增删改查等管理操作。
* 提供所有或某条链路的状态信息。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。
* 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。
* 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。

表3-4-1 拓扑数据库存储各字段及含义

|  |  |  |
| --- | --- | --- |
| 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3-4-2 链路数据库存储各字段及含义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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。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：

* 针对ADWAN系统内需要进行流量采集的链路下发流采集命令。
* 实时根据设备侧上报上的流量信息更新link上的带宽信息。
* 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。依托于向设备下发ACL，设备会采集指定应用的流量信息。

表3-5-1 ACL各字段及含义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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3-5-2 Rule各字段及含义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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3-5-3 IP-port各字段及含义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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。本文论述的应用信息的获取与呈现的对外接口就是运维管理模块提供支持。

表3-6-1 应用信息各字段及含义

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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。

由于本项目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。

## 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。

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。下表为应用管理字段示意。

表3-7-1 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3-7-2 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 |  |
| 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。

目前告警管理共有4大类告警信息，12类细分告警信息。下表为各告警信息的类型及阐述。

4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。 |

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。本项目的应用信息数据即存储在MongoDB中。

作为使用的第三方数据库，MongoDB中间层主要需求为：

* 可以连接本地或远程服务器上的MongoDB服务。
* 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。
* 对其他模块提供统一的MongoDB数据库连接接口。
* 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。而流量调度的核心则是选路算法。

流量调度这里主要功能需求为：

* 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。
* 下发设备配置，定义应用报文转发路径。
* 配置各类专线、VPN等业务。

## 用户界面

ADWAN是基于ODL框架实现的一个SDN产品。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页面设计通过Node.JS作为WEB服务的后台，通过架设代理访问ADWAN的Rest接口。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。

本模块的主要功能需求为：

* 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；
* 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；
* 通过AngularJS、jQuery、Highcharts等框架，使响应数据转化为用户直观可见的图文报表的形式。

## 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。

# ADWAN系统实现

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。它们相辅相成，共同实现了新一代的基于SDN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的运行环境

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。服务器的配置应满足以下推荐条件。

表4-1-1 ADWAN运行推荐配置

|  |  |  |
| --- | --- | --- |
| 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 |
| 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 |

## 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。

对不同厂商的网络设备，下发不同的命令配置，使用的是XML配置文件。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4-2-1 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。然后通过netConf报文下发至设备上。对应返回设备响应的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。

下图时小版本识别的XML配置图。



图4-2-2 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。

基础管理还负责用户的场景功能设置。目前ADWAN主要支持三类场景，分别是ADWAN行业纵向网、ADWAN骨干网/DCI和ADWAN分支/专线。三种场景具有三种不同的调度策略。

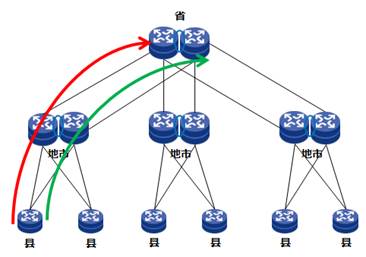


图4-2-2 场景一：ADWAN纵向网调度

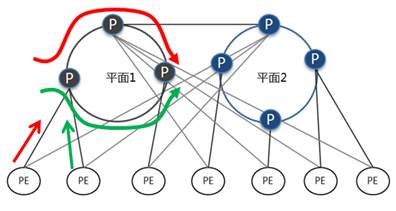


图4-2-3 场景二：ADWAN骨干网/DCI调度

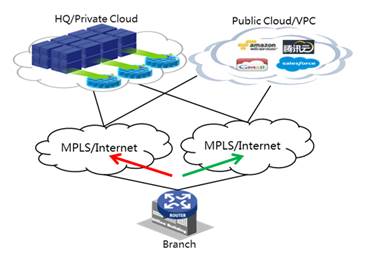


图4-2-4 场景三：ADWAN分支/专线调度

下面是ADWAN基础管理主要对外提供的接口信息。

### POST /setNetworkScene

【描述】：用户设置组网场景。根据用户不同的组网场景，采用不同的调度策略。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| sceneAttribute | 场景属性 | sceneType：场景类型，  networkType：网络类型 | 必选 |

