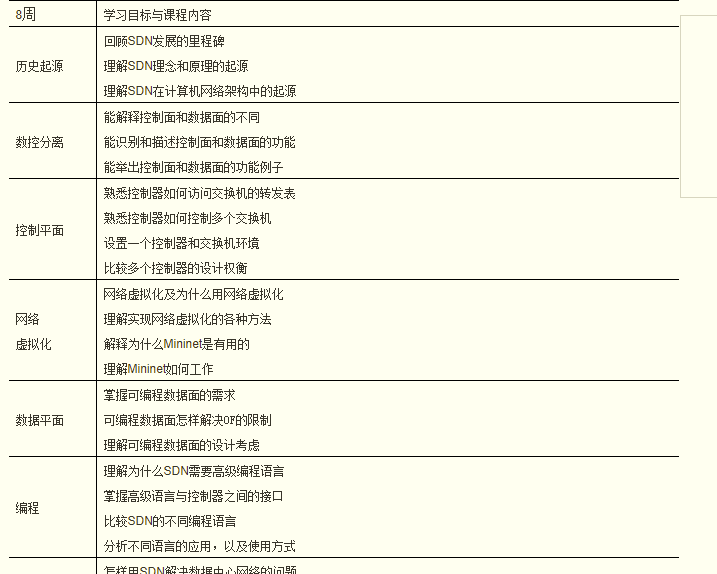
# Nick Feemster class

## Schedule





# Study plan

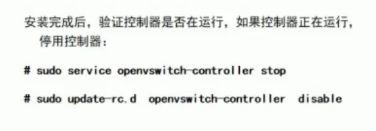
## nick feemster ’s class. Video experiment.

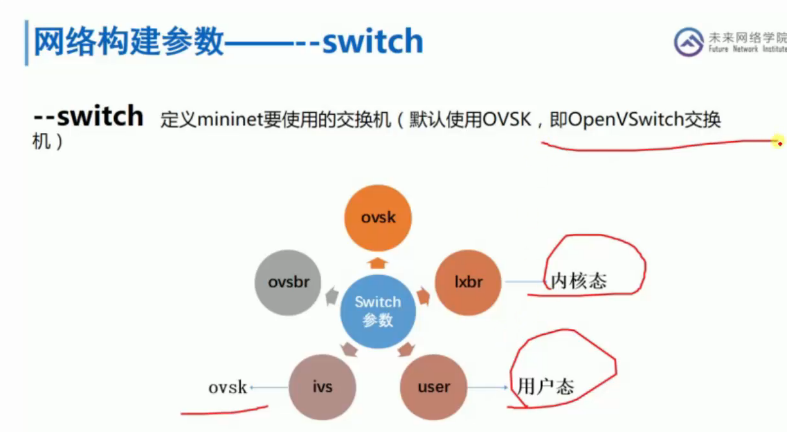
<http://noise.gatech.edu/classes/cs8803sdn/fall2014>

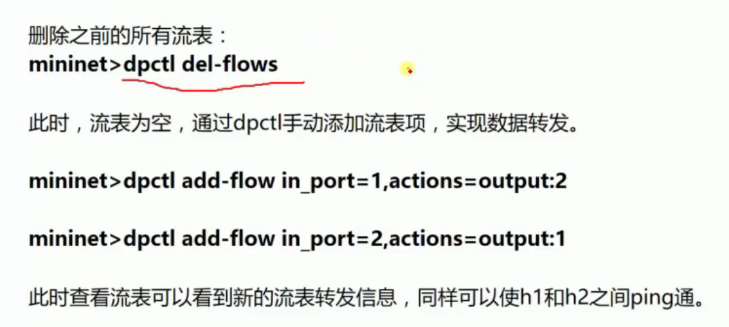
## http://edu.sdnlab.com/training/378.html

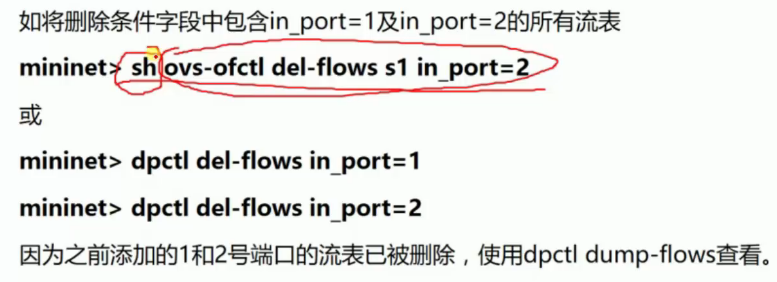
# Setup Mininet Environment

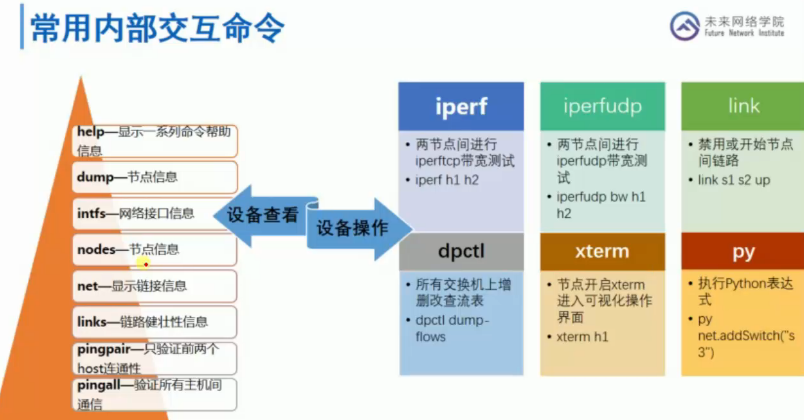
* **SSH into your VM:** Go back to your laptop. Open a terminal (Windows users try putty), and try to connect to your VM through ssh.
* ssh cs244@192.168.X.X (replace with your host-only IP address)











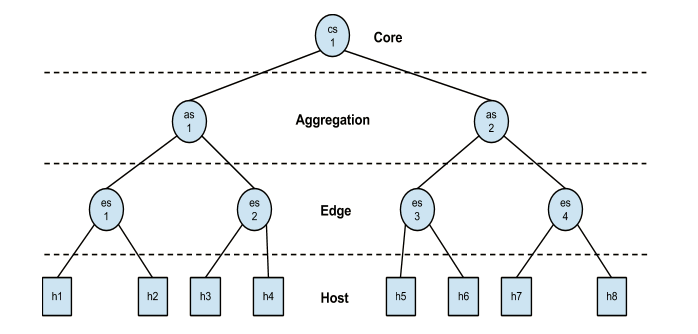
Py help(h1)

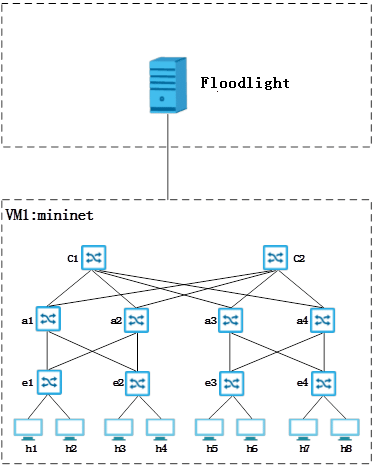
Py dir(h1)

# Mininet Topologies and API Assignment

Top of rack 架顶式

## Assignment





基于胖树（Fat-Tree）拓扑创建网络，胖树网络的好处是具有一个多层次的树状拓扑结构固有的容错能力。

相同边缘交换机下的主机间、相同聚合交换机下不同边缘交换机间、相同核心交换机不同聚合交换机下的主机.

