

基于Floodlight的SDN应用

Getting Started

注意：这实际上是一个Floodlight的fork项目。

- 如何安装和运行Floodlight，请参看：[Floodlight - Installation Guide](#) 你需要做的是把git repo url 改成

```
https://github.com/xiaochengzhong/floodlightUI
```

- 关于如何用Mininet + Floodlight 搭建测试环境请参见：<http://liwenjunmm.diandian.com/post/2013-01-16/40047359629>

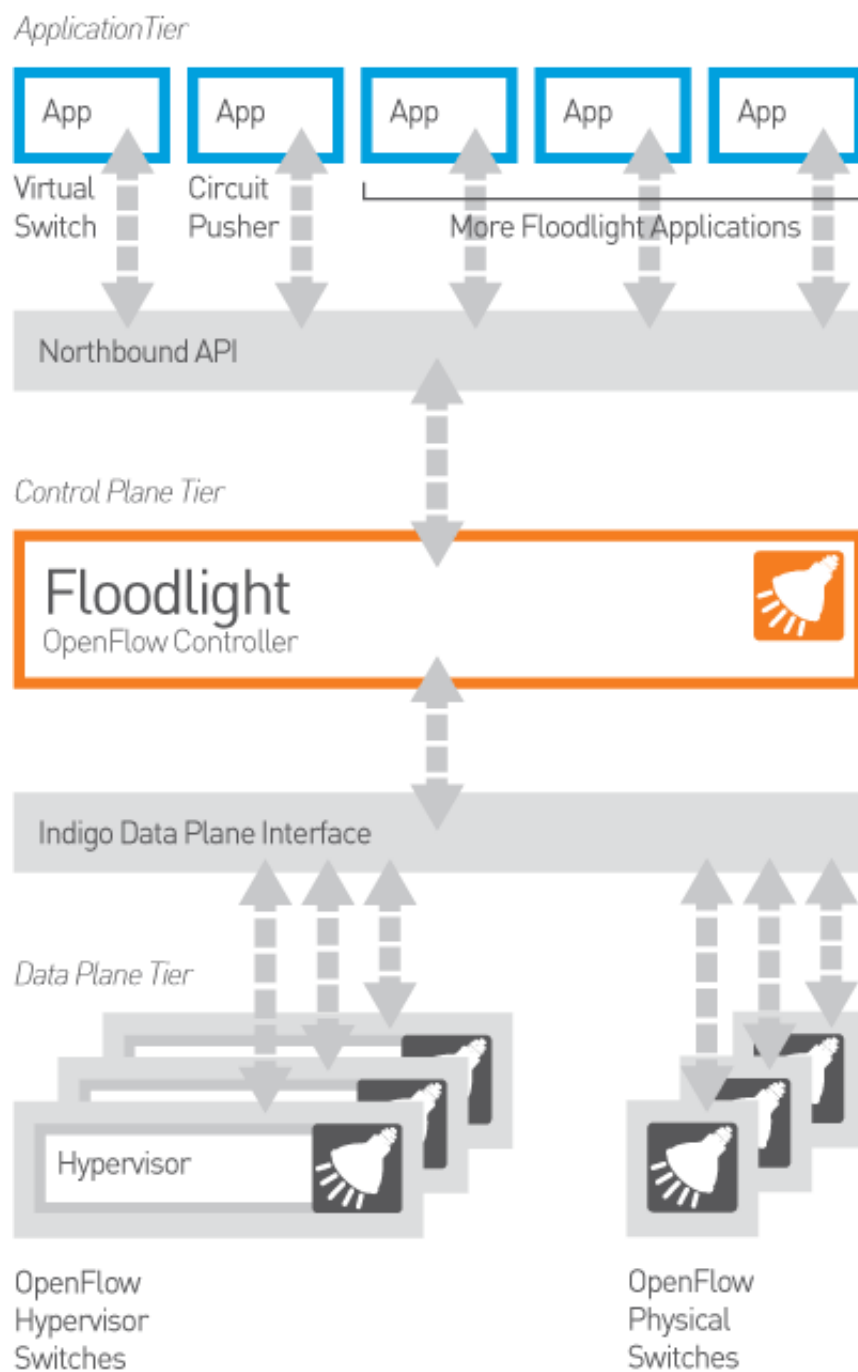
一、概述

1. Floodlight

[Floodlight](#)也即Floodlight Open SDN Controller是一个企业级的，使用Apache协议的，基于Java的OpenFlow控制器。它由[Big Switch Networks]社区的众多开发者维护。

