**第一章 绪论**

**1.1研究背景**

随着互联网的高速发展，运营商承载网络从最初简单Internet服务的“尽力而为”网络，逐步发展发展成能够提供涵盖文本、语音、视频等媒体业务的融合网络，其应用领域也逐步的向社会生活的各个方面渗透，深刻改变着人们的生产和生活方式。近年来云计算、大数据等新技术的兴起，面对云时代、大数据时代的高效、灵活的业务承载需求，传统网络的网络架构日益臃肿，面临一系列挑战：

1. 管理运维复杂：传统网络采用的是分布式控制平面，控制协议数量多，标准数量数以千计，如此庞大的控制协议体系，使得管理和维护网络变得愈发困难。

同时，设备厂商在实现这些标准协议时，都进行了一些特定的私有扩展，使得设备的操作维护变得更加复杂，进一步加剧了网络管理员操作维护网络的难度，同时大幅增加了网络的运维成本。

2. 网络封闭，创新困难：由于传统网络采用“垂直集成”的模式，控制平面和数据平面深度耦合，缺乏标准、开放的接口，且在分布式网络控制机制下，当需要在网络中部署新业务时，首先需要解决需求标准的定义，使得新技术的部署周期较长，严重制约网络的演进发展。

3. 设备日益臃肿：由于传统网络的技术体系采用“打补丁”式的演进策略，随着设备支持的功能和业务越来越多（例如，目前IETF发布的RFC标准超过7000个，且还在不停的增加新的RFC和Draft标准），其实现的复杂度显著增加。

传统网络的诸多限制导致网络架构需要改变。在这一背景下，业界一直在研究开发更加开放的新型的网络架构，促进网络逐渐向智能、开放、优化整合等方向转变。这种转变推动SDN软件定义网络的兴起。软件定义网络作为一种新型的网络架构，逻辑上集中的控制层面能够支持网络资源的灵活调度，灵活开放的接口能够支持网络能力的按需调用，将部分或全部网络功能软件化，更好地开放给用户，让用户更好地使用和部署网络，以适应快速变化的云计算、大数据以及更多的创新业务。

SDN作为一种对运营商网络具有重大影响的新技术，其价值已经被业界普遍认可，并在近些年发展迅速。但是随着对SDN架构开发和部署的不断深入, 安全性问题成为制约其发展的一个重要因素。一方面，SDN的出现给传统的网络安全研究带来了很多新的思路和解决方式；另一方面，作为一种全新的网络设计理念，其具有的集中控制性和开放性也会产生很多新的安全问题，比如控制器、基础设施层、控制器与应用层之间以及控制器和转发设备之间的安全问题等。