## Mininet command

Note, if during the execution submit.py script crashes for some reason or you terminate it using CTRL+C, make sure to clean mininet environment using:

$ sudo mn -c

Also, if it still complains about the controller running. Execute the following command to kill it:

$ sudo fuser -k 6633/tcp

net.addController( 'c0', controller=RemoteController, ip='10.0.0.20', port=6633 )

Ensure that no controllers are running by executing $ sudo killall controller, $ sudo fuser -k 6633/tcp and $ sudo mn -c to clean up mininet

脚本里net = Mininet(topo=topo,host=CPULimitedHost, link=TCLink)中的link=TCLink是设置bw、delay等的前提。

linkopts = dict(bw=10, delay='5ms', loss=1, max\_queue\_size=1000, use\_htb=True)

’’’

alternately: linkopts = {'bw':10, 'delay':'5ms', 'loss':1, 'max\_queue\_size':1000, 'use\_htb':True}

’’’

self.addLink(node1, node2, \*\*linkopts)

ImportError: No module named myModule

导入模块时，不带模块的后缀名，比如.py  
Python搜索模块的路径：  
1)、程序的主目录  
2)、PTYHONPATH目录（如果已经进行了设置）  
3)、标准连接库目录（一般在/usr/local/lib/python2.X/）  
4)、任何的.pth文件的内容（如果存在的话）.新功能，允许用户把有效果的目录添加到模块搜索路径中去  
.pth后缀的文本文件中一行一行的地列出目录。  
这四个组建组合起来就变成了sys.path了，

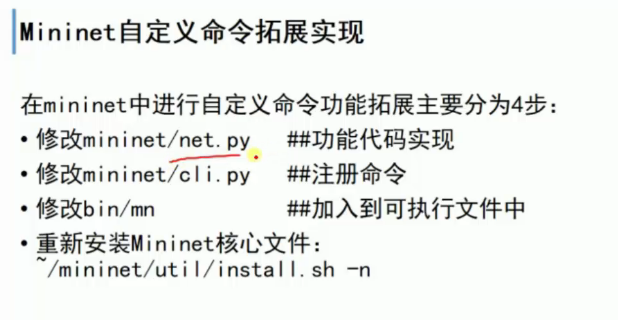
python -m SimpleHTTPServer 80 &amp

后面的80端口是可选的，不填会采用缺省端口8000。注意，这会将当前所在的文件夹设置为默认的Web目录

* --mac指定虚拟主机的mac地址顺序编号，若不带此参数则随机编号
* --controller指定of交换机的控制器
* --switch指定虚拟交换机的类型，ovsk表示虚拟交换机为ovs Kernel mode
* --custom指定自定义拓扑文件
* --topo指定加载拓扑的名字

执行ctrl+a+d命令切换到系统命令行。

通过screen mn命令启动mininet，这样可以通过ctrl+a+d和screen –r命令进行系统命令行和mininet命令行之前的窗口切换。



## Openvswitch

<http://www.cnblogs.com/popsuper1982/p/5848879.html>

<https://www.ibm.com/developerworks/cn/cloud/library/1401_zhaoyi_openswitch/>

### Open vSwitch（OvS）源代码分析之简介

<http://www.sdnlab.com/2788.html>

OVS常用命令与使用总结

http://blog.csdn.net/rocson001/article/details/73163041

* ovs-vswitchd ：主要模块，实现switch的daemon，包括一个支持流交换的Linux内核模块；
* ovsdb-server ：轻量级数据库服务器，提供ovs-vswitchd获取配置信息；
* ovs-brcompatd ：让ovs-vswitch替换Linux bridge，包括获取bridge ioctls的Linux内核模块；
* ovs-dpctl： 用来配置switch内核模块；
* 一些Scripts and specs： 辅助OvS安装在Citrix XenServer上，作为默认switch；
* ovs-vsctl ：查询和更新ovs-vswitchd的配置；
* ovs-appctl ：发送命令消息，运行相关daemon；
* ovsdbmonitor： GUI工具，可以远程获取OvS数据库和OpenFlow的流表。
* ovs-openflowd：一个简单的OpenFlow交换机；
* ovs-controller：一个简单的OpenFlow控制器；
* ovs-ofctl ：查询和控制OpenFlow交换机和控制器；
* ovs-pki ：OpenFlow交换机创建和管理公钥框架；
* ovs-tcpundump：tcpdump的补丁，解析OpenFlow的消息；

# NOS

SDN典型网络操作系统分析总结

http://www.sdnlab.com/17907.html

## ONOS

<http://www.sdnlab.com/4224.html>

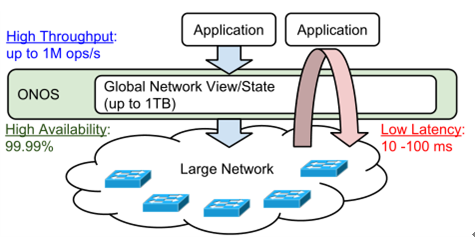
**http://www.sdnlab.com/16885.html**

ONOS：面向运营商网络的SDN操作系统

http://edu.sdnlab.com/openclass/671.html

<http://www.sdnlab.com/16912.html>

<http://developer.huawei.com/ict/cn/site-sdn-onos/article/onos-paradigm>



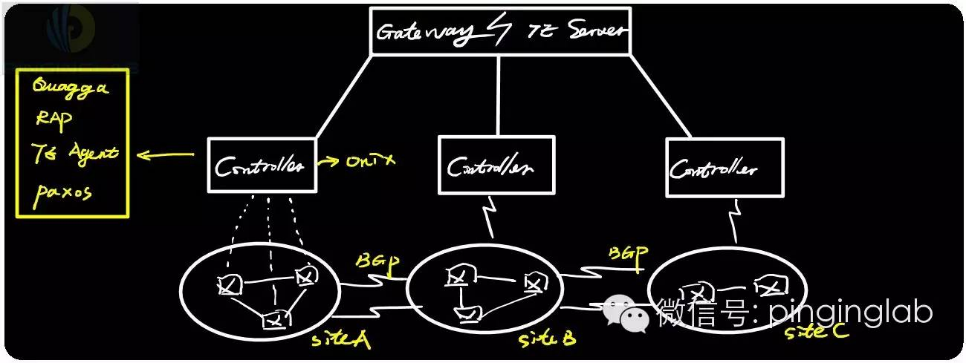
* 高吞吐量，达到1M requests/s；
* 低延迟，事件进程10-100ms；
* 全局网络状态大小，数据量最高达到1TB；
* 高可用性，99.99%的服务可用性。

## Ryu

<http://ryu.readthedocs.io/en/latest/>

<http://osrg.github.io/ryu/resources.html>

http://osrg.github.io/ryu-book/en/html/



## Opendaylight

### Install

#### Plugin install

feature:install odl-restconf odl-l2switch-switch-ui odl-openflowplugin-all odl-mdsal-apidocs odl-dlux-core odl-dlux-node odl-dlux-yangui