OpenFlow是一个由Open Networking Foundation (ONF)管理的开放标准。它定义了一种协议让远程控制器通过路由器可以修改网络设备的行为，使用定义良好的转发指令集。Floodlight被设计为同正在增长的交换机，路由器，虚拟交换机，通过支持OpenFlow标准的设备一起工作。

一般的体系机构：



Floodlight的一些重要特性：

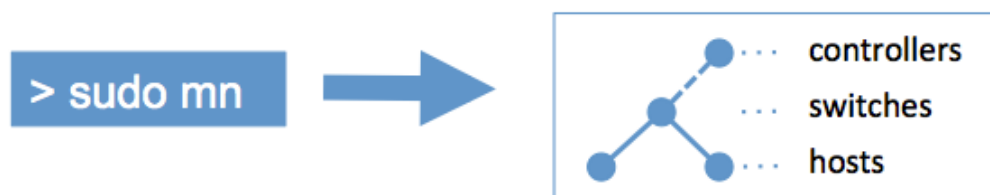
- 提供一个模块加载系统，使得拓展和提升变得简单。
- 容易构建——最小依赖。
- 支持多种虚拟的或物理的OpenFlow交换机。

- 能够处理OpenFlow和非OpenFlow的混合使用情形。它能管理多个OpenFlow硬件「岛屿」。
- 为高性能而设计。
- 支持OpenStack cloud orchestration platform

2. Mininet

Mininet是由一些虚拟的终端节点（end-hosts）、交换机、路由器连接而成的一个网络仿真器，它采用轻量级的虚拟化技术使得系统可以和真实网络相媲美。

Mininet可以很方便地创建一个支持SDN的网络：host就像真实的电脑一样工作，可以使用ssh登录，启动应用程序，程序可以向以太网端口发送数据包，数据包会被交换机、路由器接收并处理。有了这个网络，就可以灵活地为网络添加新的功能并进行相关测试，然后轻松部署到真实的硬件环境中。



Mininet的特性

- 可以简单、迅速地创建一个支持用户自定义的网络拓扑，缩短开发测试周期
- 可以运行真实的程序，在Linux上运行的程序基本上都可以在Mininet上运行，如Wireshark
- Mininet支持Openflow，在Mininet上运行的代码可以轻松移植到支持OpenFlow的硬件设备上
- Mininet可以在自己的电脑，或服务器，或虚拟机，或者云（例如Amazon EC2）上运行
- Mininet提供python API，简单易用
- Mininet是开源项目，源代码在：<https://github.com/mininet>

简单的使用方法

我们在VirtualBox安装Mininet虚拟机，虚拟机的镜像下载：<http://floodlight-download.projectfloodlight.org/files/floodlight-vm.zip>

利用它可以非常方便的模拟出一个N叉树的网络，其中每一个中间节点都是一个Switch，每一个叶节点都是一个Host。例如，模拟出一棵四层的满二叉树网络的命令为：

```
sudo mn --topo tree,depth=3,fanout=2 --controller=remote --ip
```

更多Mininet操作指导可以参见<http://mininet.org/walkthrough/>。

Mininet可以连接到Floodlight作为Controller，Controller可以是本地的或者远程的。默认情况下一个Mininet虚拟网络中的Switch只能使用同一个Controller，但是可以通过编写自定义脚本的方式来构建Mininet拓扑，并且在其中显示的指定多个Controller。