**1.2国内外研究现状**

**1.3论文研究内容**

**1.4论文结构**

**第二章 相关技术综述**

**2.1 SDN技术**

**2.2 SDN开源控制器调研分析与选型**

**2.3 访问控制技术**

**2.4 web开发技术**

**第三章 SDN应用访问控制系统的设计**

**3.1需求分析**

**3.2 总体设计**

**3.2.1 系统整体架构设计**

**3.2.2 模块设计**

**3.2.2 系统整体设计流程图**

**3.3 算法描述**

**第四章 SDN应用访问控制系统的实现**

**4.1 开发框架介绍**

**4.2 总体实现**

**4.3 主要功能模块实现**

**第五章 系统功能测试和性能分析**

**5.1功能测试**

**5.2 性能分析**

**第六章 总结和展望**

**6.1全文总结**

**6.2 展望**

**第三章 SDN应用访问控制系统的设计**

**3.1需求分析**

SDN网络应用程序和控制器之间的交互在整个北界接口(NBI)中进行。目前，各类控制器在北向接口(NBI)方面没有统一的规范，具有的安全防范措施较少。通过北向接口(NBI)暴露出来的漏洞，攻击者可以向控制器发起攻击，从而使得对来自应用层的安全威胁的防护工作变得尤为重要。应用程序通过北向接口与控制器进行交互主要包括两个方面：读取网络状态（Reading Network State）和写入网络策略（Writing Network Policies）。所面临的基本的安全问题是恶意应用程序随意访问网络状态信息和操纵网络流量，破坏网络的正常状态，威胁网络安全。针对这种情况，通过对控制器上应用程序的访问权限分析，设计一种细粒度的SDN应用访问控制系统。本系统需要对SDN应用进行应用身份信息的注册和权限的授予、以及创建较为灵活的基于属性的访问控制策略，当应用访问控制器上的资源时，先对应用进行身份认证，防止未经注册的非法应用对控制器的访问尝试；然后对经过身份认证的应用进行权限检查，确保应用只能访问具有访问权限的那些控制器资源，对权限范围外的控制器资源的试图访问予以拒绝，防止越权访问，最后对通过权限检查的应用依据之前设定好的访问控制策略进行访问控制判决，应用通过判决后才能访问控制器资源，读取相关的网络状态或写入网络策略。同时，网络管理人员可以通过系统中的UI界面查看控制器上接入的所有应用的状态信息，进行相应的操作。

本SDN应用访问控制系统的重点在于对SDN应用的身份认证、权限核查、以及进行基于属性的访问控制。因此，本系统实现的功能需求如下：

1 网络管理员能够对应用程序进行注册，在注册时除了进行身份信息的记录外，还可以对应用程序的访问权限在初始化时按应用的功能需求进行分配。同时，根据不同情况，创建合适的访问控制策略，为应用的访问控制提供策略依据。

2 当应用程序试图访问控制器资源时，对其进行身份认证，若身份信息验证通过则可以继续访问，否则视为非法应用，对访问请求予以阻止，并写入系统异常行为监测日志中。

3 当应用程序通过身份认证后，对其试图使用的权限进行检查，若满足权限集合的要求则请求可顺利通过，否则将请求信息视为无效请求，写入系统异常行为监测日志中。在该应用程序的生命周期内，访问控制系统严格按照应用程序访问权限集合对应用程序的每个访问请求进行检验，通过这种细粒度的访问权限检查方法，实现了对应用程序访问行为的监测，防止越权访问。

4 当应用程序通过权限检查后，视为合法应用。依据设定好的基于属性的访问控制策略，对应用进行访问控制判决，只有通过判决的应用才能在最终成功地访问控制器上的网络资源。同时将未通过判决的应用的有关信息写入系统异常行为监测日志中。

5网络管理员可以对应用程序的身份信息和其所具有的访问权限进行查询、修改和删除。同时，可以重置和修改访问控制策略。

6网络管理员可以通过系统查看到全局的网络状态信息，包括网络拓扑、网

节点和网络设备等。还能够查看接入控制器的所有应用的信息。

7 将所有非法的应用访问请求记录到日志文件中以进行审计。

**3.2 总体设计**

**3.2.1 系统整体架构设计**

依据上一小节的需求分析，提出了系统的整体框架图，如图所示

前端页面层

**XACML**

**访问控制**

**应用身份**

**信息管理**

**身份认证**

**应用权限管理**

**权限检查**

**数据存储层**

**日志记录**

**前端页面**

由图可以看出，本文设计的访问控制系统分为三层，分别是前端视图层、逻辑控制层和数据存储层。

1.前端视图层：网络管理员可以通过登录页面进入系统，能够查看整体的网络状态，包括控制器信息、网络拓扑信息、交换机和主机信息等。同时通过身份信息管理页面、权限管理页面、策略创建页面等三个页面与逻辑层进行数据交互，给逻辑层传递相关的数据信息。并显示应用身份、权限、策略等信息。

