

SDN 初试

SDN 初试

1. 课程说明
2. 学习方法
3. 本节内容简介
4. 推荐阅读
5. SDN 简介
 - 5.1 SDN 是什么
 - 5.2 SDN 能做什么
 - 5.3 SDN 的相关术语
6. SDN 工具
 - 6.1 交换机
 - 6.2 控制器
7. SDN 环境部署
 - 7.1 工具安装
 - 7.2 镜像安装
 - 7.3 源码安装
8. 总结
9. 作业

1. 课程说明

本课程为动手实验教程，为了能说清楚实验中的一些操作会加入理论内容，也会精选最值得读的文章推荐给你，在动手实践的同时扎实理论基础。

2. 学习方法

学习方法是多实践，多提问。启动实验后按照实验步骤逐步操作，同时理解每一步的详细内容。当然在学习本课程之前希望学习者有一定的网络基础知识。

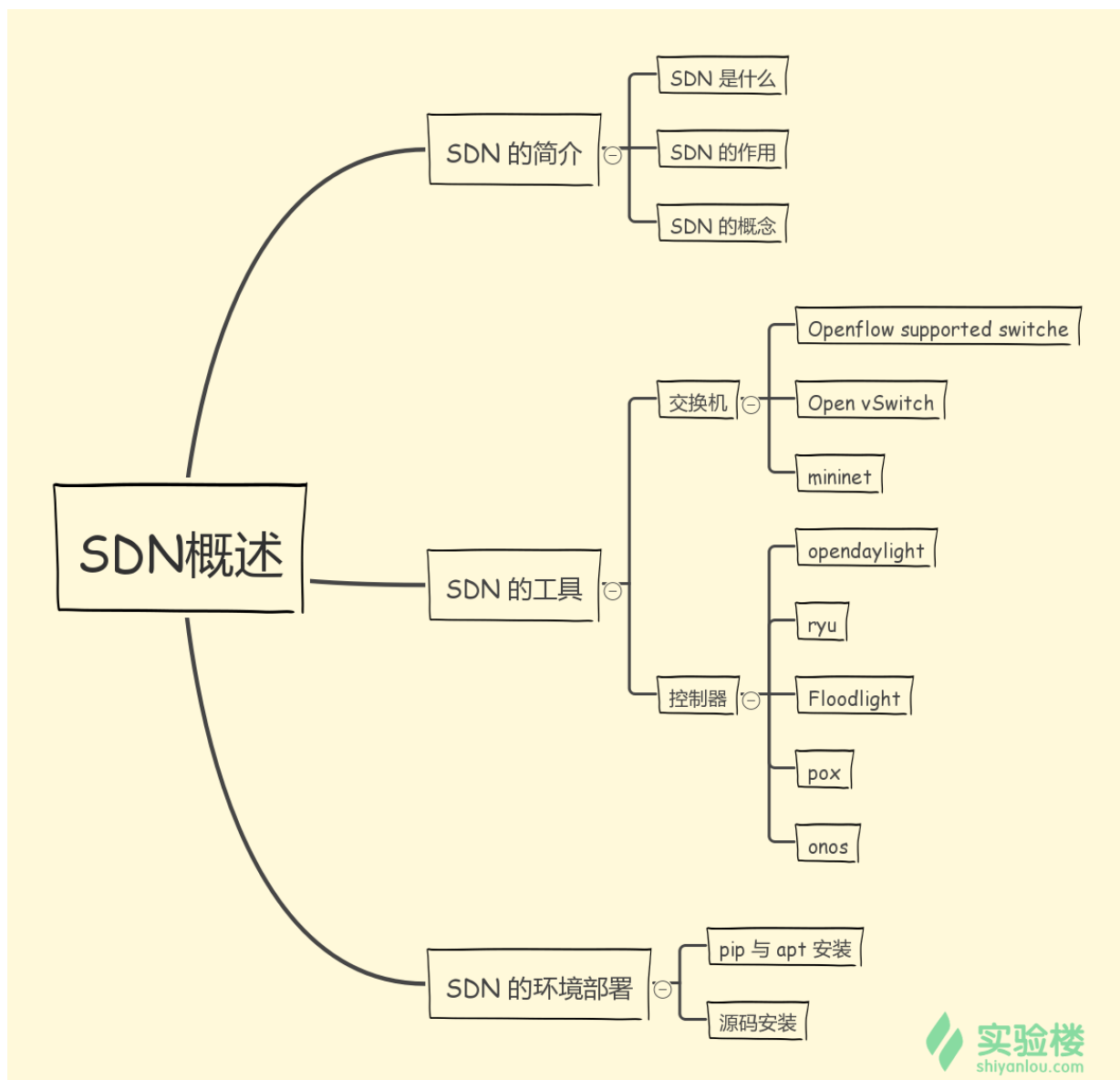
如果实验开始部分有推荐阅读的材料，请务必在实验之前或之后阅读，理论知识是实践必要的基础。