3. 应用概述

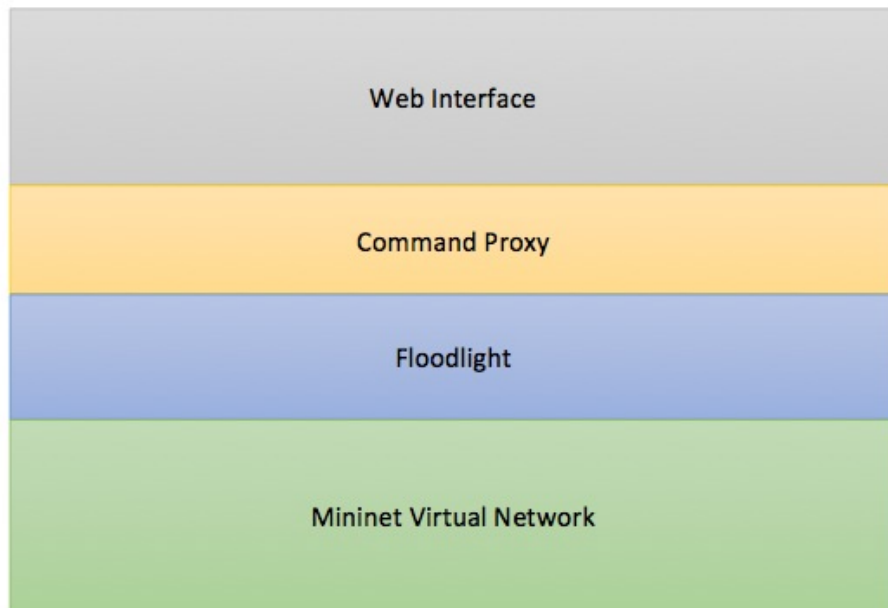
我们希望基于Floodlight现有的[API](#) (REST API, Static Flow Pusher API) 实现一个简单的管理站点。它的主要功能有：

- 查看Switch列表及各个Switch的信息
- 查看Host列表及各个Host的信息
- 查看网络拓扑图
- 查看流表
- 针对某个Switch添加/删除Static Flow
- 查看日志