2.逻辑控制层：逻辑层作为本系统的核心部分，与前端视图层和数据存储层进行交互，完成对应用访问控制的大部分功能。逻辑控制层完成对访问控制器资源的应用进行身份认证、权限检查、XACML访问控制。SDN应用只有全部通过这三个流程的审查后，才能顺利的建立网络策略，获得需要的网络信息。同时逻辑层还负责管理和维护应用的身份信息、权限信息以及访问控制的策略信息。能够把非法的访问请求记录到特定的日志文件中，为以后恶意应用的检测提供依据。

3 数据存储层：负责存储应用的身份信息、权限信息等。由于逻辑层对应用进行身份认证、权限检查时会频繁的查询数据存储层中的数据，为了提高系统认证和检查的效率，数据存储层实际的数据储存是在内存中进行的。

**3.2.2 模块设计**

**3.3.2.1前端视图层**

前端视图层负责系统UI界面的显示工作，同时方便网络管理员进行相应的操作。实质上是一个HTTP客户端，与逻辑控制层的信息交互模块使用HTTP协议进行数据交互。按功能需求可划分为五个模块,分别是管理员登录页面、应用身份信息管理页面、权限管理页面、访问控制策略创建页面、网络视图页面。下面对各模块进行详细的介绍。

1. 管理员登录页面：网络管理人员使用用户名和管理员密码登录SDN应用访问控制系统，进行网络的监管。
2. 应用身份信息管理页面：此模块完成应用的注册、应用身份信息列表的显示、应用身份信息的修改和应用注销等功能。
3. 权限管理页面：此模块具有应用权限的授予、应用权限显示、应用权限增加与移除等四个功能。
4. 访问控制策略创建页面：网络管理员可以通过此页面录入有关属性的信息，为逻辑控制层中XACML访问控制模块创建基于属性的访问控制策略policy提供信息来源。
5. 网络视图页面：网络管理员可以在此模块查看到控制器的运行状态信息和全局的网络视图，包括网络拓扑图、交换机和主机设备等网络节点的详细信息。

**3.3.2.2 逻辑控制层**

逻辑控制层实现了本系统的核心功能，包括对应用的身份认证，权限检查，基于属性的访问控制等。如图所示，共划分为七个模块，其中身份认证模块、权限检查和XACML访问控制模块的本层的核心模块，而应用身份信息管理和权限管理模块则是上述三个重要模块的基础，负责维护所需的信息。日志记录模块负责对非法应用接入、越权访问等事件的监测记录。信息交互模块通过HTTP协议，解析前端视图层发送过来的HTTP请求，并返回相应的数据。以下是本层七个模块的详细介绍：

1. 应用身份信息管理模块

应用的身份信息（身份证书）的结构如下图所示:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| APPID | APPNAME | APPKEY | REGISTRY | REGDATE | EXPDATE | ATL |

APPID:SDN应用的ID,作为其标识，每个SDN应用的ID是唯一的。

APPNAME：SDN应用的名称。

APPKEY：SDN应用的密钥。

REGISTRY：SDN应用所属的注册商。

REGDATE：SDN应用的注册时间。

EXPDATE：SDN应用的有效期。

ATL：SDN应用的信用评级，为以后恶意应用的检测提供一个参考指标。

应用身份信息管理模块实现了对每个SDN应用身份信息的增加、删除、更新、查找等四个功能。

1. 权限管理模块

权限定义：Wen 等人在论文《Towards a Secure Controller Platform for Open Flow Applications》中分析的控制器上应用程序的15种访问权限，如图所示，主要分为read、write、notification三类，并将每种权限和控制器上特定的资源做了一一映射，应用程序需要访问该资源，必须具有对应的权限。

