Mininet

快速部署SDN工具介绍

为什么选择 | 命令详解 | OpenFlow协议
| 案例应用 | 获取方式

优速网络　　SDN项目组

为什么选择

凡事必有其原因，当中有何隐情，让我们一同探索为何要选择Mininet

背景

SDN真的那么容易吗？

控制器研究

开源成为不可抵挡的趋势。NOX，POX，Floodlight等均采用公开源代码的形式，任何人都可以学习SDN，只要有相应的IT编程能力，都可以为SDN的控制器的完善做出贡献。各大设备厂商也启动了OpenDayLight项目。

SDN搭建

光有控制器还不能构成完整SDN网，但当前硬件SDN交换机还很少，也很难找到。难道要让我用OpenVSwitch去手动搭建复杂的网络吗？哦，亲，我可以告诉你我已经不感兴趣了吗？

特征

什么是Mininet，他又有何好处？

1 快速

可以快速搭建一个SDN网络，所有搭建工作均在同一台计算机上完成

2 高效

搭建的SDN网络可以媲美真实硬件环境的网络，它具备SDN网络应有的特性

3 灵活

可通过命令行、用户API、Python应用来创建主机、交换机、控制器

优势

作为轻量级软定义网络研发和测试平台，优势是什么？

- 支持Openflow、OpenvSwitch等软定义网络部件
- 方便多人协同开发
- 支持系统级的还原测试
- 支持复杂拓扑、自定义拓扑
- 提供python API
- 很好的硬件移植性（Linux兼容），结果有更好的说服力
- 高扩展性，支持超过4096台主机的网络结构

组合

对SDN网络研究需要什么样的工具组合？

Mininet 部署工具

控制器

抓包工具

注意

研究SDN网络的注意事项

1 抓包工具

wireshark 是一款抓包工具。使用wireshark检测lo网卡，并通过过滤of协议可以看到OpenFlow的网包。

2 控制器

这里的控制器（controller）我们默认使用Floodlight，其可以获取网络拓扑、向交换机写入流表，当然还有很多其他功能需要进行更为详细的配置，我们这里仅对其基本功能进行了解。

命令详解

基本命令行

1 **--topo**

定义了一个将在mininet启动时加载的拓扑

2 **--switch**

定义mininet要使用的交换机（默认使用OVSK即OpenVSwitch交换机）

3 **--controller**

定义要使用的控制器，如果没有指定则默认的控制器将默认使用hub behavior

拓扑

Mininet创建拓扑的方法

1

线形拓扑

交换机连接呈线形排列，且每个交换机所连接主机数目多为一个

2

树形拓扑

交换机连接成树形排列，且每个交换机所连接主机一般有多个

3

自定义拓扑

用Python语言写脚本，执行脚本即可创建拓扑

执行命令格式： **sudo python <script name>**

拓扑创建命令介绍

1 **sudo mn --topo linear,4**

每个主机连接一个交换机，交换机之间互相连接
举例 ：4个hosts和4 个switches.

2 **sudo mn --topo single,3**

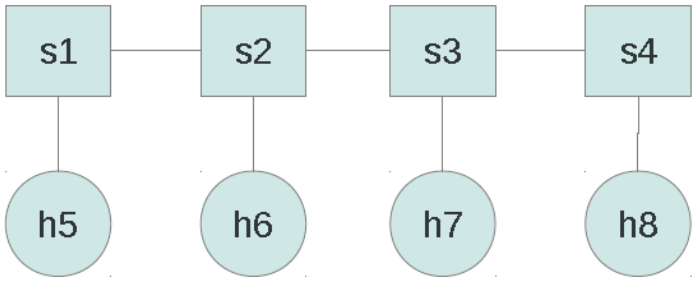
所有host都连接到同一个switch上
举例：3个hosts和1个switch.