3. 本节内容简介

本实验中我们初步接触 SDN 的相关概念。需要依次完成下面几项任务：

- SDN简介
- SDN工具

- SDN环境部署



4. 推荐阅读

本节实验推荐阅读下述内容：

- [Ethane 项目，openflow 的基础](#)
- [openflow 的提出](#)
- [SDN 的道路](#)
- [openflow 的白皮书](#)
- [SDN 的发展史](#)

5. SDN 简介

在刚刚接触 SDN 时，我们内心最希望明白这样的三个问题，这也是我们初步学习的目标：

- SDN 是什么；
- SDN 能做什么；

- SDN 优势是什么；

解决了这样三个问题，我们便知道我们为什么要学习SDN，也拥有了我们学习的方向。接下来我们将来简单的回答这样三个问题。

5.1 SDN 是什么

SDN 是一种让网络灵活、简单的体系结构。

这样的结论是源于：SDN (Software Defined Networks, 翻译为软件定义网络) 这种方式设计的初衷就是为了我们能够通过软件更加便捷、更加灵活的控制我们的网络。

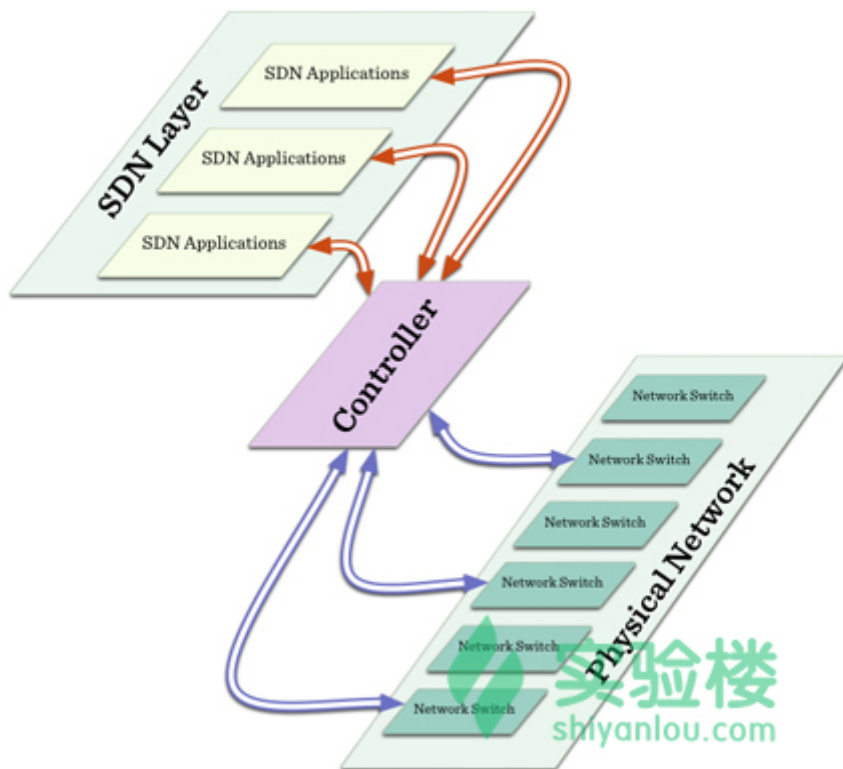
这样的网络体系结构，设计理念是时代发展的催化。

- 网络的时代，让我们能够不出门便知天下事，
- 移动信息化时代的到来，让我们的生活更加的便捷，让我们生活的方方面面都可以依赖网络，而这样的发展使得我们网络的业务极速的扩张，