table1：权限目录分类表（映射到控制器Floodlight中相对应的API）

|  |  |  |
| --- | --- | --- |
| Category | Permission | Screening method(s) |
| Read | read\_topology | getAllSwitchMap:Controller.java |
| getLinks:LinkDiscoverManager.java |
| read\_all\_flow | getFlows:StaticFlowEntryPusher.java |
| read\_statistics | getSwitchStatistics:SwitchResourceBase.java |
| getCounterValue:SimpleCounter.java |
| read\_pkt\_in\_payload | get:FloodlightContextStore.java |
| read\_controller\_info | retrieve:ControllerMemoryResource.java |
| Notification | pkt\_in\_event | addToMessageListeners: Controller.java  addListener:ListenerDispatcher.java |
| flow\_removed\_event |
| error\_event |
| Write | flow\_mod\_route | insertRow:AbstractStorageSource.java |
| flow\_mod\_drop | deleteRow: AbstractStorageSource.java |
| set\_flow\_priority | insertRow: AbstractStorageSource.java |
| set\_devices\_config | setAttribute:OFSwitchBase.java |
| set\_pkt\_out | write:IOFSwitch.java |
| writeThrottled:IOFSwitch.java |
| flow\_mod\_modify\_hdr | parseActionString:StaticFlowEntries.java |
| modify\_all\_flows | setCommand:OFFlowMod.java |

每个SDN应用都具有一个权限列表PERLIST，记录应用所具有的权限集合，描述了应用可以访问控制器资源的最大范围。

权限管理模块负责对应用的权限列表PERLIST的初始化、查询、权限增加、权限移除等工作。

1. 信息交互模块

采用REST思想，把信息看成一种资源，通过URL定义资源的访问方式，通过GET/POST/PUT/DELETE方法实现对资源的CRUD操作,接受来自前端视图层的HTTP请求，响应，最后返回RESTFUL JSON格式的数据。

1. 身份认证模块

身份认证模块的流程图如下图所示：

**是否通过认证**

**写入日志**

**应用程序请求访问控制器**

**获得应用的APPID和APPKEY**

Y

N

**权限检查**

应用通过北向接口接入floodlight控制器，开始对控制器的资源的发起访问时：

身份认证模块首先获取应用请求中携带的ID和应用密钥KEY，然后利用APPID在数据存储层（内存数据库）中查找相关的应用信息，对ID和KEY进行匹配核对，完成对身份信息的识别，生成认证结果。若识别为未经注册的非法应用，则认证结果为拒绝，同时交给日志记录模块处理；若识别为已注册应用，则认证结果为通过，再交给权限检查模块进行权限的核查。

1. 权限检查模块

权限检查模块的流程图如下图所示：

**得到应用试图使用的权限PER0**

**在内存数据库中查询与该APPID对应的权限列表PER**

**PERLIST是否含有PER0**

Y

N

**XACML访问控制**

**写入日志**

应用成功通过身份认证后，首先根据应用的访问请求，得到应用试图使用的权限PER0，然后在数据存储层（内存数据库）中通过根据应用的APPID查找到对应的权限列表PERLIST，最后判断PER0是否在PERLIST中，即是否应用具有访问该资源的权限。如果具有，则通过审核，再交给XACML访问控制模块进行基于属性的访问请求的判决。如果不具有，则被视为越权访问，应用无法访问该资源，并交给日志记录模块处理。

1. XACML访问控制模块

XACML访问控制模块是SDN访问控制系统的核心模块，实现了对应用的基于属性的访问控制。网络管理员首先设置好创建基于属性的访问控制策略所需要的相关信息，应用通过身份认证和权限检查后，根据应用和访问环境的属性，查找相应的访问控制策略，进行访问控制判决。若判决通过，则应用可以成功访问，否则拒绝访问。

XACML访问控制模块的工作流程如图所示：

**策略创建子模块**

**访问控制判决子模块**

**判决结果**

**为通过？**

Y

N

**应用成功访问控制资源**

**写入日志**

**从信息交互模块获得相关信息**

由图可以看出，XACML访问控制模块主要分为：策略创建子模块和访问控制判决子模块。策略创建子模块完成基于属性的访问控制策略的创建功能，为访问控制判决子模块提供判决依据。访问控制判决子模块执行基于属性的访问控制算法，产生访问控制的判决结果，这种算法将在下一小节进行详细描述。