3 **sudo mn --topo tree,depth=2,fanout=2**

通过depth和fan-out定义的树形拓扑

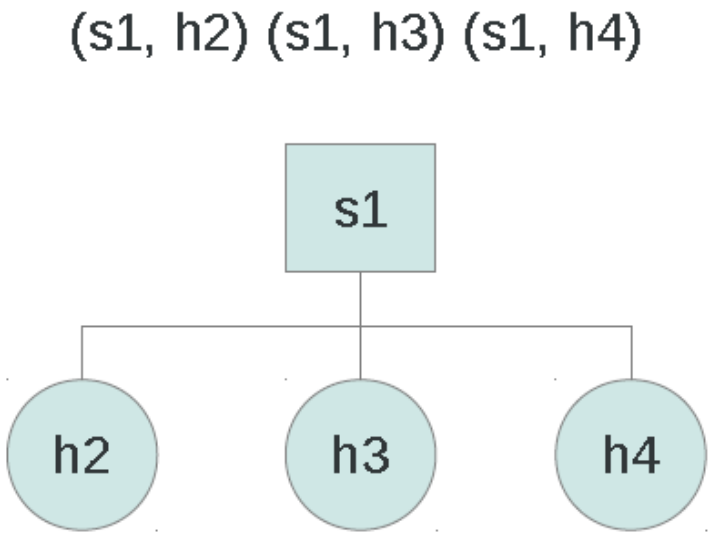
sudo mn --topo linear,4

(s1, s2) (s1, h5) (s2, s3) (s2, h6) (s3, s4) (s3, h7) (s4, h8)




```
from mininet.net
import Mininet
from mininet.topo
import LinearTopo
Linear4 =
LinearTopo(k=4)
net =
Mininet(topo=Linear4)
net.start()
net.pingAll()
net.stop()
```

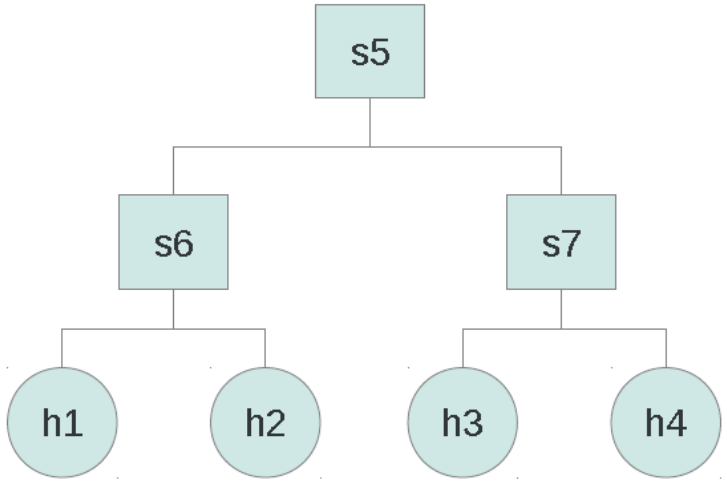
sudo mn --topo single,3




```
from mininet.net import
Mininet
from mininet.topo import
SingleSwitchTopo
Single3 =
SingleSwitchTopo(k=3)
net = Mininet(topo=Single3)
net.start()
net.pingAll()
net.stop()
```

sudo mn --topo tree,depth=2,fanout=2

(h1, s6) (h2, s6) (h3, s7) (h4, s7) (s5, s6) (s5, s7)




```
from mininet.net import Mininet
from mininet.topolib import TreeTopo
Tree22 = TreeTopo(depth=2, fanout=2)
net = Mininet(topo=Tree22)
net.start()
net.pingAll()
net.stop()
```

Mininet的常用交互命令

help	默认列出所有命令文档，后面加命令名将介绍该 命令用法
dump	打印节点信息
intfs	列出所有的网络接口
nodes	列出所有的节点信息
net	显示网络链接情况
iperf	两个节点之间进行简单的iperf□TCP测试
dpctl	在所有交换机上用dptcl执行相关命令，本地为 tcp□127.0.0.1:6634
iperfudp	两个节点之间用制定带宽udp进行测试