在没有按照顺序安装组件的情况下，无法登录进入ODL主界面。解决方法是通过logout退出 karaf平台，进入上级目录，删除data目录：rm –r data，进入bin目录：cd bin，执行./karaf clean，再次重复上面的安装组件操作。

wget -q -O -https://raw.githubusercontent.com/opendaylight/odlparent/master/settings.xml > settings.xml

mvn org.apache.maven.plugins:maven-archetype-plugin:3.0.0:generate \

-DarchetypeGroupId=org.opendaylight.controller -DarchetypeArtifactId=opendaylight-startup-archetype \

-DarchetypeRepository=https://nexus.opendaylight.org/content/repositories/public \

-DarchetypeCatalog=https://nexus.opendaylight.org/content/repositories/public/archetype-catalog.xml \

-DarchetypeVersion=1.2.2-Boron-SR2

### Resource

<https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL:Startup_Project_Archetype>

https://wiki.opendaylight.org/view/GettingStarted:\_Eclipse

<http://www.sdnlab.com/17863.html>

SDN开发笔记系列（已完结）

1、[SDN开发笔记（一）：SDN开发环境的搭建（win7环境）](http://www.sdnlab.com/17124.html)

2、[SDN开发笔记（二）：SDN控制器的使用](http://www.sdnlab.com/17356.html)

3、[SDN开发笔记（三）：ODL源码编译生成发行版控制器](http://www.sdnlab.com/17399.html)

4、[SDN开发笔记（四）：md-sal应用程序开发](http://www.sdnlab.com/17555.html)

5、[SDN开发笔记（五）：应用程序集成到ODL控制器](http://www.sdnlab.com/17830.html)

6、[SDN开发笔记（六）：yang模型理解](http://www.sdnlab.com/18066.html)

7、[SDN开发笔记（七）：L2switch源码分析（上）](http://www.sdnlab.com/18285.html)

8、[SDN开发笔记（八）：L2switch源码分析（下）](http://www.sdnlab.com/18293.html)

http://www.sdnlab.com/18316.html

通过浏览器访问http://localhost:8080（锂版本web：http://localhost:8181/index.html）页面。

### Maven

<http://www.deepsdn.com/archives/388>

### YANG

YANG模型与OpenDaylight南北向接口

http://www.sdnlab.com/14932.html

### ODL控制器设计原则

ODL在设计的时候遵循了六个基本的架构原则（以下来自opendaylight官方文档）：

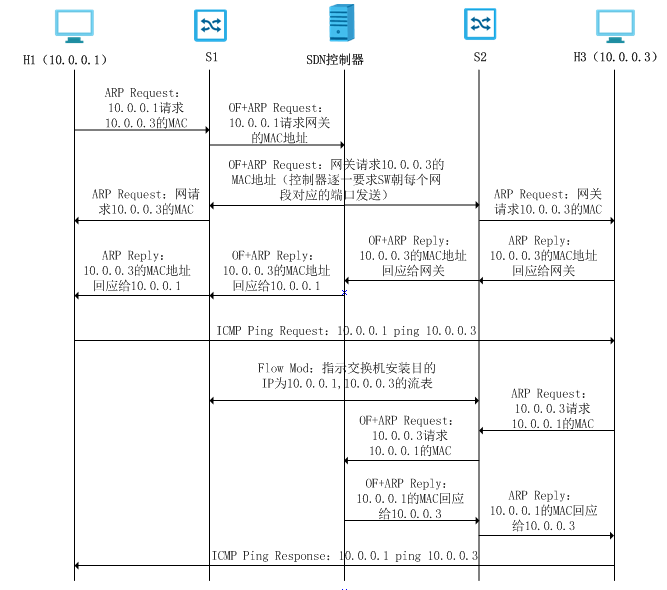
1、运行时模块化和扩展化（Runtime Modularity and Extensibility）：支持在控制器运行时进行服务的安装、删除和更新。

2、多协议的南向支持（Multiprotocol Southbound）：南向支持多种协议。

3、服务抽象层（Service Abstraction Layer）：南向多种协议对上提供统一的北向服务接口。Hydrogen中全线采用AD-SAL，Helium版本AD-SAL和MD-SAL共存，Lithium和Beryllium中已基本使用MD-SAL架构。

4、开放的可扩展北向API（Open Extensible Northbound API）：提供可扩展的应用API，通过REST或者函数调用方式。两者提供的功能要一致。

5、支持多租户、切片（Support for Multitenancy/Slicing）：允许网络在逻辑上（或物理上）划分成不同的切片或租户。控制器的部分功能和模块可以管理指定切片。控制器根据所管理的分片来呈现不同的控制观测面。



1.源主机h1（10.0.0.1）发出ARP请求h3（10.0.0.3）的MAC，交换机SW1并不知道如何转发该包，因此将其通过Packet In消息发送到Controller处理。

2.Controller收到ARP请求后，会要求每一台SW所对应10.0.0.0/8网段的非SW互联端口（只有这些端口是连接主机或传统网络的）发出ARP来请求10.0.0.2的MAC地址。这里Controller并不是简单的将收到ARP原封不动的发出，而是将源IP改为默认网关IP地址，然后发出。

3.只有h3（10.0.0.3）才会响应ARP，它将ARP Response发送到SW2。SW2也不知道如何处理，因此将ARP封装在Packet In中发送到Controller。

4.Controller发现这是ARP响应，并且正是之前10.0.0.1发送的ARP请求，因此它会将该ARP通过OF协议发到SW1，同时指示SW1将其送出的端口（也就是h1对应的端口）。SW1执行该操作。

5.Controller在收到h3的ARP后也得知了10.0.0.3的位置，它根据网络拓扑计算，可以得到全网到达10.0.0.3和10.0.0.1的转发路径，并将流表通过OF Flow Modify消息推送到每一台交换机上。

6.h1收到ARP Response后完成ARP解析过程，然后它构造ICMP PING Request数据包，其中源和目MAC分别为h1和h3的MAC，源和目IP分别为h1和h3的IP。由于SW1和SW2都已经成功的装载了到h3(10.0.0.3)的流表，因此该数据包将被顺利发送到h3。

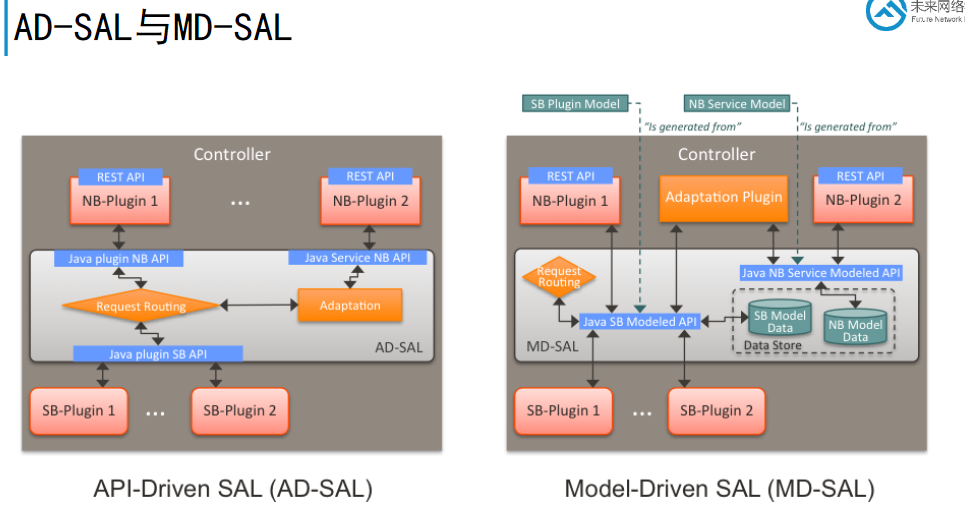
7.h3收到该ICMP PING Request，源是h1，但是此时它尚未有h1的MAC，于是还要进行一次ARP解析，SW2再次将ARP请求封装成Packet In消息发送Controller。

