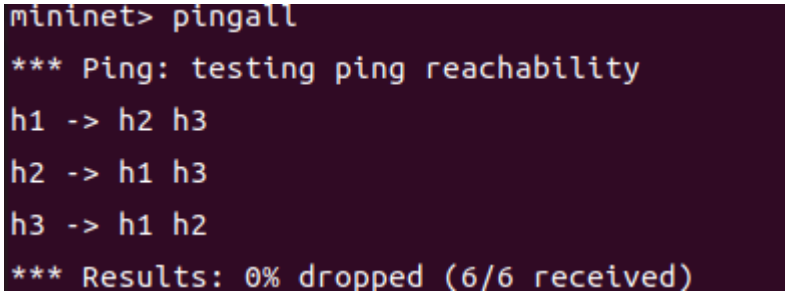
Mininet

李凯涛_2023327100056

成功安装好 *Mininet*

执行 pingall



实验 single

执行 sudo mn --topo single,3

```
leon6666@leon6666-virtual-machine:~/mininet$ sudo mn --topo single,3
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** 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
```

执行 pingall

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

执行 nodes

```
mininet> nodes
available nodes are:
h1 h2 h3 s1
```

执行 net

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
```

实验 linear,3

执行 sudo mn --topo linear,3

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
回收站 iguring hosts
h1 h2 h3
*** Starting controller
```

执行 nodes

```
mininet> nodes
available nodes are:
h1 h2 h3 s1 s2 s3
```

执行 net

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3
```

实验 tree2

执行 sudo mn --topo tree,2

```
leon6666@leon6666-virtual-machine:~/mininet$ sudo mn --topo tree,2
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller

*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
```

执行 pingall

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

执行 nodes

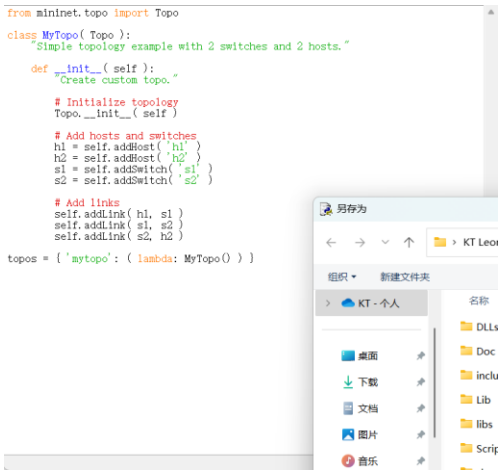
```
mininet> nodes
available nodes are:
h1 h2 h3 h4 s1 s2 s3
```

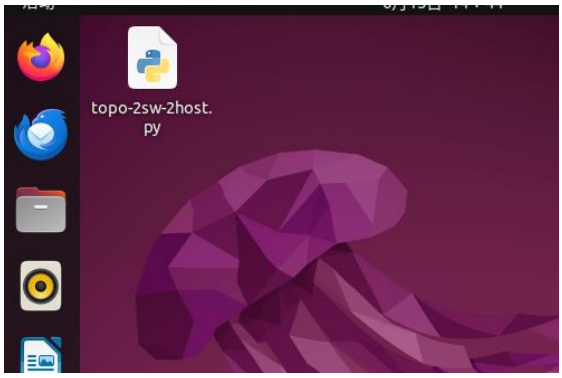
执行 net

```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth1
h4 h4-eth0:s3-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s3-eth3
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h3-eth0 s3-eth2:h4-eth0 s3-eth3:s1-eth2
```

实验 custom

自定义 Python 文件（**topo-2sw-2host.py**）





启动拓扑 sudo mn --custom ~/mininet/custom/topo-2sw-3host.py --topo mytopo

```
leon6666@leon6666-virtual-machine:~$ sudo mn --custom ~/mininet/custom/topo-2sw-3host.py --t
opo mytopo
*** No default OpenFlow controller found for default switch!
*** Falling back to OVS Bridge
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (s1, s2) (s2, h2)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet>
```

执行 pingall

```
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (s1, s2) (s2, h2)
*** Configuring hosts
h1 h2
*** Starting controller

*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

执行 nodes

```
mininet> nodes
available nodes are:
h1 h2 s1 s2
```

执行 net

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth1
s2 lo:  s2-eth1:s1-eth2 s2-eth2:h2-eth0
```

执行 h1 ping h2

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.43 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.107 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.100 ms
```