Mininet的常用交互命令

pingpair	在前两个主机之间互ping测试
link	禁用或启用两个节点之间的链路
pingall	所有host节点之间互ping
xterm	给定节点上开启xterm
source	从外部文件中读入命令
py	执行python表达式
sh	运行外部shell命令
quit/exit	退出

流表修改命令

- **dpctl dump-flows tcp:127.0.0.1:6634**

  可以看到更详细的流表信息。

  此时，流表为空，执行h2□ping□h3无法得到响应。因此我们需要通过dpctl手动添加流表项，实现转发。

- **dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2**
- **dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1**

  此时查看流表可以看到新的转发信息，同时可以在h2和h3之间ping通

命令学习进阶

```
sudo mn
--test pingall
--topo single,3
--controller=remote，
ip=192.168.208.144，
port=6633
```

- 创建一个独立的网络，有三个主机以非交互模式连接在一个交换机上
- 网络中的每个主机均完成Ping操作到其余的主机
- 这个命令使用远程的controller（控制器），其ip为192.168.208.144，端口为6633

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 1 switches
s1
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
*** Stopping 1 switches
s1 ...
*** Stopping 3 hosts
h1 h2 h3
*** Stopping 1 controllers
c0
*** Done
completed in 2.155 seconds
```

创建了三个主机和一个交换机

主机和交换机之间的连接

启动控制器和创建的交换机

执行所有主机间的ping操作

显示结果

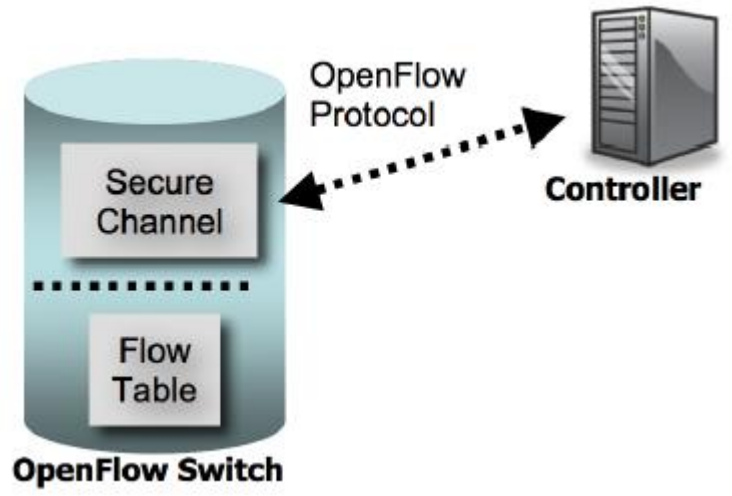
停止所有操作，并显示完成命令使用的时间

OpenFow协议

对OpenFlow协议的消息类型和连接流程进行介绍

概述

简要介绍OpenFlow协议



OpenFlow协议诞生于斯坦福大学，是创新型网络的研究产物。它所定义的网络架构不同于传统的网络架构的分布式控制方式OpenFlow协议的解决方案是集中式控制，正是由于这种思想产生了SDN（软件定义网络）的概念，基于OpenFlow协议的SDN的核心是数据层面和控制层面的分离，这样使得网络的交换机的数据转发更为简单、快速。OpenFlow协议用于控制器向交换机中传送转发表，交换机依此转发数据包。

消息

介绍OpenFlow协议的消息类型

1 Controller-to-Switch

控制器至交换机消息此类消息由控制器主动发出

2 Asynchronous

异步消息此类消息由交换机主动发出

3 Symmetric

对称消息，可以由控制器或交换机主动发起

Controller-to-Switch

Features	用来获取交换机特性
Configuration	用来配置Openflow交换机
Modify-State	用来修改交换机状态(修改流表)
Read-Stats	用来读取交换机状态
Send-Packet	用来发送数据包
Barrier	阻塞消息

Asynchronous

Packet-in	用来告知控制器交换机接收到数据包
Flow-Removed	用来告知控制器交换机流表被删除
Port-Status	用来告知控制器交换机端口状态更新
Error	用来告知控制器交换机发生错误

Symmetric

Hello	用来建立Openflow连接
Echo	用来确认交换机与控制器之间的连接状态
Vendor	厂商自定义消息

抓包

OpenFlow连接建立时抓包

Source	Destination	Protocol	Length	Info
192.168.208.144	192.168.208.143	OFP	74	Hello (SM) (8B)
192.168.208.143	192.168.208.144	OFP	74	Hello (SM) (8B)
192.168.208.144	192.168.208.143	OFP	74	Hello (SM) (8B)
192.168.208.144	192.168.208.143	OFP	74	Features Request
192.168.208.143	192.168.208.144	OFP	290	Features Reply (
192.168.208.144	192.168.208.143	OFP	98	Stats Request (C
192.168.208.143	192.168.208.144	OFP	78	Get Config Reply
192.168.208.143	192.168.208.144	OFP	1134	Stats Reply (CSM

流程

介绍OpenFlow连接建立时要控制器和交换机消息顺序

Message	Type	Description
Hello	Controller-> Switch	following the TCP handshake, the controller sends its version number to the switch
Hello	Switch-> Controller	the switch replies with its supported version number.
Features Request	Controller-> Switch	the controller asks to see which ports are available
Set Config	Controller-> Switch	in this case, the controller asks the switch to send flow Expirations
Features Reply	Switch-> Controller	the switch replies with a list of ports, port speeds, and supported tables and actions.
Port Status	Switch-> Controller	enables the switch to inform that controller of changes to port speeds or connectivity. Ignore this one, it appears to be a bug

案例应用

从网络创建代码到拓扑，讲述如何通过Mininet研究SDN网络及OpenFlow协议

创建第一个SDN网络

1

启动Mininet

输入命令：sudo　mn