二、应用设计

1. 体系结构

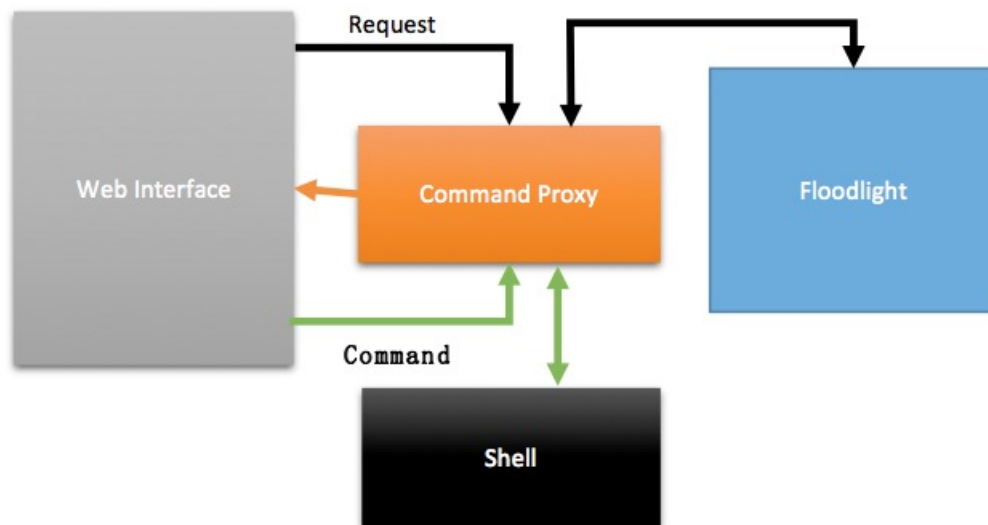
根据应用的需求和相关技术特性，我们对程序做如下体系结构设计：



我们对上图做简单的说明：

- **Mininet**虚拟网络。 底层虚拟机上的操作系统上安装有Mininet。通过Mininet我们可以创建多种类型的拓扑，并且可以把Floodlight Controller设置为远程，在我们的方案中Floodlight安装在虚拟机的宿主机上。
- **Floodlight** 负责监听和控制虚拟机中拓扑的动态。并开放REST API，执行相应的任务。
- **Proxy** 我们在宿主机上又搭建了一台服务器，后端即Command Proxy负责转发前端Web Interface的操作请求给Floodlight。
- **Web Interface** 为整个应用的前端，即主要的用户界面。负责内容展示和交互。根据用户操作生成相应的请求，发送到代理服务器。从服务器拿到结果后，对内容进行渲染展示给用户。这些页面上包括拓扑展示和操作，设备信息展示，流表展示和操作，日志显示等。

具体的模块间协作图：



为什么引入**Command Proxy**?

由于ajax不能跨域请求，所以我们建立了一个服务器端代理。Command Proxy是我们用PHP编写的一个后端脚本，它用来转发Web Interface的请求或者执行Web Interface发送的命令。

2. 构建

TODO:

三、关键功能简介

1. 查看**Switch**列表及各个**Switch**的信息

Switches (7)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
00:00:00:00:00:00:09	/192.168.69.44:48936	big switch networks	44	2716	1	2013年11月20日 下午4:01:40
00:00:00:00:00:00:0a	/192.168.69.44:48937	big switch networks	0	0	0	2013年11月20日 下午4:01:41
00:00:00:00:00:00:0b	/192.168.69.44:48938	big switch networks	0	0	0	2013年11月20日 下午4:01:41
00:00:00:00:00:00:0c	/192.168.69.44:48939	big switch networks	0	0	0	2013年11月20日 下午4:01:41
00:00:00:00:00:00:0d	/192.168.69.44:48940	big switch networks	0	0	0	2013年11月20日 下午4:01:42
00:00:00:00:00:00:0e	/192.168.69.44:48941	big switch networks	0	0	0	2013年11月20日 下午4:01:42
00:00:00:00:00:00:0f	/192.168.69.44:48935	big switch networks	0	0	0	2013年11月20日 下午4:01:40

我们首先为用户提供了可视化界面来查看路由的列表，所有路由的IP Adress、Vender、Packets、Bytes、Flows以及Connected Since信息一览无遗。

2. 查看Host列表及各个Host的信息

Hosts (8)

MAC Address	IP Address	Switch Port	Last Seen
f2:1a:65:11:d9:6c		00:00:00:00:00:00:0e-1	2013年11月20日 下午4:01:47
b2:3b:23:59:13:f5		00:00:00:00:00:00:0c-2	2013年11月20日 下午4:01:46
ee:a6:d4:f8:afe1		00:00:00:00:00:00:0e-2	2013年11月20日 下午4:01:47
2e:12:8f:24:f8:03		00:00:00:00:00:00:0f-2	2013年11月20日 下午4:01:48
52:02:72:57:bf:4e		00:00:00:00:00:00:0b-1	2013年11月20日 下午4:01:45
56:91:2c:8d:1a:5a		00:00:00:00:00:00:0f-1	2013年11月20日 下午4:01:48
9e:4d:3c:75:69:29		00:00:00:00:00:00:0b-2	2013年11月20日 下午4:01:45

同时我们继续利用Floodlight的自带API让各主机的信息也展现在了用户面前。

3. 查看网络拓扑图

Network Topology



4. 查看流表

/192.168.69.44:48936

Connected since 2013年11月20日 下午4:01:40
big switch networks
Open vSwitch
1.1.2
SN: None

Ports (3)

#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
2 (s9-eth2)	UP 10 Gbps FDX	17078	9175	275	146	0	0
65534 (dp0)	DOWN	1620	0	22	0	0	0
1 (s9-eth1)	UP 10 Gbps FDX	9880	9900	157	157	0	0

Flows (1)

Name	Cookie	Priority	Match	Action	Packets	Bytes	Age	Timeout
flow-mod-2	45035996273704960	32763	port=1	output 2	94	5798	1567 s	0 s