1. 日志记录模块

应用的访问请求在身份认证模块、权限检查模块、XACML访问控制模块中的判决结果为拒绝（即不允许访问）的时候，调用日志记录模块。将这些非法的访问请求记录到日志文件中，为以后恶意应用的检测提供依据。

日志记录的信息采用以下形式：

<date><time><app-id>< exception-event><denied-permission>

date和time：表示非法访问请求发生的时间。

app-id：表示发起访问的SDN应用ID。

exception-event：描述非法访问请求的详细信息，分为三类。分别是应用未通过身份认证/应用没有该权限/应用未通过XACML访问控制。

denied-permission：表示应用缺少的权限类型。

**3.3.2.3 数据存储层**

数据存储层通过自定义的NOSQL内存数据库模块完成信息的存储功能。把数据存储在内存中，大大加快了有关数据的查询速率。对于数据查询频繁的身份认证和权限检查模块而言，有效的提高了检查效率，有助于提高系统的性能。

NOSQL内存数据库是键值([Key-Value](http://baike.baidu.com/item/Key-Value))存储[数据库](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)，数据表设计分为两种，分别是应用信息表APPTABLE和应用权限表PERTABLE。应用信息表APPTABLE使用键APPID来对应一个应用的身份信息类AppIdentityCertificate。应用权限表PERTABLE使用键APPID来对应一个应用的权限列表PERLIST。内存数据库还实现了对数据表的增删改查功能。

**3.3 算法描述**

本文提出了一种基于属性的应用访问控制算法，该算法能够根据基于属性的访问策略进行对应用进行访问控制判决。当应用程序访问控制器资源时，应用的身份属性和访问时外部环境的当前状态信息都可以成为决定访问请求能否被通过的属性。例如应用的注册商、注册日期、信用评级、访问的当前时间等，都可以视为访问控制决策中的重要属性。

参考文献[11]给出的一个基于ABAC的API访问控制模型，并对其进行改进，得到了一种基于属性的应用访问控制算法，如图3-1所示：算法的输入为REQ和POL和ATT，其中REQ表示应用的访问请求，请求中包括应用的身份信息APP和当时的访问环境信息ENV；POL表示策略的集合，集合中的每个poli都为设定的一个访问控制策略；ATT是该次访问的属性集，属性集中的各属性atti可能来自于REQ，也有可能来自外部环境。算法的输出Decision是布尔型，表示对于请求的判决结果。

|  |
| --- |
| 算法　基于属性的应用访问控制算法 |
| 输入：  REQ<APP, ENV>  POL = {pol1,pol2,…polm}  ATT = {att1,att2,…attm} |
| 输出:  Decision : Boolean |
| BEGIN  For all(poli ∈POL) do  IF ATT\not\rightarrowpoli THEN  Decision ← Deny  END IF  END For  Decision ← Permit  END. |

基于属性的应用访问控制算法首先通过策略执行点PEP把应用的访问请求转化为Request，再将Request交给策略决策点PDP进行评估, 最后策略决策点PDP查找相应的访问控制策略Policy和有关的属性信息，产生评估结果Response。若评估结果Response为Permit，则允许访问，若是Deny则拒绝。

**第四章 SDN应用访问控制系统的实现**

**4.1 开发框架介绍**

**4.2 总体实现**

本系统是在floodlight的基础上使用java语言进行模块扩展来实现的。按照先前的系统设计实现系统的各个模块，将模块的信息添加到floodlight的配置文件中，当floodlight启动运行时按照配置文件加载各个模块。采用模块管理器FloodlightModuleLoader实现对模块的自动加载、启动和初始化功能，首先在配置文件floodlightdefault.properties中声明各个模块的路径，然后FloodlightModuleLoader读取配置文件，根据路径查找到相应模块，查看该模块启动所需要依赖的其它模块，并将其加入到模块加载列表中，最后FloodlightModuleLoader根据模块加载列表依次初始化和启动各个模块。模块的加载过程如图所示：