```
*** 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:
```

创建了两个主机和一个交换机

定义节点间的连接

开启默认的控制器和建立的交换机

创建第一个SDN网络

2

在Mininet中输入命令

mininet > <command>

mininet> nodes
mininet> net
mininet> dump

available nodes are:
c0 h1 h2 s1

h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0
s1-eth2:h2-eth0
c0

<Host h1: h1-eth0:10.0.0.1 pid=9916>
<Host h2: h2-eth0:10.0.0.2 pid=9917>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=9920>
<OVSController c0: 127.0.0.1:6633 pid=9908>

3

观察控制器的情况

http://192.168.208.144:8080/ui/index.html

Dashboard Topology Switches Hosts

Live updates

Controller Status

控制器管辖内的所有机器

Hostname:	localhost:6633
Healthy:	true
Uptime:	28313 s
JVM memory bloat:	488116624 free out of 515899392
Modules loaded:	n.f.topology.TopologyManager, n.f.flowcache.FlowReconcileManager, n.f.devicemanager.internal.DefaultEntityClassifier, n.f.storage.memory.MemoryStorageSource, n.f.counter.CounterStore, n.f.restserver.RestApiServer, n.f.firewall.Firewall, n.f.core.FloodlightProvider, n.f.perfmon.PktInProcessingTime, n.f.devicemanager.internal.DeviceManagerImpl, n.f.linkdiscovery.internal.LinkDiscoveryManager, n.f.staticflowentry.StaticFlowEntryPusher, n.f.threadpool.ThreadPool, n.f.flowcache.FlowCache,

Switches (1)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:00:00:00:00:00:01	/192.168.208.143:51333	Nicira Networks, Inc.	0	0	0	2014年1月6日 下午5:05:17

Hosts (2)

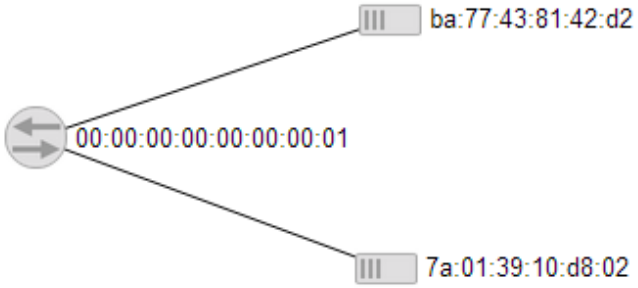
MAC Address	IP Address	Switch Port	Last Seen
7a:01:39:10:d8:02		00:00:00:00:00:00:00:01-2	2014年1月6日 下午5:05:26
ba:77:43:81:42:d2		00:00:00:00:00:00:00:01-1	2014年1月6日 下午5:05:26

Floodlight Dashboard Topology Switches Hosts

Live updates

Network Topology

控制器发现的网络拓扑



创建含两个switch的SDN网络

1

打开wireshark

登录到虚拟机命令行界面，打开wireshark，使其后台运行，命令为:**sudo wireshark &**

2

启动Mininet

输入命令： sudo mn --custom mininet/custom/topo-2sw-2host.py --topo mytopo --controller=remote，ip=<ip>, port=6633

这个创建命令好复杂，来我们研究一下

解释

创建的拥有两个switch的SDN网络命令分析

sudo mn --custom mininet/custom/topo-2sw-2host.py --topo mytopo --controller=remote，ip=<ip>, port=6633

- 这里的controller是另外创建的虚拟机，后面跟随的是其ip和端口
- 这里可以看到一个比较陌生的.py，这是使用的自定义拓扑模式，用python语言写的自定义拓扑。
- 除了使用mn命令进行交互式操作以外，mininet最为强大之处是提供api可以直接通过python编程进行灵活的网络实验。在mininet/example目录下给出了几个python程序的例子，包括使用gui方式创建拓扑、运行多个测试，在节点上运行sshd，创建多个节 点的tree结构网络等等。运行这些程序就可以得到令人信服的结果，而且这些程序大都十分短小，体现了mininet平台的强大易用性。

创建

创建的拥有两个switch的SDN网络

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
```