- 业务需求如潮水一般，有潮起潮落，为了更合理的利用资源诞生了按需付费，弹性服务的云服务。

云服务以灵活吸引用户，而不同的用户有不同的业务需求，也就避免不了的拥有不同的网络需求，使得网络变得更加的灵活，而这样的灵活仅仅在应用层次上改变是不够的，最终的数据的传输需要通过路由，交换机等网元设备，若是这样的设备也能够灵活控制，这样的效率将是我们所期许的。灵活控制的变革也会使得网络前进一大步。

传统的网元设备从厂商引进，只能按照规则来配置、使用该设备，就像一个电视机，给了我们遥控器，我们只需要知道如何使用即可，但是当拥有大量设备时我们只能够一台台的去配置，并且若有漏洞，或者有定制化的需求出现时，使用者将是无可奈何的。

SDN 这样的体系结构使用自定义的软件来主导这一切，自然而然的解决了问题。



5.2 SDN 能做什么

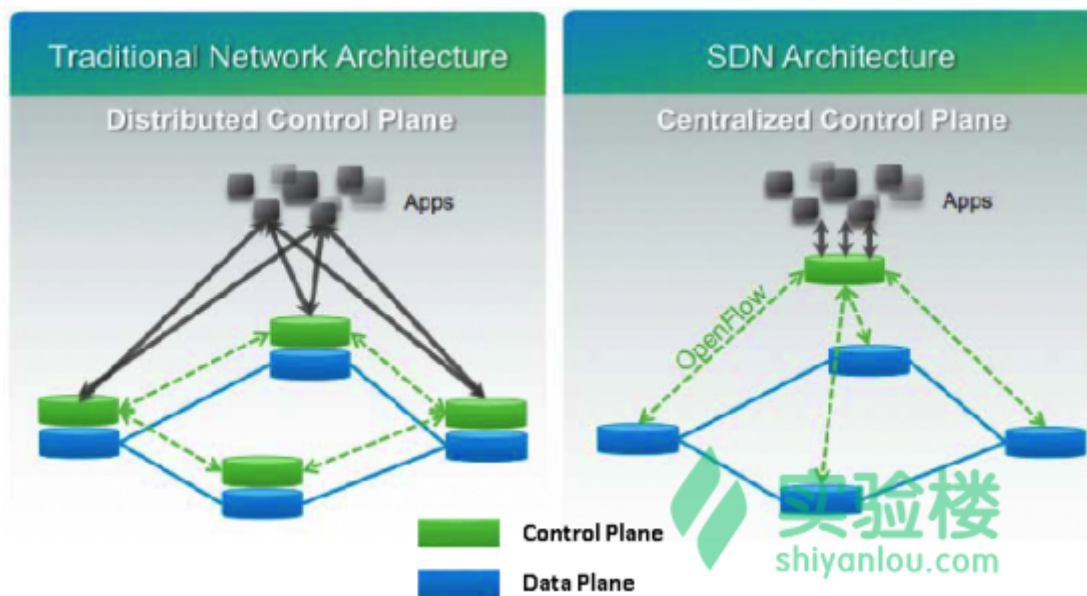
在 SDN 的体系架构中定义这样的三个平面：

- 业务平面：该层面又可细分为这样两个平面：
 - 管理平面：负责一些组件的初始化、配置等工作
 - 应用平面：负责将用户的请求通过某种方式发送给下一个层面，控制平面。
- 控制平面：负责将应用平面的请求转换为实际的操作，从而控制基础网元设备的行为，起一个承上启下的作用，可以说是SDN 体系结构中的核心，该平面就像公司的领导一样。
- 数据平面：负责数据转发，该平面就像公司中的搬运工一样。