入口**main()**函数

启动**FloodlightModuleLoader**

**FloodlightModuleLoader**查看配置文件，寻找需要加载的模块

关联模块与对应的服务

生成模块加载列表

**FloodlightModuleLoader**对模块进行初始化并启动

前端视图层

**SystemClientModule**

逻辑控制层

**AppInfoManageModule**

**AppPermissionManageModule**

**AppAuthenticationModule**

**AppPermissionCheckModule**

**XacmlCtrModule**

**LogManagementModule**

**InformationInteractionModule**

数据存储层

**DataInfoStorgeModule**

系统的功能模块列表

系统的功能模块列表分为AppInfoManageModule,AppPermissionManageModule, AppAuthenticationModule,AppPermissionCheckModule,XacmlCtrModule,LogManagementModule,InformationInteractionModule,SystemClientModule和DataInfoStorgeModule。逻辑控制层的功能由前七个模块实现，前端视图层的功能由SystemClientModule实现，DataInfoStorgeModule实现数据存储层的功能。

**4.3 主要功能模块实现**

**4.3.1逻辑控制层各模块实现**

逻辑控制层实现了AppInfoManageModule,AppPermissionManageModule, AppAuthenticationModule,AppPermissionCheckModule,XacmlCtrModule,LogManagementModule,InformationInteractionModu七个模块。其中，AppInfoManageModule实现应用的身份信息管理功能，AppPermissionManageModule实现应用的权限管理功能，AppAuthenticationModule实现应用的身份认证功能，AppPermissionCheckModule实现应用的权限检查功能，XacmlCtrModule实现基于属性的访问控制功能。

4.3.1.1 AppInfoManageModule实现

AppInfoManageModule模块主要实现对应用身份信息的管理，包括应用身份的增加，删除，修改，查询等。主要方法有applicationRegistration()、applicationUpdate()、applicationCancellation()、applicationFinder()和applicationFinderAll()。

SDN应用的身份信息的数据结构表示如下：

AppIdentityCertificate{

String appId; //应用标识

String appName;//名称

String appKey;//应用密钥

String registry;//注册商

String registrationDate;//注册时间

String expDate;//有效期

int ATL//信用级别

}

applicationRegistration()方法实现在应用列表APPLIST中注册一个新的应用，applicationUpdate()方法实现对应用列表中应用身份信息的更新，applicationCancellation()方法实现应用的删除注销功能，application-Finder()和applicationFinderAll()实现对应用身份信息的查询功能。

4.3.1.2 AppPermissionManageModule实现

AppPermissionManageModule模块实现对应用权限的初始化，权限增加，权限移除，权限查询等功能。主要方法有:

appPermissionSetInitialization()

appPermissionSetAdd()

appPermission-SetDelete()

appPermission-SetFinderALL()

应用的权限类型定义的数据结构表示如下：

PermissionType{read\_topology,read\_all\_flow,read\_statistics,read\_pkt\_in\_payload,read\_controller\_info,pkt\_in\_event,flow\_removed\_event,error\_event,flow\_mod\_route,flow\_mod\_drop,set\_flow\_priority,set\_devices\_config,set\_pkt\_out,flow\_mod\_modify\_hdr,modify\_all\_flows}

一共有15种权限类型，每个应用的权限列表PERLIST使用ArrayList储存，并通过hashmap来实现应用ID和权限列表PERLIST的一一映射。

AppPermissionSetInitialization()方法实现对应用权限列表的初始化功能，appPermissionSetAdd()方法实现对应用权限列表的增加功能，appPermissionSetDelete()实现对应用权限的移除功能，appPermissionSetFinderALL()实现应用的权限查询显示功能。