8.Controller已经得知h1的MAC，可直接响应，并通过OF向SW2返回ARP结果和所需要送出的端口（h2接入的端口）。

9.h3学到ARP后，即可构造ICMP Response包，发送到SW2，SW2根据h1目的地址匹配转发表将其转发到SW1，SW1根据h1目的地址匹配转发表将其发送到h1对应的端口。h1到h3的双向通道至此完全打通。

### OpenDaylight中MD-SAL

http://www.sdnlab.com/1418.html



### OpenDaylight VTN

OpenDaylight VTN 简单介绍及部署

<http://www.sdnlab.com/4093.html>

OpenDaylight VTN管理OpenStack网络的环境搭建与验证

http://www.sdnlab.com/7469.html

Opendaylight VTN虚拟网功能验证与开发环境搭建

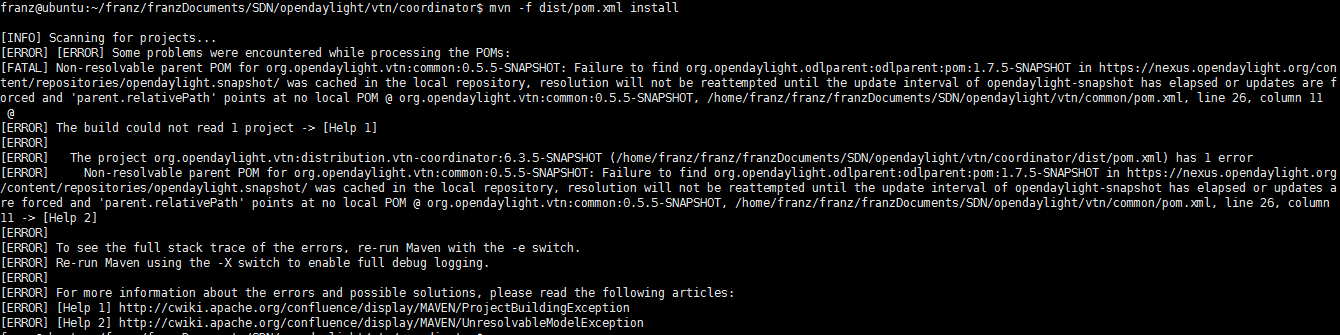
<http://www.sdnlab.com/10467.html>

http://www.sdnlab.com/4093.html/2/

OpenDaylight Virtual Tenant Network (VTN) Overview

<http://docs.opendaylight.org/en/stable-boron/developer-guide/virtual-tenant-network-(vtn).html>

<http://docs.opendaylight.org/en/stable-boron/user-guide/virtual-tenant-network-(vtn).html?highlight=VTN%20install#virtual-tenant-network-vtn-api>



https://www.youtube.com/user/opendaylightproject/videos

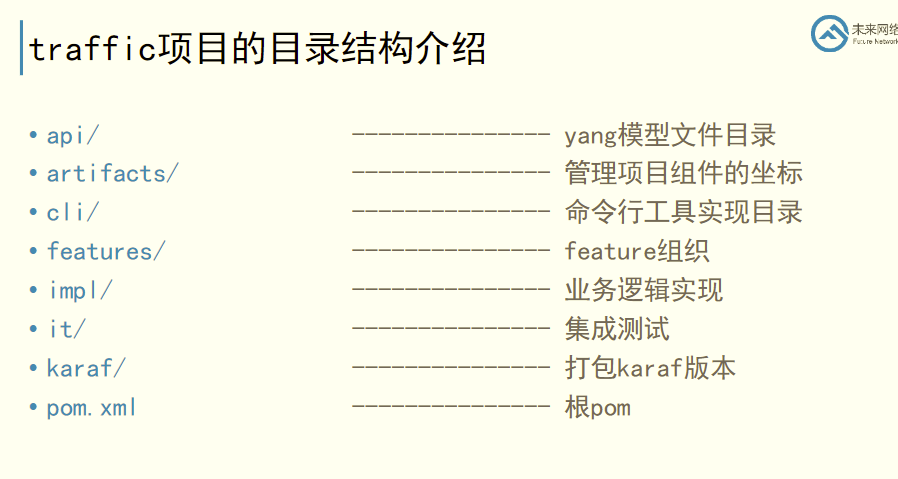
#### install

sudo apt-get install pkg-config gcc make ant g++ maven git libboost-dev libcurl4-openssl-dev libssl-dev unixodbc

sudo apt-get install cmake libgtest-dev

https://wiki.opendaylight.org/view/OpenDaylight\_Virtual\_Tenant\_Network\_(VTN):Installation:VTN\_Coordinator

content structure



# Pox Firewall Assignment

## Pox

<https://openflow.stanford.edu/display/ONL/POX+Wiki>

SDN常用控制器安装部署之POX篇

<http://www.sdnlab.com/2771.html>

POX的教程可以参考这个视频，感觉讲得很清楚【需FQ】：

https://www.youtube.com/watch?v=4KGnxPzxc6w&t=192s

https://www.udacity.com/wiki/cn/assignment7-sdn-firewall

### pox run

Modules are looked for everywhere that Python normally looks, plus the

"pox" and "ext" directories. Thus, you can do the following:

./pox.py forwarding.l2\_learning

You can pass options to the modules by specifying options after the module name. These are passed to the module's launch() funcion. For example,

to set the address or port of the controller, invoke as follows:

./pox.py openflow.of\_01 --address=10.1.1.1 --port=6634

pox.py also supports a few command line options of its own which should be given first:

--verbose print stack traces for initialization exceptions

--no-openflow don't start the openflow module automatically

### Pox desktop

./pox.py samples.pretty\_log web messenger messenger.log\_service messenger.ajax\_transport openflow.of\_service poxdesk openflow.discovery poxdesk.tinytopo poxdesk.terminal forwarding.l2\_learning浏览器访问http://pox-ip:8000/poxdesk，默认端口8000，POX的Web访问界面

Tcpdump

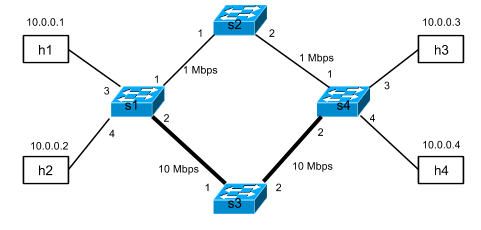
<http://www.cnblogs.com/ggjucheng/archive/2012/01/14/2322659.html>

## Assignment

In this assignment, your task is to implement a layer 2 firewall that runs alongside the MAC learning module on the POX OpenFlow controller. The firewall application is provided with a list of MAC address pairs i.e., access control list (ACLs). When a connection establishes between the controller and the switch, the application installs flow rule entries in the OpenFlow table to disable all communication between each MAC pair.