这样的体系结构实现了控制平面与数据平面的完全分离（逻辑上的分离与物理上的分离），而这样的分离就像是科幻电影中将所有思考的能力都交给了人工智能的大脑，而其他执行命令的机器人并没有思考的能力，只能够接受大脑的支配，大脑的全权掌控能力使得即使出现了什么问题亦或者是由新功能的增加我们也能够及时升级系统来补救或者升级，而不用每个机器人都召回升级，并且这样的集中掌控能力使得我们定制化也将更加的方便。

不再像传统的网元，所有的逻辑控制都由厂商定制好，无法自由变动，统统在黑盒子里。

当然学习过网络基础（类似CCNA 课程）的同学会说其实在传统的设备中控制平面与数据平面也是相互分离的，由控制器（CPU）来完成一些灵活的功能，由线卡来负责数据在端口间的转发。



(此图来自于 Cisco)

传统的设备其实在物理结构上是紧密耦合的，这样做虽然能够使得转发效率更高，对会有其他的一些弊端，例如分布式的控制需要逐一配置、可能部分的设备数据流采样会导致CPU 的升高等问题。

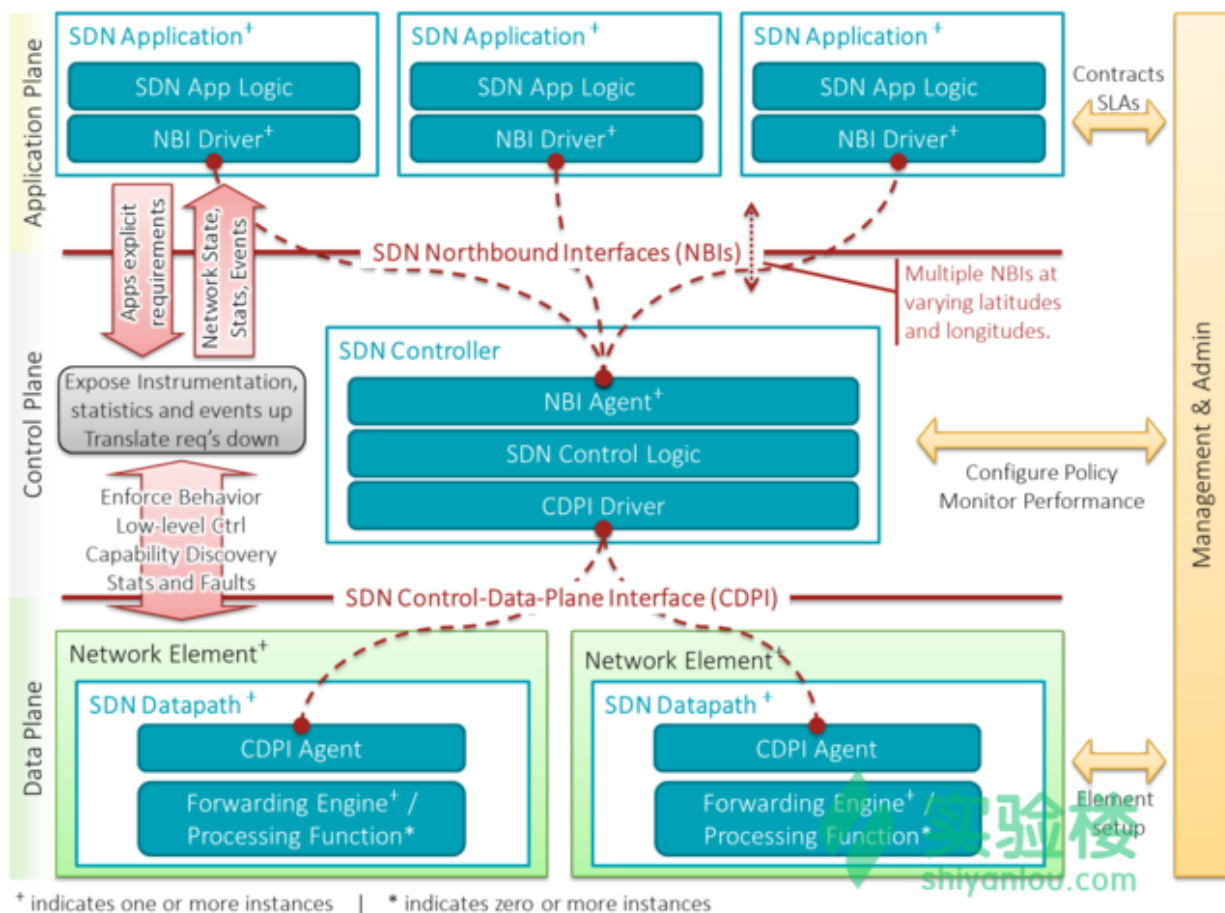
这样的对比并不是为了说SDN 的体系结构是完美无缺的，SDN 在发展时期也会有会有一些的弊端，或者不如意的地方，但是在整体结构上是比传统的结构更加的灵活，更适合于当前的大环境的使用，当然 SDN 发展至今也有十个年头，逐渐的被大部分的厂家所认可，甚至实施了部署，例如SDN 最为成功的案例 Google 利用 SDN 搭建的数据中心网络 B4，还有 Vmware、Juniper、Cisco、Dell 等等大巨头纷纷投入 SDN 的怀抱，足以看出 SDN 的吸引力。