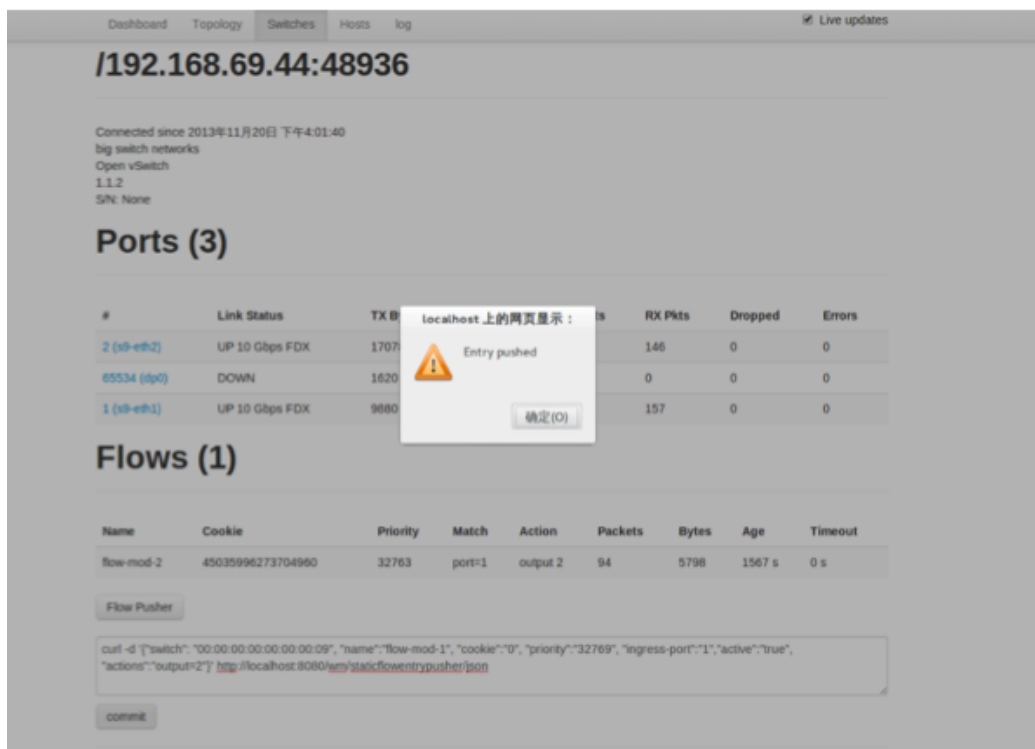
Flow Pusher

curl -d '{"switch": "00:00:00:00:00:00:00:09", "name": "flow-mod-1", "cookie": "0", "priority": "32769", "ingress-port": "1", "active": "true", "actions": {"output": "2"}, "http://localhost:8080/vm/staticflowentrypusher/json

commit

我们还提供了查看流表的功能。用户可以看到流的名称、Cookie、优先级、数据包数量等信息。

5. 针对某个Switch添加/删除Static Flow



应用给用户提供了增加和删除流表的功能。在流表的下方我们为用户提供了命令输入窗口，在这里用户可以自主输入添加流表命令，比如示例命令如下：

```
curl -d '{"switch": "00:00:00:00:00:00:00:09", "name": "flow-m
```

来添加一条流表（具体命令格式详见Floodlight REST API说明）。上图是在添加流表，下图则是添加流表成功的样子。我们可以明显看到流表中多了一条记录。

Switch 00:00:00:00:00:00:09 /192.168.69.44:48936

Connected since 2013年11月20日 下午4:01:40
big switch networks
Open vSwitch
1.1.2
S/N: None

Ports (3)

#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
2 (s9-eth2)	UP 10 Gbps FDX	18001	9602	290	153	0	0
65534 (dp0)	DOWN	1620	0	22	0	0	0
1 (s9-eth1)	UP 10 Gbps FDX	10376	10396	165	165	0	0

Flows (2)