### POST /bgp-config

【描述】：设置BGP的基本配置，修改后BGP邻居需要重新建立。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| asNum | AS号 | Integer | 必选 |
| bgpId | BGP ID值 | IPv4 Address | 必选 |

### POST /bgplsSetPeers

【描述】：设置BGP-LS多域邻居。采用通过BGP-LS自动发现设备时需要在设备上进行相应的BGP-LS配置。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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| bgplsArea | BGP-LS多域信息 | Object | 必选 |
| areaName | BGP-LS域名称 | String | 必选 |
| hostPeer | 主邻居Peer地址 | IPv4 Address | 必选 |
| standbyPeer | 备用邻居Peer地址 | IPv4 Address | 可选 |

### POST /addNetconf

【描述】：增加一条Netconf配置信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| netconfCfgName | Netconf配置名称 | String | 必选 |
| Protocol | Netconf承载的协议 | String | 可选，默认ssh |
| netconfSshPort | Netconf承载协议为ssh时的端口号 | Integer | 可选，默认830 |
| netconfUser | 设备netconf对应的用户名 | String | 必选 |
| netconfPasswd | 设备netconf对应的密码 | String | 必选 |
| isGlobal | 是否全局配置 | Boolean | 可选，默认全局配置 |

### POST /addSnmp

【描述】：增加一条SNMP配置信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| snmpCfgName | SNMP配置名称 | String | 必选 |
| snmpVersion | 设备支持的SNMP版本 | Integer | 必选 |
| snmpReadCom | SNMP读团体字 | String | 可选，snmpVersion为1或2时与snmpWriteCom至少配置一个 |
| snmpWriteCom | SNMP写团体字 | String | 可选，snmpVersion为1或2时与snmpReadCom至少配置一个 |
| snmpV3User | SNMP V3版本的用户名 | String | 可选，只在snmpVersion为3时可以配置，且和snmpV3AuthProtocol同时配置或同时不配置 |
| snmpV3AuthProtocol | SNMP V3版本认证算法  0：md5  1：sha  65535：无加密 | Integer | 可选，只在snmpVersion为3时可以配置，且与snmpv3AuthPassword同时配置或同时不配置 |

### POST /distributeConfigAsync

【描述】：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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| deviceId | 需要下发配置的设备标识 | String | 必选 |
| configName | 配置命令名称 | String | 必选 |
| Params | 配置内的各参数的键值对 | Map | 可选 |

## 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。

主要设计接口为以下：

### POST /addDeviceManual

【描述】：通过手动的方式增加一个网络设备

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| deviceName | 设备名称 | String | 必选 |
| manageIP | 设备的管理IP | IPv4 Address | 必选 |
| deviceType | 设备类型：   1. 通用设备 2. 虚设备 | Integer | 可选 |
| description | 该设备的设备描述 | String | 可选 |
| role | 通用设备角色（1-99） | Integer | 可选 |
| company | 厂商信息：  0：H3C  1：HP  2：HuaWei  3：Cisco  4：Juniper  65535：Unknow | Integer | 可选 |
| model | 设备型号，如MSR810 | String | 可选 |
| netconfCfgId | Netconf配置标识 | String | 可选 |
| snmpCfgId | SNMP配置标识 | String | 可选 |
| cityName | 设备所属城市名称 | String | 可选 |