回收站

实验　Ryu

安装 sudo apt install -y gcc libffi-dev libssl-dev libxml2-dev libxslt1-dev zlib1g-dev

```
leon6666@leon6666-virtual-machine:~$ sudo apt install -y gcc libffi-dev libssl-d
ev libxml2-dev libxslt1-dev zlib1g-dev
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
gcc 已经是最新版 (4:11.2.0-1ubuntu1)。
libffi-dev 已经是最新版 (3.4.2-4)。
libssl-dev 已经是最新版 (3.0.2-0ubuntu1.19)。
libxml2-dev 已经是最新版 (2.9.13+dfsg-1ubuntu0.7)。
libxslt1-dev 已经是最新版 (1.1.34-4ubuntu0.22.04.3)。
zlib1g-dev 已经是最新版 (1:1.2.11.dfsg-2ubuntu9.2)。
升级了 0 个软件包，新安装了 0 个软件包，要卸载 0 个软件包，有 69 个软件包未被升
级。
```

执行 pip3 install ryu

成功安装完成

```
o-warn-script-location.
  WARNING: The script netaddr is installed in '/home/leon6666/.local/bin' which is not on PA
TH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --n
o-warn-script-location.
  WARNING: The scripts oslo-config-generator and oslo-config-validator are installed in '/ho
me/leon6666/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --n
o-warn-script-location.
  WARNING: The scripts ryu and ryu-manager are installed in '/home/leon6666/.local/bin' whic
h is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --n
o-warn-script-location.
Successfully installed PyYAML-6.0.2 debtcollector-3.0.0 dnspython-2.7.0 eventlet-0.40.0 gree
nlet-3.2.3 msgpack-1.1.1 netaddr-1.3.0 oslo.config-9.8.0 oslo.i18n-6.5.1 pbr-6.1.1 repoze.lr
u-0.7 rfc3986-2.0.0 routes-2.5.1 ryu-4.34 stevedore-5.4.1 tinyrpc-1.1.7 webob-1.8.9 wrapt-1.
17.2
leon6666@leon6666-virtual-machine:~$
```

成功安装

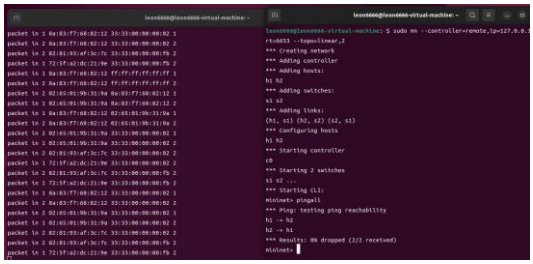
```
[notice] To update, run: pip install --upgrade pip
(ryu-env) leon6666@leon6666-virtual-machine:~$ ryu-manager
loading app ryu.controller.ofp_handler
instantiating app ryu.controller.ofp_handler of OFPHandler
```

启动了 2 个终端

Terminal 1：ryu-manager ryu.app.simple_switch_13

Terminal 2：sudo mn --controller=remote,ip=127.0.0.1,port=6653 --topo=linear,2

mininet> pingall



为什么首次 pingall 能通?