Name	Cookie	Priority	Match	Action	Packets	Bytes	Age	Timeout
flow-mod-1	45035996273704960	-32767	port=1	output 2	4	252	36 s	0 s
flow-mod-2	45035996273704960	32763	port=1	output 2	98	6042	1669 s	0 s

Flow Pusher

删除和添加基本上一样，区别仅在于命令不同。这是删除上面添加的流表的示例命令：

```
curl -X DELETE -d '{"name":"flow-mod-1"}' http://localhost:80
```

Ports (3)

#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
2 (s9-eth2)	UP 10 Gbps FDX	18627	9846	300	153	0	0
65534 (dp0)	DOWN	1620	0	22	0	0	0
1 (s9-eth1)	UP 10 Gbps FDX	10689	10709	170	165	0	0

Flows (2)


Name	Cookie	Priority	Match	Action	Packets	Bytes	Age	Timeout
flow-mod-1	45035996273704960	-32767	port=1	output 2	4	252	36 s	0 s
flow-mod-2	45035996273704960	32763	port=1	output 2	98	6042	1669 s	0 s

Flow Pusher

```
curl -X DELETE -d '{"name":"flow-mod-1"}' http://localhost:8080/wm/staticflowentrypusher/json
```

commit

localhost 上的网页显示：

 Entry flow-mod-1 deleted

确定(O)

Ports (3)

#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
2 (s9-eth2)	UP 10 Gbps FDX	18627	9846	300	157	0	0
65534 (dp0)	DOWN	1620	0	22	0	0	0
1 (s9-eth1)	UP 10 Gbps FDX	10689	10709	170	170	0	0

Flows (1)

Name	Cookie	Priority	Match	Action	Packets	Bytes	Age	Timeout
flow-mod-2	45035996273704960	32763	port=1	output 2	101	6233	1724 s	0 s

Flow Pusher

对命令中的一些参数的说明

Possible properties of a flow entry:

Key	Value	Notes
switch	<switch ID>	ID of the switch (data path) that this rule should be added to xx:xx:xx:xx:xx:xx:xx:xx
		Name of the flow entry, this is the primary

name	<string>	key, it MUST be unique
actions	<key>= <value>	See table of actions below Specify multiple actions using a comma-separated list Specifying no actions will cause the packets to be dropped
priority	<number>	default is 32767 maximum value is 32767
active	<boolean>	
wildcards		
ingress-port	<number>	switch port on which the packet is received Can be hexadecimal (with leading 0x) or decimal
src-mac	<mac address>	xx:xx:xx:xx:xx:xx
dst-mac	<mac address>	xx:xx:xx:xx:xx:xx
vlan-id	<number>	Can be hexadecimal (with leading 0x) or decimal
vlan-priority	<number>	Can be hexadecimal (with leading 0x) or decimal
ether-type	<number>	Can be hexadecimal (with leading 0x) or decimal
tos-bits	<number>	Can be hexadecimal (with leading 0x) or decimal
protocol	<number>	Can be hexadecimal (with leading 0x) or decimal
src-ip	<ip address>	xx.xx.xx.xx[/xx]

	[/mask]	
dst-ip	<ip address> [/mask]	xx.xx.xx.xx[/xx]
src-port	<number>	Can be hexadecimal (with leading 0x) or decimal
dst-port	<number>	Can be hexadecimal (with leading 0x) or decimal

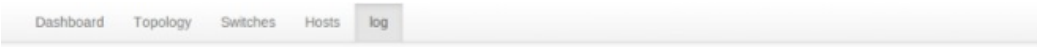
Possible actions within the "action" field:

Key	Value	Notes
output	<number> all controller local ingress-port normal flood	no "drop" option (instead, specify no action to drop packets)
enqueue	<number>: <number>	First number is port number, second is queue ID Can be hexadecimal (with leading 0x) or decimal
strip-vlan		
set-vlan-id	<number>	Can be hexadecimal (with leading 0x) or decimal
set-vlan-priority	<number>	Can be hexadecimal (with leading 0x) or decimal
set-src-mac	<mac address>	xx:xx:xx:xx:xx:xx

set-dst-mac	<mac address>	xx:xx:xx:xx:xx:xx
set-tos-bits	<number>	
set-src-ip	<ip address>	xx.xx.xx.xx
set-dst-ip	<ip address>	xx.xx.xx.xx
set-src-port	<number>	Can be hexadecimal (with leading 0x) or decimal
set-dst-port	<number>	Can be hexadecimal (with leading 0x) or decimal