### POST /updateDevice

【描述】：通过设备的唯一标识更新设备的一些状态信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| deviceId | 设备的唯一标识 | String | 必选 |
| manageIP | 设备的管理IP | IPv4 Address | 可选 |
| deviceType | 设备类型：   1. 通用设备 2. 虚设备 | Integer | 可选 |
| description | 该设备的设备描述 | String | 可选 |
| role | 通用设备角色（1-99） | Integer | 可选 |
| company | 厂商信息：  0：H3C  1：HP  2：HuaWei  3：Cisco  4：Juniper  65535：Unknow | Integer | 可选 |
| netconfCfgId | Netconf配置标识 | String | 可选 |
| snmpCfgId | SNMP配置标识 | String | 可选 |
| cityName | 设备所属城市名称 | String | 可选 |

### POST /deleteDevice

【描述】：通过设备的唯一标识删除ADWAN中的某一台设备。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| deviceId | 设备的唯一标识 | String | 必选 |

### POST /getDevice

【描述】：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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| deviceId | 设备的唯一标识 | String | 可选 |

### POST /getTerminalPoints

【描述】：获取ADWAN中所有设备的所有接口信息或者通过设备唯一标识获取该设备的接口信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| deviceId | 设备的唯一标识 | String | 可选 |

### POST /getTerminalPointById

【描述】：通过接口唯一标识获取该接口的状态信息

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| tpId | 接口的唯一标识 | String | 必选 |

## 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。

主要设计接口如下：

### POST /getTopo

【描述】：通过拓扑信息以及拓扑内的设备ID列表和链路ID列表。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| 无 | -- | -- | -- |

### POST /addLinkManual

【描述】：通过手动的方式增加一条链路。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkName | 链路名称 | String | 必选 |
| linkType | 链路类型 | Integer | 可选 |
| srcDeviceId | 源节点设备标识 | String | 必选 |
| srcTpId | 源节点接口标识 | String | 必选 |
| destDeviceId | 目的节点设备标识 | String | 必选 |
| destTpId | 目的节点接口标识 | String | 必选 |
| metric | 链路的metric值 | Integer | 可选 |
| bandwidth | 链路的带宽值 | Long | 可选 |
| reservableBandWidth | 链路的可分配带宽值 | Long | 可选 |

### POST /updateLink

【描述】：通过链路的唯一标识更新链路的一些状态信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | Strong | 必选 |
| linkName | 链路名称 | String | 可选 |
| linkType | 链路类型 | Integer | 可选 |
| srcDeviceId | 源节点设备标识 | String | 可选 |
| srcTpId | 源节点接口标识 | String | 可选 |
| destDeviceId | 目的节点设备标识 | String | 可选 |
| destTpId | 目的节点接口标识 | String | 可选 |
| metric | 链路的metric值 | Integer | 可选 |
| bandwidth | 链路的带宽值 | Long | 可选 |
| reservableBandWidth | 链路的可分配带宽值 | Long | 可选 |

### POST /deleteLink

【描述】：通过链路的唯一标识删除ADWAN中的某一条链路。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | String | 必选 |

### POST /getLink

【描述】：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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | String | 可选 |

## 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注册。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。

这里进行应用匹配的方式主要是通过五元组进行匹配的。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4-5-1 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4-5-2 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，不符合则进行下一个流组的匹配。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。

同时，在应用信息存储在MongoDB的过程中，为了取数据的快速性，我们还设置了存储模型，即前文所属的分集合、分层的数据存储模型。简单来讲，分集合模型即我们的数据存储采用的按季度存储的模式。每一季度会新建一个新的集合来存储每分钟、每10分钟、每6小时、每24小时的数据。这样做的操作是为了在用户取数据的时候不需要进行全年扫面，指进行对应季度或连续两个季度的数据扫描即可。而分层模型即除了存储原始的每分钟的数据外，我们会定期取一段时间的数据均值建立新的集合存储。即每10分钟、每6小时、每24小时、每7天的层次模型。根据用户查询的时间段的长短，从不同的层次模型获取数据，达到迅速响应的目的。