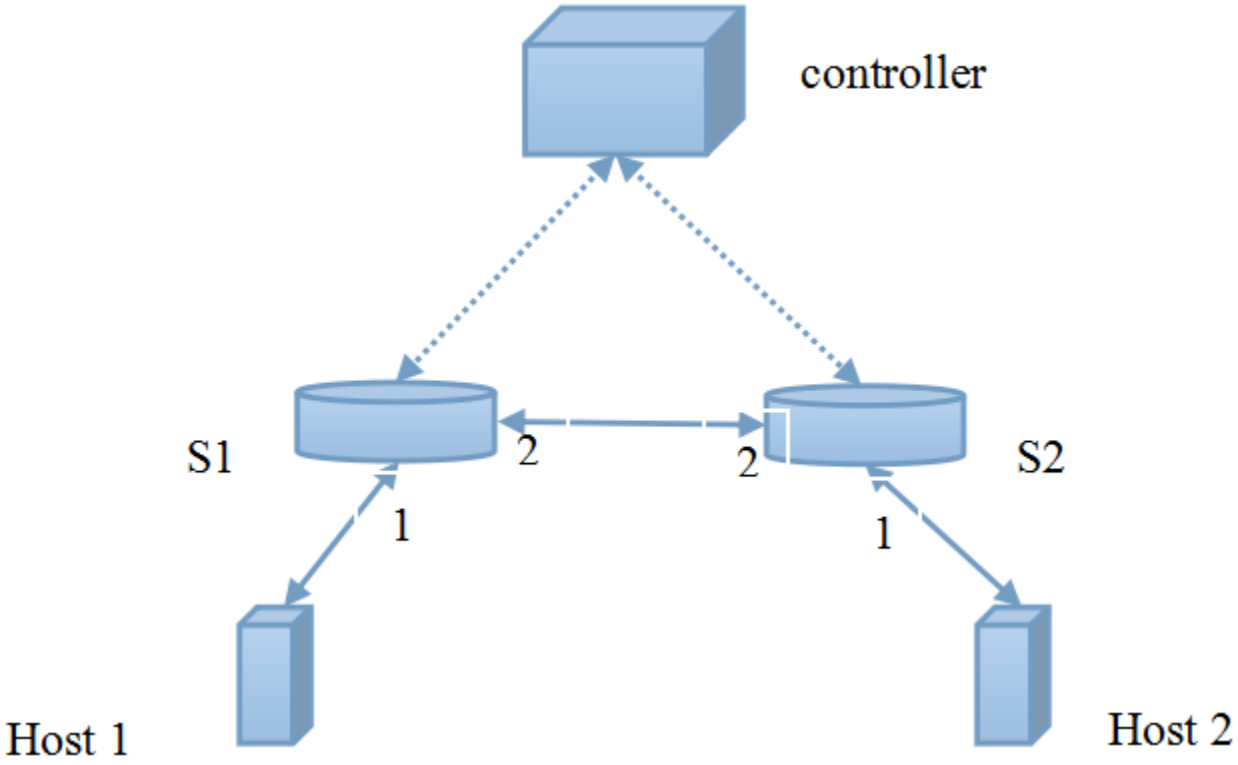
创建了2个主机和2个交换机，并添加控制器

定义节点间的连接

开启默认的控制器和建立的交换机

拓扑

创建的拥有两个switch的SDN网络



创建含两个switch的SDN网络

3

测试网络

通过进行ping操作，对网络进行研究
mininet > h1 ping h2

4

抓包情况分析

在步骤3进行后，通过wireshark进行抓包，分析抓包情况
对OpenFlow协议进行研究

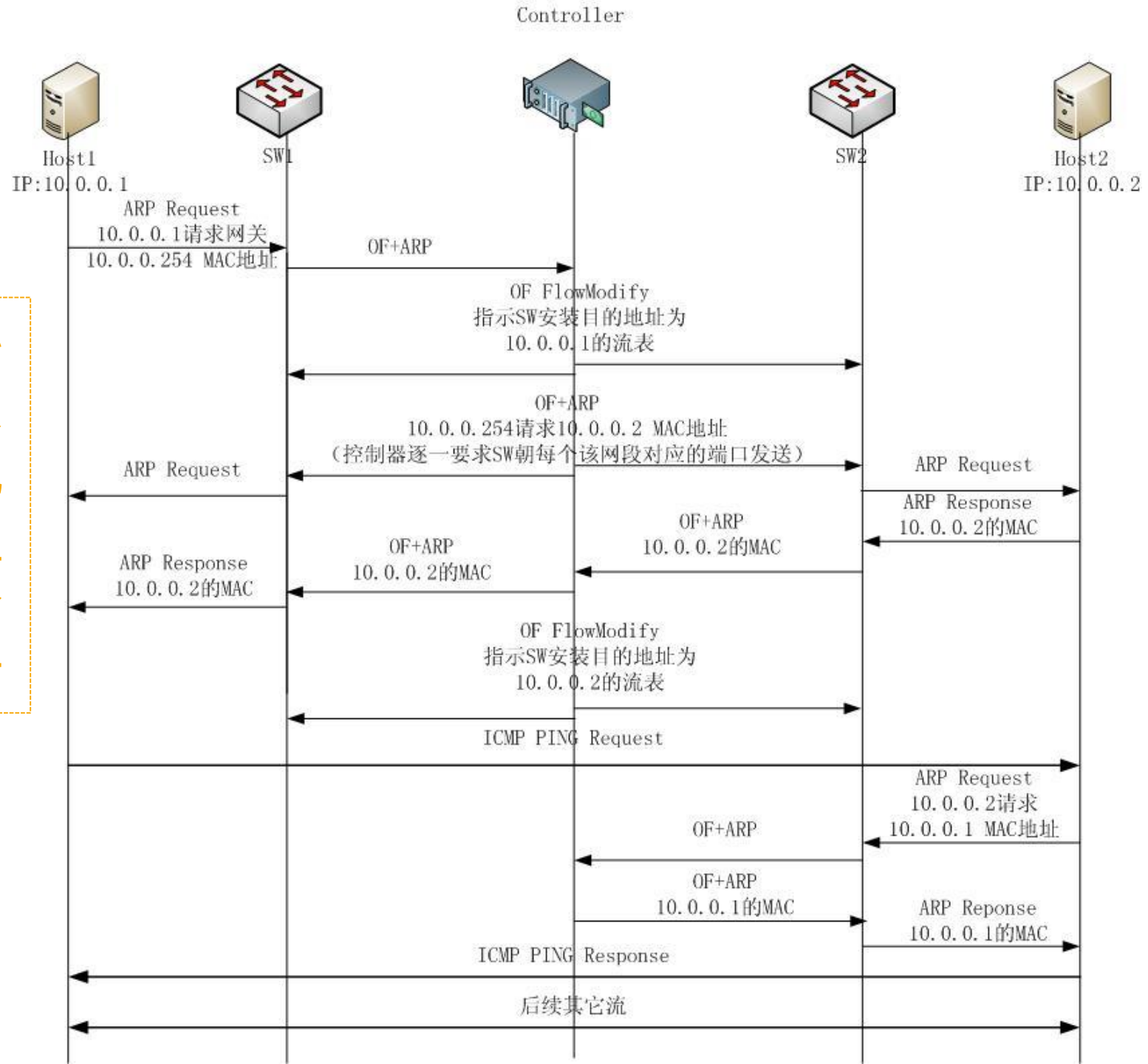
抓包

mininet > h1 ping h2

抓包情况

ba:ea:36:38:13:eb	Broadcast	OFP+ARP	126 Packet In
192.168.208.144	192.168.208.143	OFP	90 Packet Out
ba:ea:36:38:13:eb	Broadcast	OFP+ARP	126 Packet In
192.168.208.144	192.168.208.143	OFP	90 Packet Out