Your firewall should be agnostic of the underlying topology. It should take MAC pair list as input and install it on the switches in the network. To make things simple, we will implement a less intelligent approach and will install rules on **all** the switches in the network.

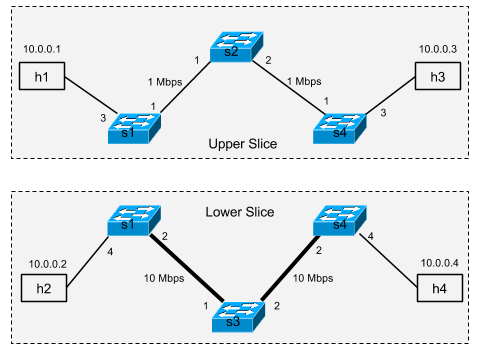
# Network Virtualization Assignment

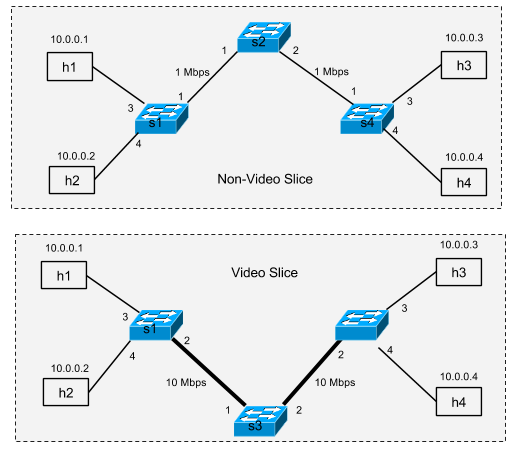


For simplicity, we’ll have each site represented by a single OpenFlow switch, s1 and s4, respectively. The sites, s1 and s4, have two paths between them:

* a low-bandwidth path via switch s2
* a high-bandwidth path via switch s3

Switch s1 has two hosts attached: h1 and h2, and s4 has two hosts attached: h3 and h4.



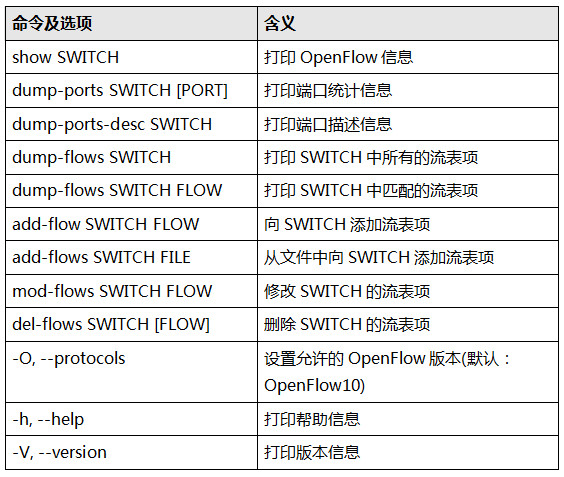


You can now test that your slices work properly be ensuring that video (in this case, port 80) traffic traverses the 10 Mbps link and non-port 80 traffic traverses the 1 Mbps link. For example, you can test the two paths between h2 and h3 as below. (Your code should work for *all* pairwise paths.)

## Answer

https://github.com/onstutorial/onstutorial/wiki/Flowvisor-Exercise

## Ovswitch



## Flowvisor

<http://www.sdnlab.com/3081.html>

<http://blog.csdn.net/sherkyoung/article/details/38405839>

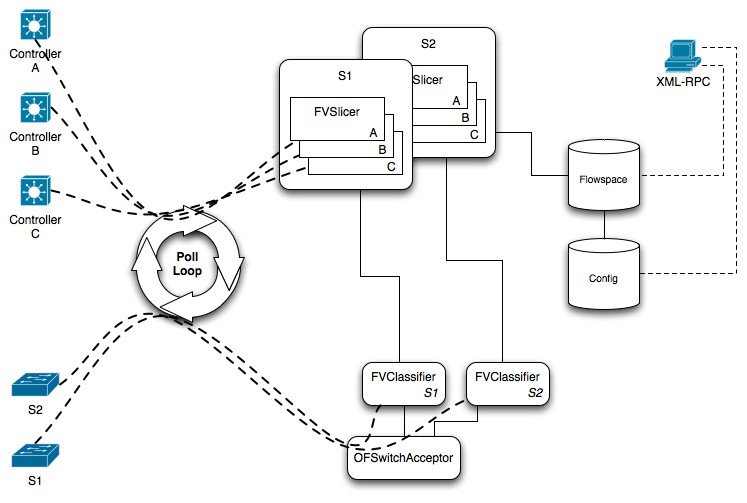
<http://groups.geni.net/geni/wiki/FlowVisor>

http://blog.csdn.net/sherkyoung/article/details/38405907

https://openflow.stanford.edu/display/ONL/Flowvisor

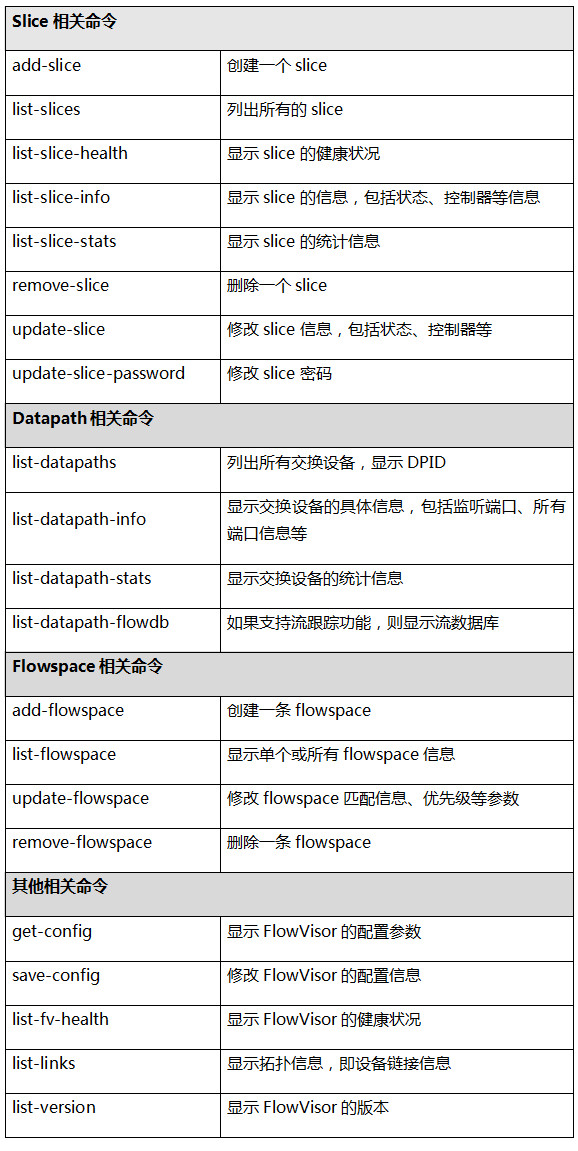
FlowVisor的设计有如下几个原则：(a) FlowVisor对控制器和交换机而言是透明的，它们都感知不到FlowVisor的存在；(b)各个虚网之间是隔离的，即使在广播条件下各个虚网的流量也必须实现隔离；(c)虚网划分策略是丰富且可扩展的，由于当前网络虚拟化的技术还不成熟，因此划分虚网的策略需要是灵活的、模块化的且可扩展的。