下面是流采集模块的主要接口。

### POST /registerFgInPhysicallIf

【描述】：注册应用接口，并同时指定应用组和物理口的对应关系。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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| flowGroupInstanceId | 流组实例ID | String | 必选 |
| collectInfos | 设备的接口 | List<String> | 必选 |
| isStoreHistoryData | 是否存储历史统计信息 | Boolean | 可选，默认flase |

### POST /registerLinkBandwidth

【描述】：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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | String | 必选 |

## 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。

在应用信息的获取与呈现中，主要提供的接口如下：

### POST /getLinkQuality

【描述】：获取link质量信息，用户通过此接口查询链路质量信息（延时、抖动、丢包率）。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | String | 必选 |

### POST /getLinkBandwidth

【描述】：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）。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | String | 必选 |

### POST /getLinkFlowgroupBandwidth

【描述】：获取指定link上的应用组带宽信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | String | 必选 |
| topN | 输出流组信息的个数 | Integer | 可选 |

### POST /getHistoryLinkFg

【描述】：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：

|  |  |
| --- | --- |
| 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 |

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| linkId | 链路的唯一标识 | String | 必选 |
| startTime | 统计的开始时间 | String | 可选 |
| endTime | 统计的结束时间 | String | 可选 |

### POST /clearHistoryData

【描述】：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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| dataType | 要删除的历史信息类型。  0：全部历史信息  1：链路历史信息  2：设备历史信息  3：应用历史信息 | Integer | 必选 |

### POST /getDeviceRunStatus

【描述】：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。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| deviceId | 设备的唯一标识 | String | 必选 |

## 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。

其主要实现的接口如下：

### POST /addFlowgroup

【描述】：增加一条流组信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| flowgroupName | 流组名称 | String | 可选 |
| flowIds | 流标识列表 | List<String> | 可选 |
| networkScopeId | 网络范围标识 | String | 可选 |
| scheduleMode | 调度方式：  0：自动  1：自定义  2：不调度 | Integer | 可选 |
| pathMode | 路径方式：  0：自动  1：自定义 | Integer | 可选 |
| backupBandwidthPercent | 备用路径占带宽和主路径占带宽之间的百分比 | Integer | 可选 |
| pathNum | 选路条数  1：代表单路径  2：代表主备路径 | Integer | 可选 |
| comment | 备注 | String | 可选 |

### POST /deleteFlowgroup

【描述】：删除一条流组。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| flowgroupId | 流组的唯一标识 | String | 必选 |

### POST /addFlow

【描述】：新增一条流信息。

【请求消息参数】：

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

### POST /addNetworkScope

【描述】：设置一个网络范围信息。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| networkScopeName | 网络范围名称 | String | 可选 |
| subNetworkScope | 子网 | List<subNetworkScope> | 必选 |

表4-7-1 subNetworkScope字段

|  |  |  |
| --- | --- | --- |
| 字段 | 描述 | 类型 |
| srcDevices | 源节点 | List<String> |
| dstDevices | 目的节点 | List<String> |
| includeDevices | 路径需要经过的节点标识 | List<String> |
| includeLinks | 路径需要经过的链路标识 | List<String> |
| excludeDevices | 路径需要排除的节点标识 | List<String> |
| excludeLinks | 路径需要排除的链路标识 | List<String> |
| commet | 备注 | String |

## 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ID、告警信息状态等参数确定告警的显示状态。主要告警参数如下表。

表4-8-1 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 |

其主要实现的接口如下：

### POST /getAlarms

【描述】：获取告警信息。可以根据告警的参数进行查询符合条件的告警信息。同时支持分页返回。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| startTime | 开始时间 | Long | 可选 |
| endTime | 结束时间 | Long | 可选 |
| pageNum | 第几页的告警信息 | Integer | 可选 |
| numOfPage | 每页显示几条告警信息 | Integer | 可选 |
| alarmId | 告警标识 | String | 可选 |
| level | 告警等级、严重程度 | Integer | 可选 |
| status | 告警处理状态 | Integer | 可选 |
| recoverTime | 告警恢复时间 | Long | 可选 |
| isRecover | 告警是否恢复 | Boolean | 可选 |
| source | 告警来源 | Integer | 可选 |
| type | 告警类型 | Integer | 可选 |
| content | 告警内容 | String | 可选 |
| appendix | 附加参数 | String | 可选 |