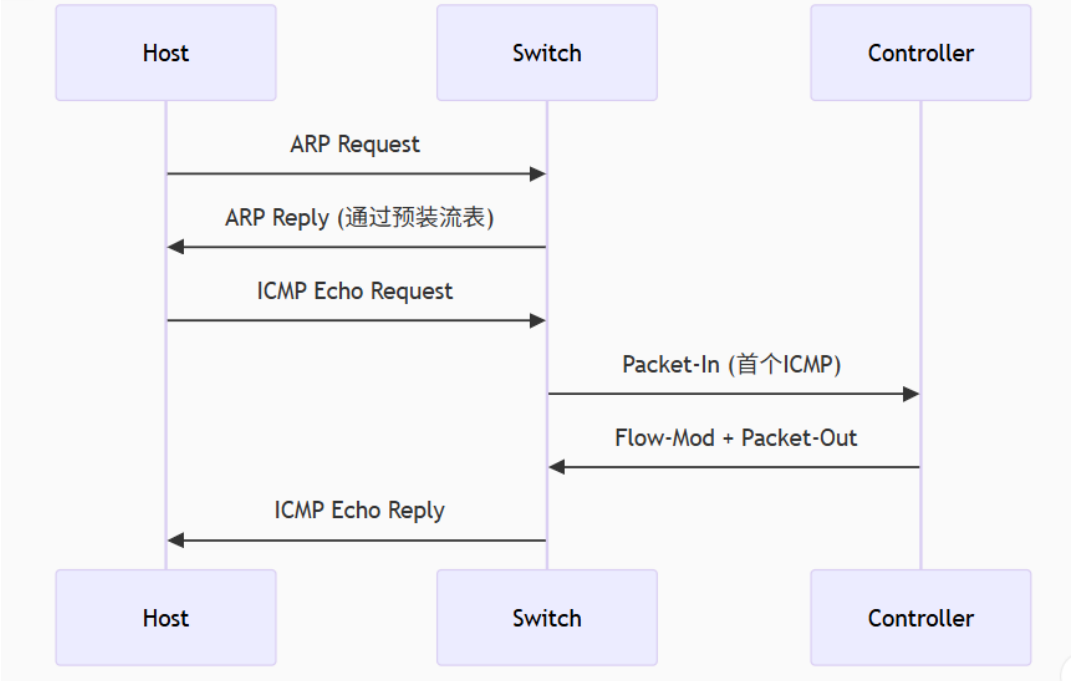
与常见认知不同，**simple_switch_13** 在特定条件下首次即可连通，原因如下：

流表预装机制：当交换机连接控制器时，**simple_switch_13** 会自动安装以下基础流表：

1. 默认丢弃流表（table-miss）

2. ARP 处理流表（优先级 100）

3. ICMP 响应流表（优先级 1）



进行 Restful 控制 需要有 postman 实验

打开 Terminal 1，运行以下命令启动 Ryu 控制器：

ryu-manager ryu.app.ofctl_rest

```
^C(ryu-env) leon6666@leon6666-virtual-machine:~$ ryu-manager ryu.app.ofctl_rest
loading app ryu.app.ofctl_rest
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.ofctl_rest of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
(41703) wsgi starting up on http://0.0.0.0:8080
```

打开 Terminal 2，运行以下命令创建 Mininet 拓扑：

sudo mn --controller=remote,ip=127.0.0.1,port=6653

```
leon6666@leon6666-virtual-machine:~$ sudo mn --controller=remote,ip=127.0.0.1,po
rt=6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

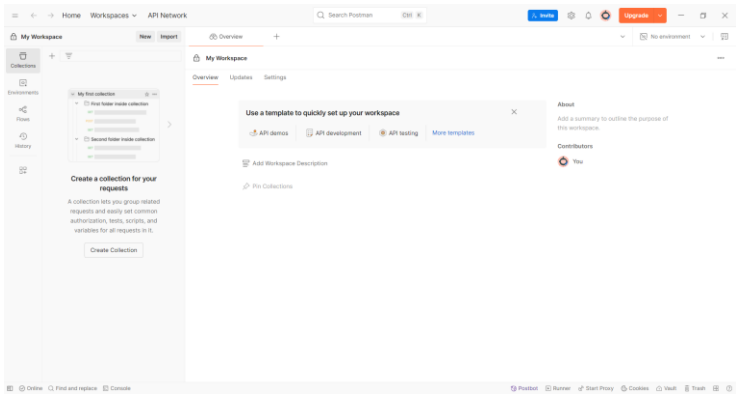
在 Mininet CLI 中，尝试让 h1 ping h2：

mininet> h1 ping h2