FlowVisor主要由FVClassifier、FVSlicer和FlowSpace数据库三部分组成。FVClassifier用于维护与物理OpenFlow交换设备的连接，处理IO请求并记录每个物理设备的端口、性能等信息，每个FVClassifier对应一个OpenFlow交换设备。FVSlicer用于维护与控制器的连接，管理OpenFlow会话并对控制器下发的信令进行处理。当流从一个物理的OpenFlow交换设备到达FlowSpace后，FlowSpace根据数据库中切片规则，把OF消息交给本切片内的FVSlicer发送给连接的控制器。



FlowVisor的核心主要是对上行消息的映射和对下行信令的过滤。如图1所示，控制器和交换机与FlowVisor的所有联系都通过Poll Loop来交互，交换机交给控制器的数据包通过FlowVisor的OFSwitchAcceptor模块接收后，FVClassifier对数据包的来源进行分析，再连接到FlowVisor中的数据库与Flowspace进行匹配，从而决定此数据包应该交由哪个控制器来处理。同理，控制器下发给交换机的数据包会通过相反的过程到达交换机。

FlowVisor安装完成后，它的配置文件位于/etc/flowvisor/config.json。配置文件中包括简单的Slice、FlowSpaceRule样例，以及FlowVisor的监听端口、日志等级、版本等信息。api\_webserver\_port是xml请求的监听端口，默认为8080； api\_jetty\_webserver\_port是JSON请求的监听端口；listen\_port是交换机的监听端口，默认为6633。参数配置如下所示：



## oepnvirteX

带你走进OpenVirteX之简单网络拓扑实验

<http://www.sdnlab.com/2861.html>

带你走进OpenVirteX之环境搭建

<http://www.sdnlab.com/2418.html>

### OpenVirteX体系结构之综述

<http://www.sdnlab.com/2323.html>

http://www.sdnlab.com/3502.html

<https://ovx.onlab.us/documentation/architecture/>

### install procedure

java:

download oracle java 1.7:

http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html

# set java environment

export JAVA\_HOME=/home/[Hadoop](http://www.linuxidc.com/topicnews.aspx?tid=13)/jdk1.7.0\_67

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=.:$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH

mvn

# set maven environment

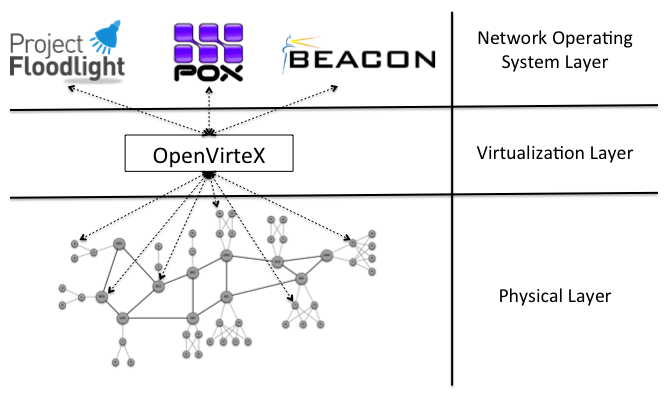
export M2\_HOME=/home/[Hadoop](http://www.linuxidc.com/topicnews.aspx?tid=13)/apache-maven-3.2.3

export PATH=$M2\_HOME/bin:$PATH

java编译中出现了Exception in thread “main" java.lang.UnsupportedClassVersionError

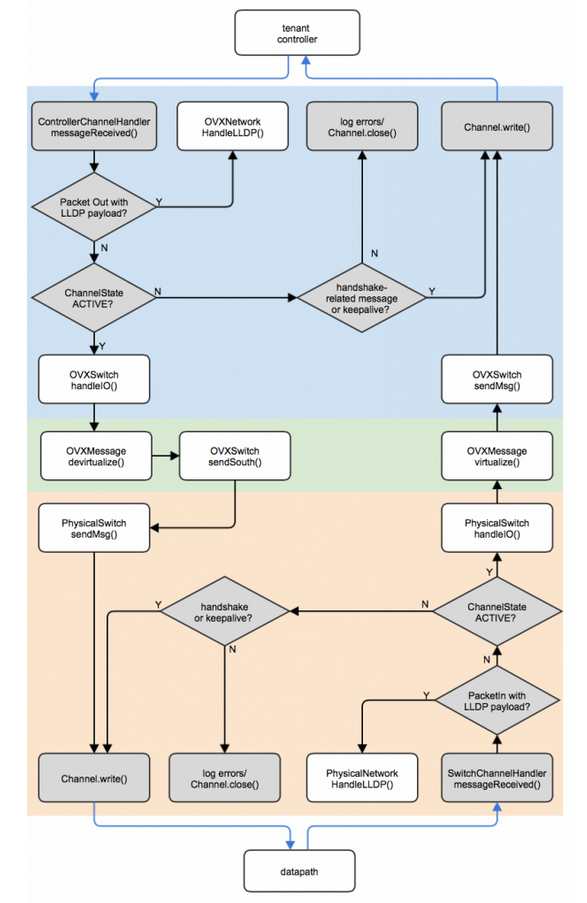
   这个问题确实是由较高版本的JDK编译的java [class文件](https://www.baidu.com/s?wd=class%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4mHFhnvwbPWbvm1-9PjmL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWmznjR4nj0vnH0kP1nvnHcd)试图在较低版本的JVM上运行产生的错误。

### Architecture



The only restriction we impose in OVX is that you cannot partition a single physical switch into multiple virtual switches.

If you prefer, you can specify a manual route by configuring the routing algorithm to manual and then using the setLinkPath call.



### What’s the difference between OpenVirteX and FlowVisor? When should I use which?

The best way to understand this is to see what both do to the packet header space (flowspace).

In short, OpenVirteX gives you a full packet header space (a virtual copy for your own virtual network), whereas FlowVisor lets you divide up a single packet header space into subsets to assign to your slices.