5.3 SDN 的相关术语

从上文我们得知 SDN 的网络体系中主要分为业务平面、控制平面、数据平面。

通常工作于业务平面的是一些SDN 应用，也就是一群 Application，针对用户的一些需求；工作于控制平面是控制器（Controller），也就是控制数据实际走向的核心；工作于数据层的是各个网元设备，也就是一些交换器等，真正转发数据的工具。

将这三个平面串起来的是openflow 协议与 API，通过 API 与 openflow 协议来相互通信，这其中控制平面与业务平面相互通信的接口通常称为北面接口（Northbound Interfaces），与之对应的控制平面与数据平面相互通信的接口通常称为南面接口（Southbound Interfaces）。



(此图来自于 [wikipedia SDN Architecture Overview \(PDF\)](#), Version 1.0, December 12, 2013. CC BY-SA 3.0)

openflow 协议可以看出是 SDN 的关键，SDN 发展初期、概念不明确时，openflow 一度是 SDN 的代名词。

当然 openflow 也存在一些问题，所以不断的推出新版本，为此创始人 Nick McKeown 教授等人提出了 P4 (Programming Protocol-Independent Packet Processors)，华为公司也提出了相应的解决方案 POF (Protocol Oblivious Forwarding)。由此可以明白 openflow 不等于 SDN，他只是一个最典型的、被当前所广泛认可的一个南向协议而已。

通过上文所述或许你对 SDN 的印象还是朦胧的，但是至少我们知道了：

- SDN 是一个新的体系结构；
- SDN 是为了解决管理复杂、封闭、不灵活而诞生；
- SDN 的核心思想：
 - 控制平面于数据平面的分离；
 - 由控制平面核心控制器集中式控制，并可编程化；