## 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。

|  |  |
| --- | --- |
| 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 |

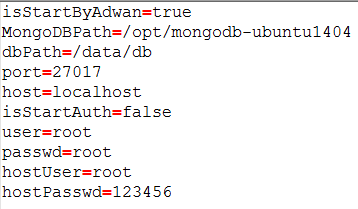


图4-9-1 mongo.properties配置示例

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SSH库jSch实现的SSH连接。

在前文中介绍过MongoDB的特性。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”。

MongoDB中间层的主要实现接口如下：

### POST /getMongoDBClient

【描述】：返回一个MongoDBClient。

【请求消息参数】：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 类型 | 备注 |
| 无 | -- | -- | -- |

## 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：

* 基于带宽利用率调度，双链路上行。正常情况下两种业务走在同一链路上，但是随着带宽利用率的逐渐增加，当链路的利用率达到设定的阈值时，控制器会下发相应的策略，将一条链路的业务调节到另一条链路上。
* 基于延时调度，双链路上行。正常情况下两种业务走在同一条链路上，但是当一条链路上的延时超过用户所设置的阈值时，控制器会下发相应的策略，将一条链路上的业务调节到另一条链路上。
* 基于丢包调度，双链路上行。正常情况下两种业务走在同一条链路上，但是当一条链路上的丢包率超过用户所设置的阈值时，控制器会下发相应的策略，将一条链路上的业务调节到另一条链路上。
* 基于时间段调度，双链路上行。周期性的业务调度保障业务需求、某一时间段对高优先级业务可靠性保证。