6. 查看日志

最后我们还完成了查看日志的功能。整个拓扑网络的日志监听完整地在这里反映出来了，并能够根据日志实时更新页面。页面的样子如下：



Log Info

Tail Stop

```
2013-11-20 16:32:31.138 WARN [n.f.t.TopologyManager] The following switch ports have multiple links incident on them, so these ports will be treated as broadcast domain ports. [[id=00:00:00:00:00:0d, port=3], [id=00:00:00:00:00:0a, port=3]]
2013-11-20 16:31:54.699 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0d outPort=3, dst=00:00:00:00:00:0a, inPort=3]
2013-11-20 16:31:54.697 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0a outPort=3, dst=00:00:00:00:00:09, inPort=1]
2013-11-20 16:28:46.97 WARN [n.f.t.TopologyManager] The following switch ports have multiple links incident on them, so these ports will be treated as broadcast domain ports. [[id=00:00:00:00:00:0d, port=3], [id=00:00:00:00:00:0a, port=3]]
2013-11-20 16:28:09.637 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0a outPort=3, dst=00:00:00:00:00:09, inPort=1]
2013-11-20 16:28:09.632 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0d outPort=3, dst=00:00:00:00:00:0a, inPort=3]
2013-11-20 16:26:46.74 WARN [n.f.t.TopologyManager] The following switch ports have multiple links incident on them, so these ports will be treated as broadcast domain ports. [[id=00:00:00:00:00:0d, port=3], [id=00:00:00:00:00:0a, port=3]]
2013-11-20 16:26:09.606 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0a outPort=3, dst=00:00:00:00:00:09, inPort=1]
2013-11-20 16:26:09.604 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0d outPort=3, dst=00:00:00:00:00:0a, inPort=3]
2013-11-20 16:25:01.54 WARN [n.f.t.TopologyManager] The following switch ports have multiple links incident on them, so these ports will be treated as broadcast domain ports. [[id=00:00:00:00:00:0d, port=3], [id=00:00:00:00:00:0a, port=3]]
2013-11-20 16:24:24.619 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0a outPort=3, dst=00:00:00:00:00:09, inPort=1]
2013-11-20 16:24:24.589 WARN [n.f.i.l.s.notification] Link added: Link [src=00:00:00:00:00:0d outPort=3, dst=00:00:00:00:00:0a, inPort=3]
2013-11-20 16:21:09.36 DEBUG [n.f.s.w.ListStaticFlowEntriesResource] Listing all static flow entries for switch: 00:00:00:00:00:09
2013-11-20 16:21:01.9 WARN [n.f.t.TopologyManager] The following switch ports have multiple links incident on them, so these ports will be treated as broadcast domain ports. [[id=00:00:00:00:00:0d, port=3], [id=00:00:00:00:00:0a, port=3]]
```

7. 通信

以下所有命令的具体信息请参看：[Mininet Walkthrough](#)

查看节点中连接信息，在mininet控制台输入以下命令：

```
mininet> net
```

结果如下：

```
mininet> net
s9 <-> s10-eth3 s13-eth3
s10 <-> s11-eth3 s12-eth3 s9-eth1
s11 <-> h1-eth0 h2-eth0 s10-eth1
s12 <-> h3-eth0 h4-eth0 s10-eth2
s13 <-> s14-eth3 s15-eth3 s9-eth2
s14 <-> h5-eth0 h6-eth0 s13-eth1
s15 <-> h7-eth0 h8-eth0 s13-eth2
```