OVX supports the following command line parameters.

| **Name** | **Alias** | **Type** | **Description** | **Default** |
| --- | --- | --- | --- | --- |
| -p | –of-port | integer | OpenFlow port | 6633 |
| -h | –of-host | string | hostname | 0.0.0.0 |
| -dh | –db-host | string | database host | 127.0.0.1 |
| -dp | –db-port | int | database port | 27017 |
|  | –db-clear |  | clear database |  |
|  | –stats-refresh | integer | physical statistics polling interval (in seconds) | 30 |
| –ct | –client-threads | integer | number of threads to handle controller connections | 32 |
| –st | –server-threads | integer | number of threads to handle switch connections | 32 |
| –ub | –use-bddp |  | use BDDP for network discovery (only use if you know what you are doing) |  |

## [KVM详解](http://blog.chinaunix.net/uid-30022178-id-5749329.html)

http://blog.chinaunix.net/uid-30022178-id-5749329.html

# NetASM Programming

First, we will provide a brief overview of the NetASM language and its instruction set. We will then take you through the process of installing the NetASM assembler on your VM and testing it by running Hub and stateful MAC learner examples. After the overview, you will be asked to augment the stateful MAC learner example with an Access Control List (ACL) table to implement a Layer 2 firewall. We will provide more details on creating and submitting the code later in the instructions. As always, make sure that you follow each step carefully.

https://github.com/NetASM/NetASM-python/wiki

## DSL

<http://www.zhoujingen.cn/blog/2748.html>

The NetASM instruction set has three types of instructions:

* **Initialization**: to create state elements (like tables and registers)
* **Topology**: to define how the packet is traversed and processed in the data plane
* **Control**: to provide an external control to populate the states (i.e., over OpenFlow or other interfaces)

## Assigment

To implement this stateless firewall, you will need to perform the following tasks:

* Create a new ACL table using the MKT instruction and update the initialization code.
* Update the topology code using the topology instructions BRTF and DRP, such that only communication between the MAC pair entries listed in the ACL table are allowed.
* Write firewall rules in the ACL table with allowed MAC pairs, using the control instruction WRT.

### Demand

By contrast, in this assignment, using NetASM, you will change the layout of an existing switch i.e., stateful MAC learner, and will augment the layout by adding a *stateless* layer 2 firewall stage that sits after the MAC learner and enables communication only for the source and destination MAC pair for which there is an entry in the Access Control List (ACL). *By stateless, we mean that the state elements (like table and registers) are updated by the remote controller and not the switch itself. This is a perfect example showcasing a combination of both stateful and stateless operations.*

## Hashkell

<http://fleurer-lee.com/lyah/chapters.htm>

http://learnyouahaskell.com/introduction#about-this-tutorial

H-99: Ninety-Nine Haskell Problems

https://wiki.haskell.org/H-99:\_Ninety-Nine\_Haskell\_Problems

## P4

P4入门教程：搭建开发和实验环境

<http://mp.weixin.qq.com/s/r5SmE5DPm1RVd6K1YxVPBQ>

[4] P4官网 <http://p4.org/>  
[5] P4语言规范 <https://p4lang.github.io/p4-spec/>  
[6] P4邮件列表 <http://lists.p4.org/mailman/listinfo/>  
[7] Barefoot官网 <https://www.barefootnetworks.com/technology/>

P4语言编程详解

<http://www.sdnlab.com/17882.html>

P4入门教程（2）：P4程序的编译、运行验证与分析

<http://www.sdnlab.com/19936.html>

NSP4——P4网络模拟器的使用

<http://www.sdnlab.com/19370.html>

**P4 SIGCOMM 2015 Tutorial**

<https://github.com/p4lang/tutorials/tree/master/SIGCOMM_2015>

**P4：编写协议无关的包处理器**

http://www.maojianwei.com/2016/06/15/P4-Programming-Protocol-Independent-Packet-Processors/

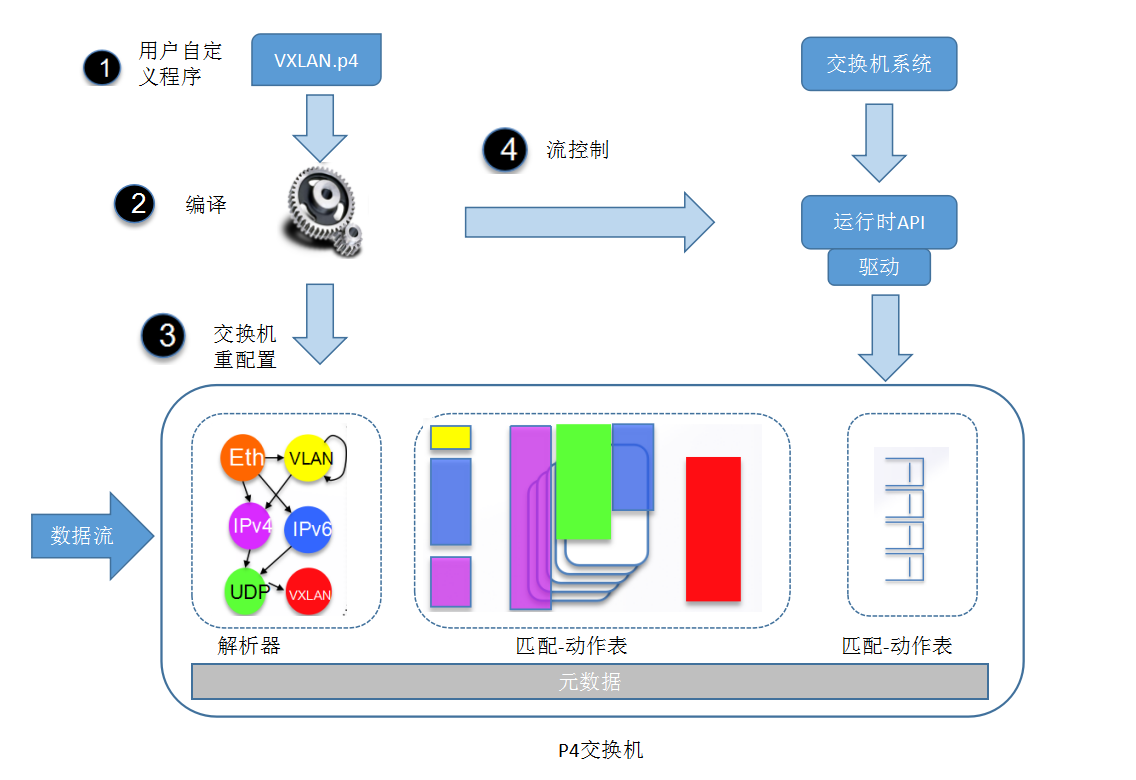
### P4架构及特性

**（1）协议无关性**  
网络设备不与任何特定的网络协议绑定，用户可以使用P4语言描述任何网络数据平面协议和数据包处理行为。这一特性通过自定义包解析器、匹配-动作表的匹配流程和流控制程序实现。

**（2）目标无关性**  
用户不需要关心底层硬件的细节就可实现对数据包的处理方式的编程描述。这一特性通过P4前后端编译器实现，前端编译器将P4高级语言程序转换成中间表示IR，后端编译器将IR编译成设备配置，自动配置目标设备。

**（3）可重构性**  
允许用户随时改变包解析和处理的程序 ，并在编译后配置交换机，真正实现现场可重配能力。  
为了实现上述特性，P4语言的编译器采用了模块化的设计，各个模块之间的输入输出都采用标准格式的配置文件，如p4c-bm模块的输出作为载入到bmv2模块中的JSON格式配置文件。P4的架构图如图1所示。

### P4程序工作流程



### P4的开发环境笔者认为可以粗略分为三种：

* 第一种是商用开发环境，诸如Barefoot出品的Capilano™，专门为其自家设计的芯片Tofino而推出的一款IDE。
* 第二种是面向FPGA等硬件运行条件的开发环境，这种情况下P4程序大致需要先使用前端编译器来编译生成一种中间表示文件，然后再使用后端编译器根据中间表示文件生成目标运行文件。中间表示文件即IR文件，类似介于C语言和汇编之间的链接文件。目标运行文件类似windows平台的exe文件，linux平台的可执行二进制文件，以及能直接加载/烧写入FPGA运行的文件。
* 第三种是面向软件交换机的开发环境，这种环境对于接触过OpenFlow开发和Open vSwitch（OVS）的朋友应该很熟悉。这种情况下P4程序只需要经过一次编译过程，生成数据平面的JSON格式描述文件，最后在启动软件交换机时将JSON描述文件导入即可。

# Pyretic Firewall Assignment

## Assignment

A Firewall is a network security system that is used to control the flow of ingress and egress traffic usually between a more secure local-area network (LAN) and a less secure wide-area network (WAN). The system analyses data packets for parameters like L2/L3 headers (i.e., MAC and IP address) or performs deep packet inspection (DPI) for higher layer parameters (like application type and services etc) to filter network traffic. A firewall acts as a barricade between a trusted, secure internal network and another network (e.g. the Internet) which is supposed to be not very secure or trusted.