4.3.1.3 AppAuthenticationModule和AppPermissionCheckModule实现

AppAuthenticationModule负责应用的身份认证功能，主要方法是appAuthentication(String appId,String appKey)。appAuthentication()通过查询数据存储层中应用的身份信息，把身份信息中的ID和KEY与传入的appId和appKey进行核对，若appId和appKey完全正确，则身份认证成功，返回认证结果。

AppPermissionCheckModule实现对应用的权限检查功能。主要方法有appPermissionSetFinder(String appId,PermissionType permission)。appPermissionSetFinder()首先通过appId在数据存储层查找得到应用的权限列表PERLIST，然后再判断PERLIST是否含有permission，即应用是否具有permission权限，若PERLIST含有permission，则返回通过的检查结果，反之则拒绝。

4.3.1.4 XacmlCtrModule实现

XacmlCtrModule主要负责对应用的访问控制功能，实现基于属性的访问控制算法。采用开源框架XACML，主要方法包括creatPolicy()，creatRequest()，creatPDP()和getResult()。模块的实现流程如图所示：

**写入日志**

**从信息交互模块**

**获得属性的相关信息**

**创建访问控制策略Policy**

Denyy

**策略执行点PEP将应用的访问求转化为Request**

**将Policy交给策略管理点PAP**

**PEP通过上下处理器将Request转交给策略决策点PDP**

**PDP通过PAP查找相关的Policy**

**PDP生成评估结果**

**应用成功访问控制器资源**

Permit

流程主要分为三个步骤，首先creatPolicy()创建访问控制策略Policy并交给策略管理点PAP，然后creatRequest()把应用的访问请求转化为Request, creatPDP()创建策略决策点PDP，PDP读入Request并在PAP中查找相应的策略，最后通过getResult()得到最后的判决结果。

4.3.1.5 LogManagementModule实现

LogManagementModule采用java日志框架logback实现，把异常信息存储在日志文件中。主要方法有getLogger()和logger.info()。生成的日志文件的相关信息在配置文件logback.xml中定义。应用的访问请求在身份认证模块、权限检查模块、XACML访问控制模块中的判决结果为拒绝（即不允许访问）时，LogManagementModule首先通过日志工厂loggerFactory的getLogger()方法生成日志管理器logger,然后日志管理器logger使用logger.info()方法把有关信息写入日志文件中，最后关于异常的访问行为的详细信息在日志文件被记录。

4.3.1.6 InformationInteractionModule实现

InformationInteractionModule模块负责作为前端视图层的后台模块，实现与客户端的数据交互功能。InformationInteractionModule模块的实现框架如图所示：

采用Restlet框架，继承了Application类，通过AppListResource、PermissionListResource、PolicyCreatResource把将应用身份信息，权限信息，访问控制策略信息封装成ServerResource形式的Rest资源，定义相应的URL访问路径，并实现了资源的get/put/post/delete方法。Restlet中的Application监听本地的8080端口，SystemClientModule直接通过HTTP请求，实现对InformationInteractionModule模块中资源的CRUD操作。

**4.3.2前端视图层各模块实现**

前端视图层主要实现SystemClientModule模块。SystemClientModule采用html/css和js技术编写，使用前端开源工具包bootstrap对界面进行美化，利用javaScript框架jquery对页面的动作进行响应。把数据用ajax进行封装，向逻辑控制层的InformationInteractionModule模块发送http请求，并接收InformationInteractionModule模块返回的json数据，从而实现数据的交互。

SystemClientModule模块包含五个子模块，分别是分别是管理员登录页面loginPage、应用身份信息管理页面appInfoManagePage、权限管理页面permissionManagePage、访问控制策略创建页面policyCreatePage、网络视图页面networkPage。下面对各个子模块的实现进行详细介绍。

4.3.2.1 管理员登录页面loginPage

网络管理员在文本框中输入用户名和密码，点击登录按钮后，使用jquery获取输入的用户名和密码，并对其进行验证。若用户名和密码正确，则登录成功，跳转至系统应用身份信息管理页面appInfoManagePage。实现后的loginPage如图所示：