6. SDN 工具

为了更深入的了解 SDN，我们需要更多的动手实验，而实验需要实验的环境，环境主要在于交换机的模拟与控制器上。

6.1 交换机

交换机的实验方式主要有这样的几种方式：

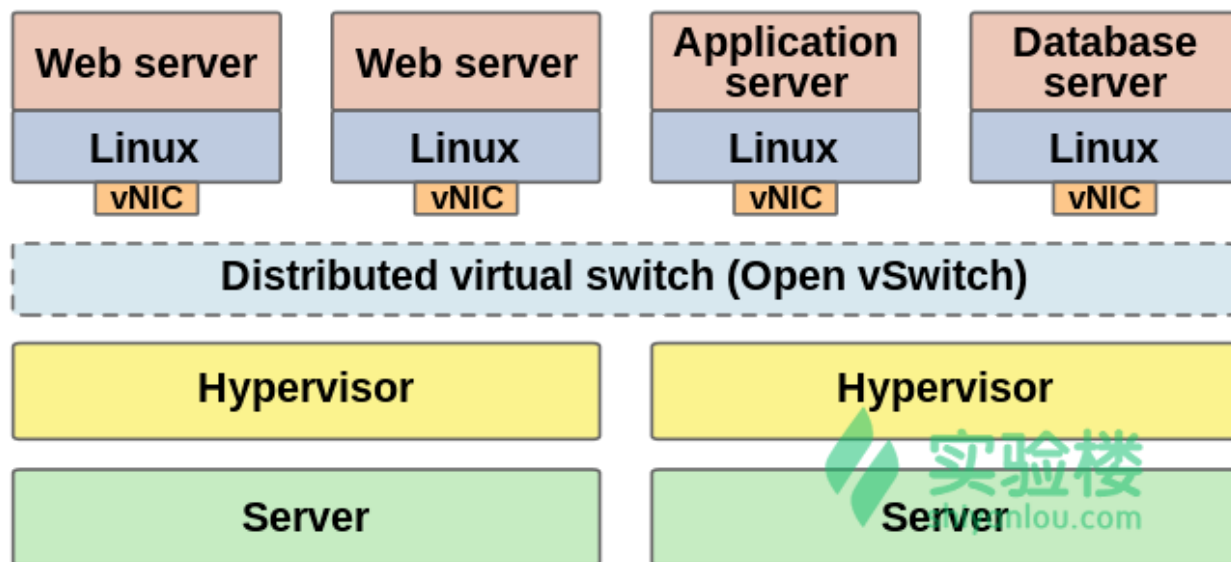
- 直接使用支持 openflow 真机；
- open vswitch；
- mininet；

第一种方式便是直接使用支持openflow 的真实 switch：

- 优点：这样的方式能够直接见到效果，还能够见到实物，研究的兴趣更高，并且能够避免一些模拟器所带来的外界因素（如以前使用GNS3的时候，该模拟器虽然使用真机镜像，但是很多功能是不支持的，因为有些功能是使用硬件完成的，所以会导致即使正确的配置也会出错）。
- 缺点：这样的实验方式的成本很高。

第二种便是使用 open vswitch：

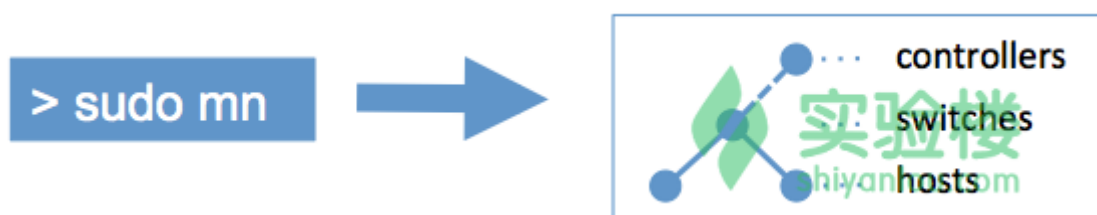
- 优点：只需要安装软件与一定配置就可以将一台物理改造成一个交换机，并且因为虚拟化技术使得该设备上可以创建多个交换机，不会因为需要多台交换机环境便增加多台物理设备。
- 缺点：在搭建与使用上有一定的门槛。



（此图来自于wikipedia By Goran tek-en - Own workDerived from:virtual networking）

第三种便是使用 mininet 模拟器：

- 优点：使用简单；
- 缺点：只是模拟的网络拓扑环境，并不是真实的网络。



其实还可以利用支持 openwrt 的路由器，通过 openwrt 给路由器刷 openflow，但这样的方式较为麻烦，有兴趣的同学可以尝试一番。

在接下来的实验中我们将使用 mininet 模拟器来完成我们后续的学习。而mininet 是什么？

- mininet 是一个由 python 开发的一个轻量级的模拟工具；
- mininet 虚拟节点的创建是通过 Linux Network Namespace 来实现。