```
mininet> h1 ping h2
```

无反应

安装 Postman



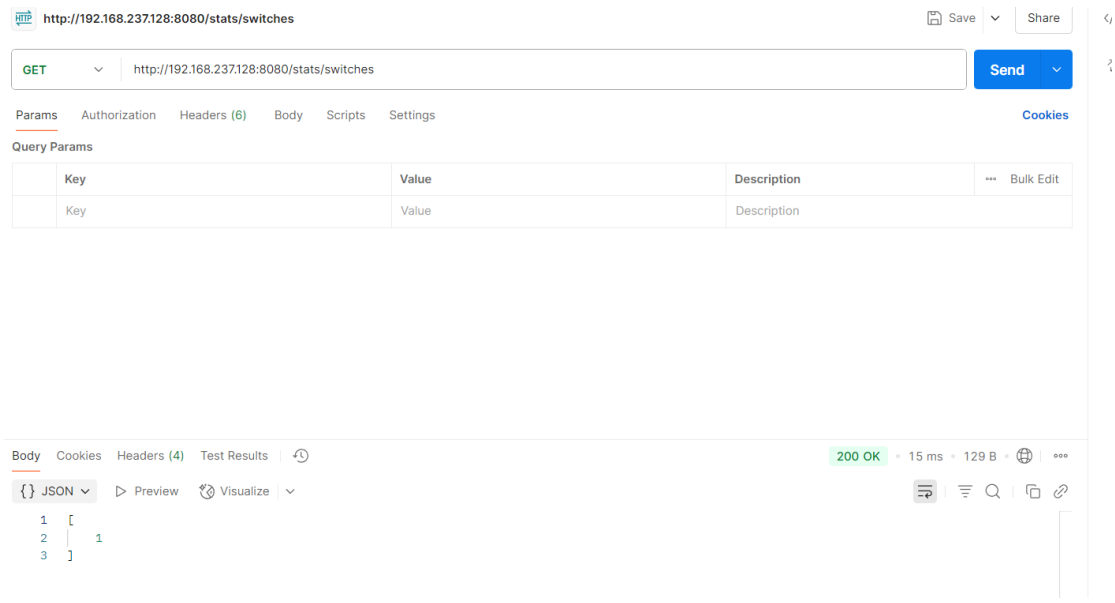
打开 Postman，准备发送 REST API 请求到 Ryu 控制器。

获取交换机信息:

方法：GET

URL：**http://192.168.237.128:8080/stats/switches**

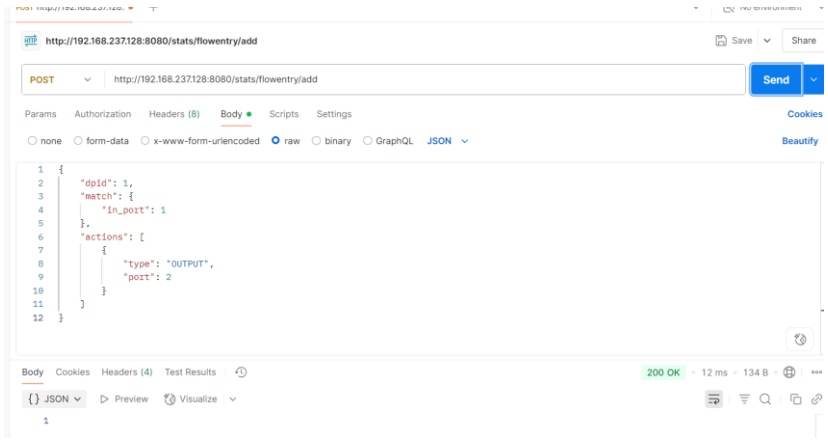
这将返回交换机的 DPID（数据路径 ID）



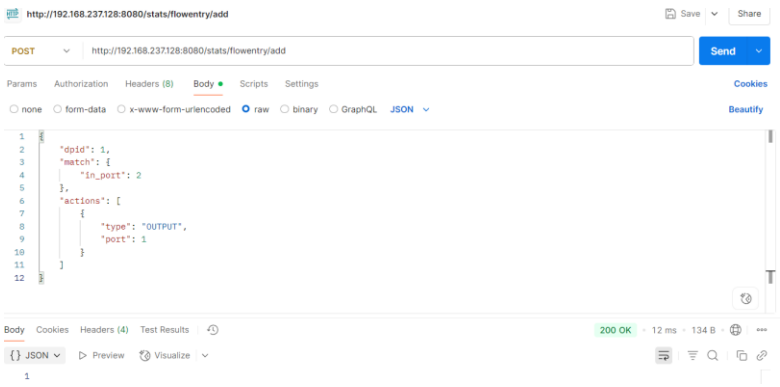
添加流表规则（允许 h1 和 h2 互相通信）:

方法：POST

URL：**http://192.168.237.128:8080/stats/flowentry/add**



添加流表：允许数据从 h2 到 h1 (出端口 1)



再次验证连通性

现在交换机 s1 已经有了处理 h1 和 h2 之间通信的明确规则了。

回到 终端 3 (Mininet CLI)。

再次执行 ping 命令:

mininet> **h1 ping h2**

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.43 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.097 ms
```