4.3.2.2 应用身份信息管理页面appInfoManagePage

AppInfoManagePage实现应用身份信息的查询，注册，修改和注销功能，含有四个方法：showAppListInfo() $("#registityButton").click(function(),$("#updateButton").click(function())和$("#deleteButton").click(function())。

其中，showAppListInfo()使用ajax向InformationInteractionModule以get方式发送http请求，得到json格式封装的关于应用身份信息的数据，最后解析并展示在页面上。$("#registityButton").click(function()通过表单获得网络管理员输入的注册数据，封装成json格式后，在ajax中以put方式向后台模块发送http请求，实现应用的注册功能。同理，("#updateButton").click(function())通过表单获得网络管理员要修改的数据，在ajax中以post方式向后台模块发送http请求，实现应用身份信息的修改功能。$("#deleteButton").click(function())在ajax中以delete方式向后台模块发送http请求，实现应用注销的功能。图—是实现后的界面图：

4.3.2.2 权限管理页面permissionManagePage

和appInfoManagePage类似，permissionManagePage实现应用权限的初始化，增加，移除和查询功能。主要有四个方法：$("#submitButton").click(function())，$("#checkButton").click(function())， $("#addButton").click(function()和$("#removeButton").click(function())。

$("#submitButton").click(function())使用ajax发送put方式的http请求，实现应用权限的初始化功能。$("#checkButton").click(function())使用ajax发送get方式的http请求，实现应用权限的查询功能。$("#addButton").click(function()和$("#removeButton").click(function())分别使用post和delete方式发送http请求来实现应用权限的增加和移除功能。图—是实现后的界面图：

4.3.2.3 访问控制策略创建页面policyCreatePage

policyCreatePage把允许访问的时间段[time-quantum](https://www.baidu.com/link?url=G6Rst-BmrDbZAe-BzxhW1HjhRG8ZCd5GrwnSQRlbXUdEWjK89jmWdl_w5mB7ZgjMp1k2qH59mwVyH4OwTHE39n1hvCbVMJhzupPqdE8Dkn0g9qltxuFwiBkQ9ohqfL91&wd=&eqid=e13b35fb00017cfa000000035912be48)和允许的应用注册商registrary等数据转化为json格式后，以ajax的put方式向InformationInteractionModule发送http请求，为访问控制策略的创建提供必要的信息。实现后的界面图如图所示：

4.3.2.4 网络视图页面networkPage

networkPage通过对floodlight的RestApi调用，得到有关控制器，网络拓扑，交换机和网络节点的详细信息，解析并展示在页面上。实现后的界面图如图所示：

**4.3.3 数据存储层各模块实现**

数据存储层实现应用列表APPLIST和权限列表PERLIST的存储功能。主要模块DataInfoStorgeModule使用一个Hashmap<String tableName,Object table>存储和管理APPTABLE和PERTABLE这两张表。APPTABLE使用Hashmap<String appid, AppIdentityCertificate app>存储每个应用的身份信息，PERTABLE使用Hashmap<String appid, Arraylist permissionlist >每个应用的权限列表。在floodlight控制器启动时自动加载DataInfoStorgeModule模块，分配相应的存储空间。

**第五章 系统测试**

在任何系统的开发中，测试都是其中很重要的一部分。测试有助于发现系统的漏洞和不完善的地方，有助于改进系统功能、提高系统性能。本章对实现后的SDN应用访问控制系统进行功能以及性能上的测试，以期发现错误，保证系统的完整性和可靠性。本系统是在floodlight的基础上进行模块扩展实现的，测试时通过mininet模拟底层测试网络，用Python编写Rest应用circuitpusher.py作为测试应用来验证系统有关的访问控制功能。

系统测试环境如下：

操作系统：ubuntu Kylin 14.04 Lts

浏览器: Google Chrome 51.0.2704.106m

SDN控制器：floodlight-0.91V

Mininet版本：2.1.0

**5.1功能测试**