6.2 控制器

选择好了交换机的模拟工具，接下来我们需要选择一款合适的控制器框架。

当前有很多的控制框架构架，被广泛使用的主要有这样几种：

- Floodlight: 使用 Java 开发，由 Big Switch Networks 设计的开源控制器；
- Opendaylight: 使用 Java 开发，由 Linux 基金会推出的一个开源项目，其会员有Cisco（很多项目的主导），Juniper, HP, IBM, Brocade，微软，Huawei 等等知名供应商，功能强大，但也十分庞大；
- Open contrail: 使用 Java 开发，由 Juniper 主导设计、研发的开源控制器；
- onos: 使用 Java 开发，由 ON.Lab（由最早创造发明 SDN 的斯坦福与伯克利等大学联合运营商与制造商发起的开源社区）实现发布的首款开源的SDN 网络操作系统，主要面向服务提供商、企业骨干网；
- Pox: 使用 python 开发，支持的协议不是太多；
- Ryu: 使用 python 开发，由 NTT 公司设计研发的开源控制器，较于pox 更新支持的 openflow 协议版本更多，支持的协议更多；



（此图来自于 [linkedin](#)）

为了更容易上手我们选择了 Ryu 控制器，在接下来的实验中我们都使用 Ryu。

7. SDN 环境部署

7.1 工具安装

我们可以通过 apt、pip 这样的安装包管理工具来装我们所需的mininet 与 Ryu 工具。

通过这样的方式我们可以十分便捷的安装mininet:


```
#更新源
sudo apt-get update


#安装 mininet
sudo apt-get install -y mininet
```

稍等片刻便安装完成，我们可以通过这样的方式来检测我们是否安装成功：

```
sudo mn
```

若是看到这样的输出则表示成功安装：

```
shiyanolou:app/ $ sudo mn [15:43:49]
*** Error setting resource limits. Mininet's performance may be affected.
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet> 
```



正常启动不会有 error 信息，上面的 error 信息是环境所导致的，出现这样的情况一般有两种：

- 若是在 ubuntu 中部署则可能是因为你的内核版本在3.13.0-96 到 3.13.0-100 左右，因为在这个版本之间无法使用 `sudo sysctl -w net.ipv4.neigh.default.gc_thresh1=4096` 命令来修改 ARP 的缓存大小。
- 若是在 docker 中部署则可能是因为没有给予privilege 权限。

没有参数，直接运行 `sudo mn` 默认会创建一个简单的拓扑，由一个控制器、一个交换机与两个节点构成，并进入 mininet 的命令行界面。

输入 `exit` 我们便能推出 mininet 的命令行界面。

在成功安装 mininet 之后我们只需要一行简单的命令即可安装ryu:


```
sudo pip install ryu
```

稍等片刻久安装成功了，我们直接输入这样的命令查看是否能够正常使用：

```
ryu-manager
```

若是看到这样的命令输入，说明安装成功能够正常使用：

```
shiyanolou:app/ $ ryu-manager  
loading app ryu.controller.ofp_handler  
instantiating app ryu.controller.ofp_handler of OFPHandler
```



7.2 镜像安装

mininet 是一个开源的模拟工具，我们可以直接访问[github 中的该项目](#)可以看到其源码与相关的文档，在 wiki 中我们看到官方还为我们提供了镜像，使我们安装更便捷。