基于策略我们可以和不同的业务进行组合，满足不同的客户场景需求。例如对于视频会议，我们可以开启延时策略加丢包策略加带宽利用率即可。

## 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。

页面展示：

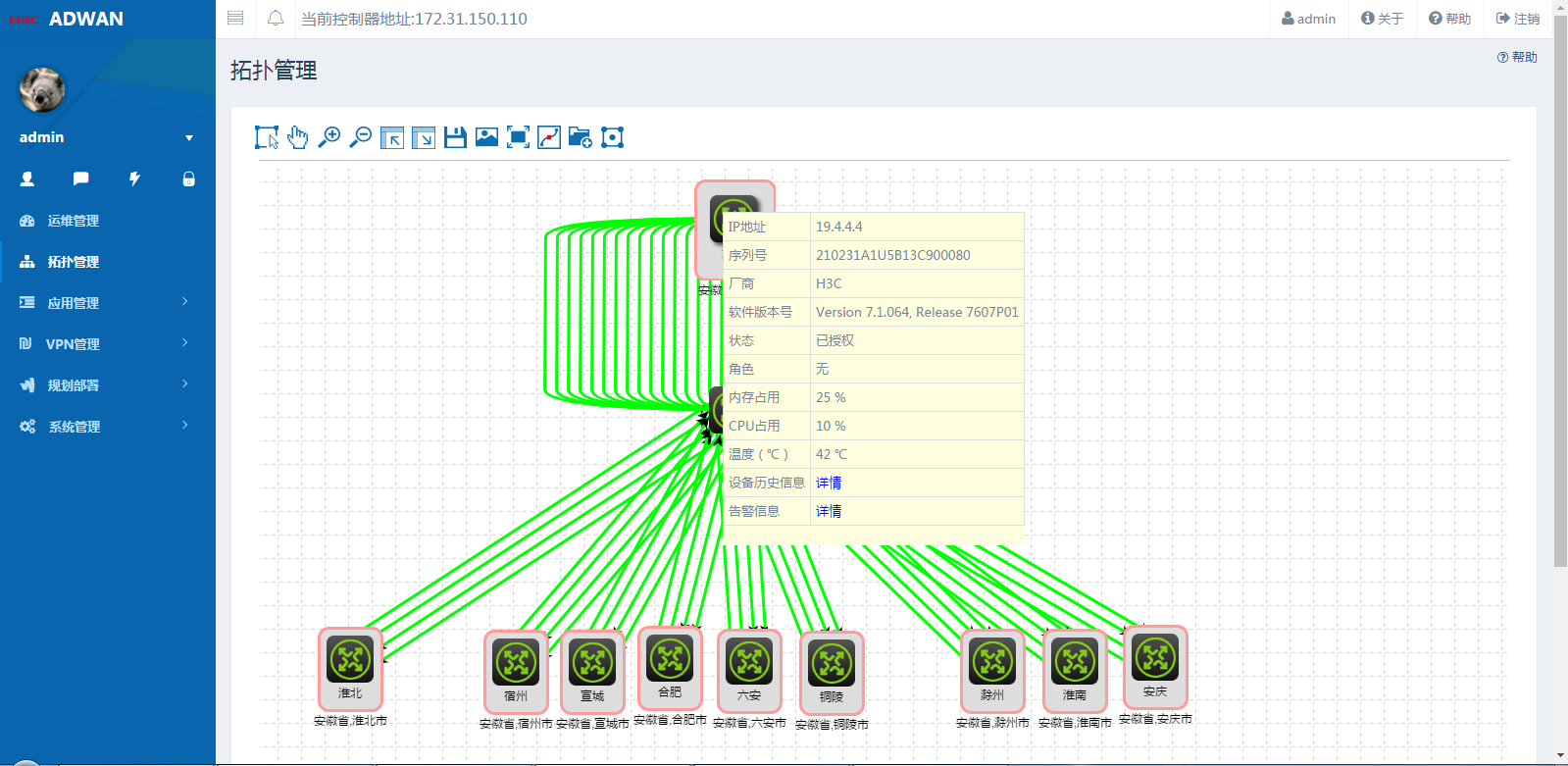


图4-11-1 拓扑页面

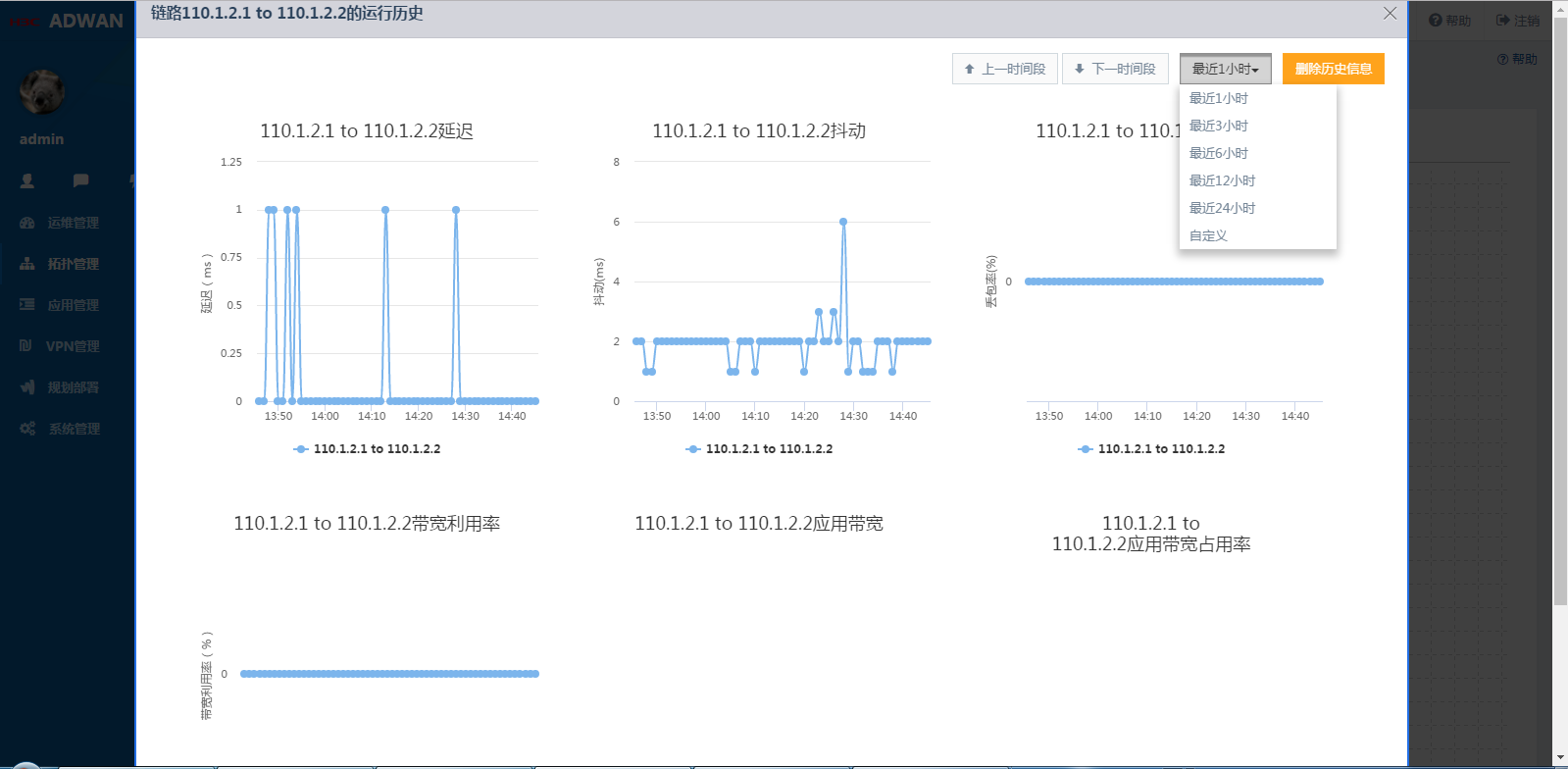


图4-11-2 link上的历史信息

## 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。

# SDN总结与展望

本章主要对整个项目进行总结，并探讨一下SDN未来的发展展望。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方案是一个标准的SDN网络架构，同时也是一个分层、开放、灵活的网络架构，如下图所示，整个ADWAN 方案分为网络设备、控制器+APP、管理编排三个层次。



图5-1-1 ADWAN架构

其通过基于OpenDayLight框架，屏蔽了底层不同协议、不同设备的差异，向上层提供统一的、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。

* 运算压力大，软件设计复杂，系统稳定性存在隐患。作为集中式的SDN控制器，极端条件下其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、SDN产品需要面临的一项挑战。
* 控制器接口标准尚未标准化。目前主流的网络设备厂商都在开发自己的SDN产品。ADWAN作为华三公司的一款应用驱动广域网的SDN产品，其最佳的搭配模式是和己公司的产品进行协调。目前各大网络设备公司的SDN产品接口各自为政，尚未实现标准化，这在一定程度上延缓了SDN商用化的进程。
* 网络设备集中控制的安全性隐患。SDN产品采取的时集中式的控制架构，ADWAN也不例外。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。

通过上面问题的分析考察，SDN未来的发展还需要面临一些问题，但是不可否认的是SDN是未来的网络的发展方向。通过技术的不断演进，各公司、组织的协调制定统一的标准，可以预见未来以SDN为技术的ADWAN类的项目相会在广域网中大放异彩。

# 致谢

随着学位论文的终稿，我的校园生活也即将踏入尾声，回首往昔，在华科攻读硕士研究生的这段时光里，往事还历历在目。在这里我得到了许多人的帮助，非常高兴在这里成长。

首先需要感觉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进步。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。在论文终稿之际，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。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11月20日

# 参考文献