查看所有节点的IP端口信息，在mininet控制台输入以下命令：

```
mininet> dump
```

结果如下：

```
mininet> dump
c0-0: IP=192.168.69.44 intfs= pid=5117
s9: IP=None intfs=s9-eth1,s9-eth2 pid=5135
s10: IP=None intfs=s10-eth1,s10-eth2,s10-eth3 pid=5137
s11: IP=None intfs=s11-eth1,s11-eth2,s11-eth3 pid=5139
s12: IP=None intfs=s12-eth1,s12-eth2,s12-eth3 pid=5141
s13: IP=None intfs=s13-eth1,s13-eth2,s13-eth3 pid=5143
s14: IP=None intfs=s14-eth1,s14-eth2,s14-eth3 pid=5145
s15: IP=None intfs=s15-eth1,s15-eth2,s15-eth3 pid=5147
h1: IP=10.0.0.1 intfs=h1-eth0 pid=5119
h2: IP=10.0.0.2 intfs=h2-eth0 pid=5121
h3: IP=10.0.0.3 intfs=h3-eth0 pid=5123
h4: IP=10.0.0.4 intfs=h4-eth0 pid=5125
h5: IP=10.0.0.5 intfs=h5-eth0 pid=5127
h6: IP=10.0.0.6 intfs=h6-eth0 pid=5129
h7: IP=10.0.0.7 intfs=h7-eth0 pid=5131
h8: IP=10.0.0.8 intfs=h8-eth0 pid=5133
```

查看特定switch或host的端口信息，在mininet控制台输入以下命令：

```
mininet> s9 ifconfig -a //s9 is a switch name
```

结果如下：

```
s15-eth3  Link encap:Ethernet  HWaddr 6e:44:38:5a:89:6d
           inet6 addr: fe80::6c44:38ff:fe5a:896d/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:211 errors:0 dropped:2 overruns:0 frame:0
           TX packets:167 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:16602 (16.6 KB)  TX bytes:13169 (13.1 KB)

s9-eth1    Link encap:Ethernet  HWaddr 9e:e1:6b:74:87:aa
           inet6 addr: fe80::9ce1:6bff:fe74:87aa/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:210 errors:0 dropped:0 overruns:0 frame:0
           TX packets:218 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:16739 (16.7 KB)  TX bytes:17411 (17.4 KB)

s9-eth2    Link encap:Ethernet  HWaddr e6:ec:d4:8b:95:02
           inet6 addr: fe80::e4ec:d4ff:fe8b:9502/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:204 errors:0 dropped:0 overruns:0 frame:0
           TX packets:254 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:16224 (16.2 KB)  TX bytes:20065 (20.0 KB)
```

查看特定switch或host的arp或route信息，在mininet控制台输入以下命令：

```
mininet> s9 arp //s9 is a switch name
```

```
mininet> s9 route //s9 is a switch name
```

结果如下：

```
mininet> s9 arp
mnexec -p arp
Address                HWtype  HWaddress           Flags Mask            Iface
10.0.2.2                ether   52:54:00:12:35:02   C                     eth0
10.0.2.2                ether   52:54:00:12:35:02   C                     eth0
mininet> s9 route
mnexec -p route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.2.0       *               255.255.255.0   U        0      0      0 eth0
default        10.0.2.2        0.0.0.0         UG       100    0      0 eth0
```

查看两个节点间的连接信息，在mininet控制台输入以下命令：


```
mininet> h1 ping -c 1 h2 //h1 , h2 is a host name
```

结果如下：

```
mininet> h1 ping -c 1 h2
mnexec -p ping -c 1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=8.37 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 8.376/8.376/8.376/0.000 ms
```

查看所有节点间的连接信息，在mininet控制台输入以下命令：

```
mininet> pingall
```

结果如下：

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (0/56 lost)
```

四、组员及分工

- 钟晓诚MF1332095，靳峥MF1332025：Static Flow Pusher API实现。
- 吕翔MF1332042，唐毅明MF1332057：编写测试用的发包脚本，项目文档编写。
- 孟焱MF1332044，倪卫明MF1332045：拓扑设计与环境搭建，Web页面UI设计，日志功能实现。