通过[该链接](#)我们可以下载到 VMware 与 VirtualBox 的镜像，直接导入镜像我们即可使用。

7.3 源码安装

最后一种方式便是通过源码安装，这样的方式可以获得最新版本的mininet。

首先通过 git 命令下载最新的源码：

```
git clone git://github.com/mininet/mininet
```

下载成功之后进入项目目录中：

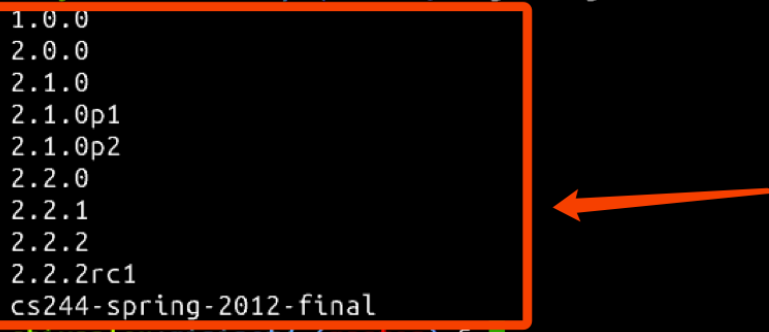
```
cd mininet
```

因为是直接下载的源码，我们可以选择我们想要安装的版本，可以通过 `git tag` 看到：

```

shiyanolou:~/ $ git clone git://github.com/mininet/mininet
正克隆到 'mininet'...
remote: Counting objects: 9213, done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 9213 (delta 21), reused 0 (delta 0), pack-reused 9171
接收对象中: 100% (9213/9213), 2.85 MiB | 741.00 KiB/s, done.
处理 delta 中: 100% (6082/6082), done.
检查连接... 完成。
shiyanolou:~/ $ cd mininet
shiyanolou:mininet/ (master) $ git tag
1.0.0
2.0.0
2.1.0
2.1.0p1
2.1.0p2
2.2.0
2.2.1
2.2.2
2.2.2rc1
cs244-spring-2012-final
shiyanolou:mininet/ (master) $

```




我们通过这样的命令切换到相关版本的源码下：

```
git checkout -b 2.2.2
```

然后通过这样的命令来安装：


```
util/install.sh -a
```

`-a` 参数会帮我们装所有我们需要的相关工具如openflow、wireshark、dissector、pox。我们也可以使用 `-nfv` 参数来精简安装。

```

in'
libtool: install: /usr/bin/install -c cbench /usr/local/bin/cbench
make[2]: 没有什么可以做的为 `install-data-am'。
make[2]:正在离开目录 `/home/shiyanlou/oflops/cbench'
make[1]:正在离开目录 `/home/shiyanlou/oflops/cbench'
Making install in doc
make[1]: 正在进入目录 `/home/shiyanlou/oflops/doc'
make[1]: 没有什么可以做的为 `install'。
make[1]:正在离开目录 `/home/shiyanlou/oflops/doc'
Enjoy Mininet!
shiyanolou:mininet/ (2.2.2) $

```



[16:42:46]

Brackets



看到 **Enjoy Mininet!** 这样的标题说明我们安装成功，同样我们可以用上述方式来检验mininet 是否成功安装（在最新的 mininet 源码中提供了 ryu 的安装，只是其脚本有问题所以会安装失败，有兴趣的同学可以修改一下 install.sh 脚本，在 462 行左右）。

同样我们通过源码来安装 ryu。

若是为了使得 python 的运行环境不被干扰我们可以使用 virtualenv：

```
#安装 virtualenv
sudo pip install virtualenv

#创建文件夹
mkdir SDN

#进入该文件夹
cd SDN

#创建隔离环境
virtualenv env

#使用该隔离环境
source env/bin/activate

#下载 ryu 的源码
git clone git://github.com/osrg/ryu.git

#进入该目录
cd ryu

#安装 ryu
python ./setup.py install

#安装 ryu 的相关依赖
pip install -r tools/pip-requires
```

我们可以使用上述相同的办法测试我们是否成功的安装了ryu。

8. 总结

本节实验中我们学习了以下内容：

- SDN 简介
- SDN 工具
- SDN 环境部署

请务必保证自己能够动手完成整个实验，只看文字很简单，真正操作的时候会遇到各种各样的问题，解决问题的过程才是收获的过程。

9. 作业

有条件的同学尝试在本地源码安装mininet 与 ryu 。