In this assignment, your task is to implement a layer 2 firewall that runs alongside the MAC learning module on the Pyretic runtime. The firewall application is provided with a list of MAC address pairs i.e., access control list (ACLs). When a connection establishes between the controller and the switch, the application installs static flow rule entries in the OpenFlow table to disable all communication between each MAC pair.

### Demand

The pyretic\_firewall.py is populated with a skeleton code. It consists of a main function and a global variable (policy\_file) that holds the path of the firewall-policies.csv file. Whenever a connection is established between the Pyretic controller and the OpenFlow switch the main functions gets executed.

Your task is to read the policy file and update the main function. The function should install policies in the OpenFlow switch that drop packets whenever a matching src/dst MAC address (for any of the listed MAC pairs) enters the switch

# Kinetic Assignment

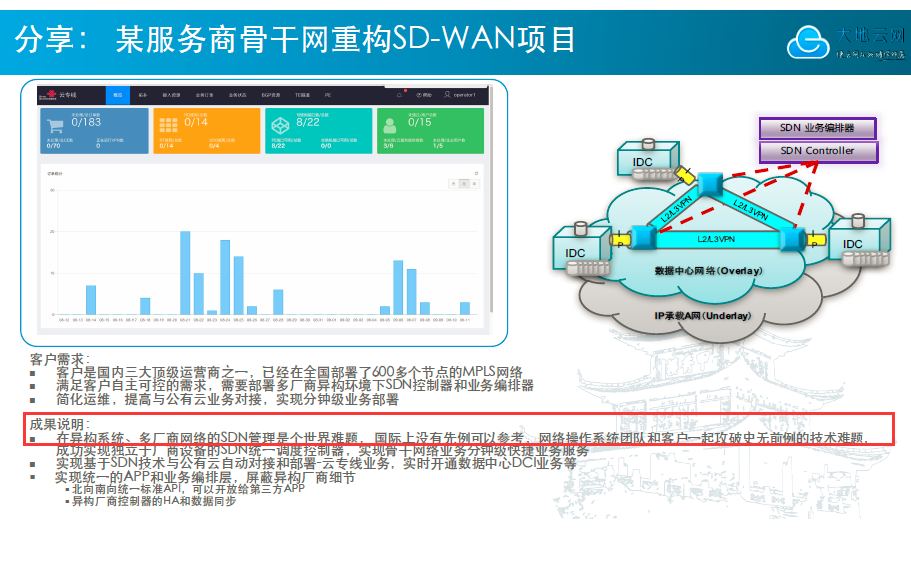
<http://kinetic.noise.gatech.edu/index.html>

# SDX Assignment

# Research point



云网融合，SDN在云数据中心的应用http://www.sdnlab.com/19626.html



# GNS

GNS3+eNSP+HCL模拟器从入门到精

http://edu.51cto.com/center/course/index/index?id=2453

GNS3从入门到精通

<http://mp.weixin.qq.com/s/EMLNfLjSPEbKrhbrU-8VYg>

# NFV

http://edu.sdnlab.com/training/378.html

什么是MANO？

NFV MANO（网络功能虚拟化管理和编排）是用于管理和协调虚拟化网络功能（VNF）和其他软件组件的架构框架。

NFV MANO有三个主要功能块：NFV编排器，VNF管理器和虚拟基础设施管理器（VIM）

1.NFV编排器由两层构成：服务编排和资源编排

2.VNF管理器能够管理VNF的生命周期

3.VIM能够控制并管理NFV基础设施，包括了计算、存储和网络等资源。

inux上虚拟网络与真实网络的映射

http://www.sdnlab.com/13539.html?from=timeline

<http://edu.51cto.com/topic/201.html>

# Docker

<https://mp.weixin.qq.com/s/o_r6kVNacnzg5mIxJy_NWw>

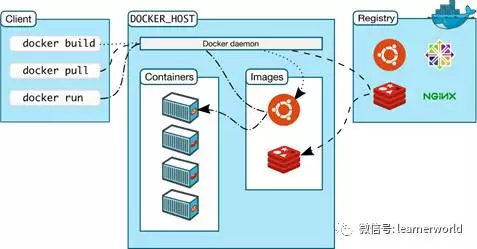
docker网络配置

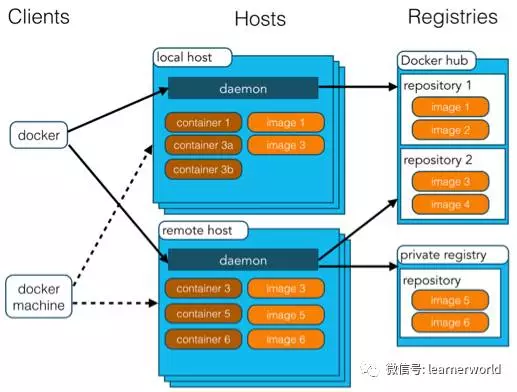
https://mp.weixin.qq.com/s/TqrdSudhMHMtmEQegBVsAw

## 1、Docker是什么？

Docker是一个开源的应用容器引擎，基于Go语言并通过Apache2.0协议开源。Docker可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化。

## 2、Docker架构模式





# Blog

http://www.brianlinkletter.com/

# Word

The next exercise will demonstrate some of the topology virtualization **tricks OVX has up its sleeve.**

POX itself has **a couple of** optional commandline arguments than can be used at the start of the commandline:

The switches may **take a little bit of time** to connect

A firewall acts as a barricade between a trusted, secure internal network and another network

Embraces innovation and can **think out of box**

You got a **trick up your sleeve**.

藏有玄机;鬼招

You may feel that this may increase the potential for **route flapping**

and **as such** we are free to start and stop any elements in that virtual network.

the pay **notwithstanding**

I can **honestly** say that I have never dreaded a day of work

 Being a professor **carries certain perks** over being a postdoc (e.g., more **cachet**, more autonomy, more pay)

When you **embark** on a new project

Thus, a **tenure**-track faculty position is really only one possibility

 In addition to **ensconcing** yourself in the ivory tower

Tough to reason about

since we were contributing to a classified project without being **privy** to the classified information

(or lack **thereof**)

you need the passion for creating new knowledge and the **tenacity** to push your visions through to conclusions when the **going gets tough**

 Pyretic enables network programmers and operators to write **succinct** modular network applications

 their structure makes them **amenable** to verification

**Converges** faster to target optimum on failure

and not worry about **mundane** things