b2:3b:69:b5:58:a7	ba:ea:36:38:13:eb	OFP+ARP	126 Packet In
192.168.208.144	192.168.208.143	OFP	170 Packet Out
192.168.208.144	192.168.208.143	OFP	146 Flow Mod (
b2:3b:69:b5:58:a7	ba:ea:36:38:13:eb	OFP+ARP	126 Packet In
192.168.208.144	192.168.208.143	OFP	90 Packet Out
10.0.0.1	10.0.0.2	OFP+ICMP	182 Packet In
192.168.208.144	192.168.208.143	OFP	170 Packet Out
192.168.208.144	192.168.208.143	OFP	146 Flow Mod (

协议流程分析


Controller
Host1
IP:10.0.0.1
SW1
SW2
Host2
IP:10.0.0.2
ARP Request
10.0.0.1请求网关
10.0.0.254 MAC地址
OF+ARP
OF FlowModify
指示SW安装目的地址为
10.0.0.1的流表
OF+ARP
10.0.0.254请求10.0.0.2 MAC地址
（控制器逐一要求SW朝每个该网段对应的端口发送）
ARP Request
ARP Request
ARP Response
10.0.0.2的MAC
OF+ARP
10.0.0.2的MAC
OF+ARP
10.0.0.2的MAC
ARP Response
10.0.0.2的MAC
OF FlowModify
指示SW安装目的地址为
10.0.0.2的流表
ICMP PING Request
ARP Request
10.0.0.2请求
10.0.0.1 MAC地址
OF+ARP
OF+ARP
10.0.0.1的MAC
ARP Reponse
10.0.0.1的MAC
ICMP PING Response
后续其它流

如何获取

获取

获取Mininet的方法

1 镜像文件

官方网站已给我们制作好镜像文件，可以直接下载
http://mininet.org/download/

2 自行安装

如果对mininet的安装感兴趣，可以下载源代码自行编译安装mininet。详见http://mininet.org/download/

谢谢观赏

设计制作：冀烨

Mininet

优速网络　SDN项目组