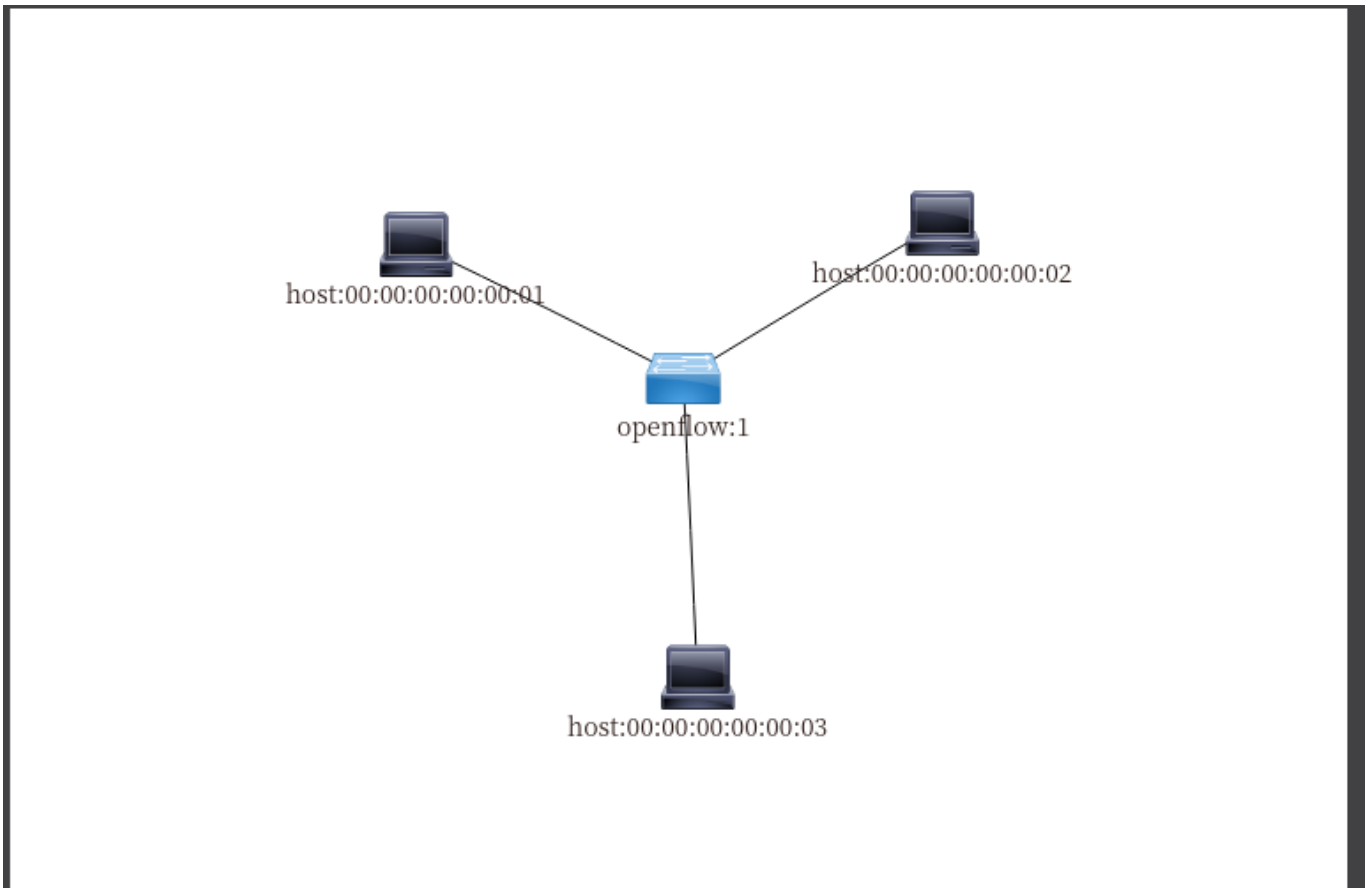
102101643_黄鹏伟_实验七

一、搭建拓扑链接ODL控制器



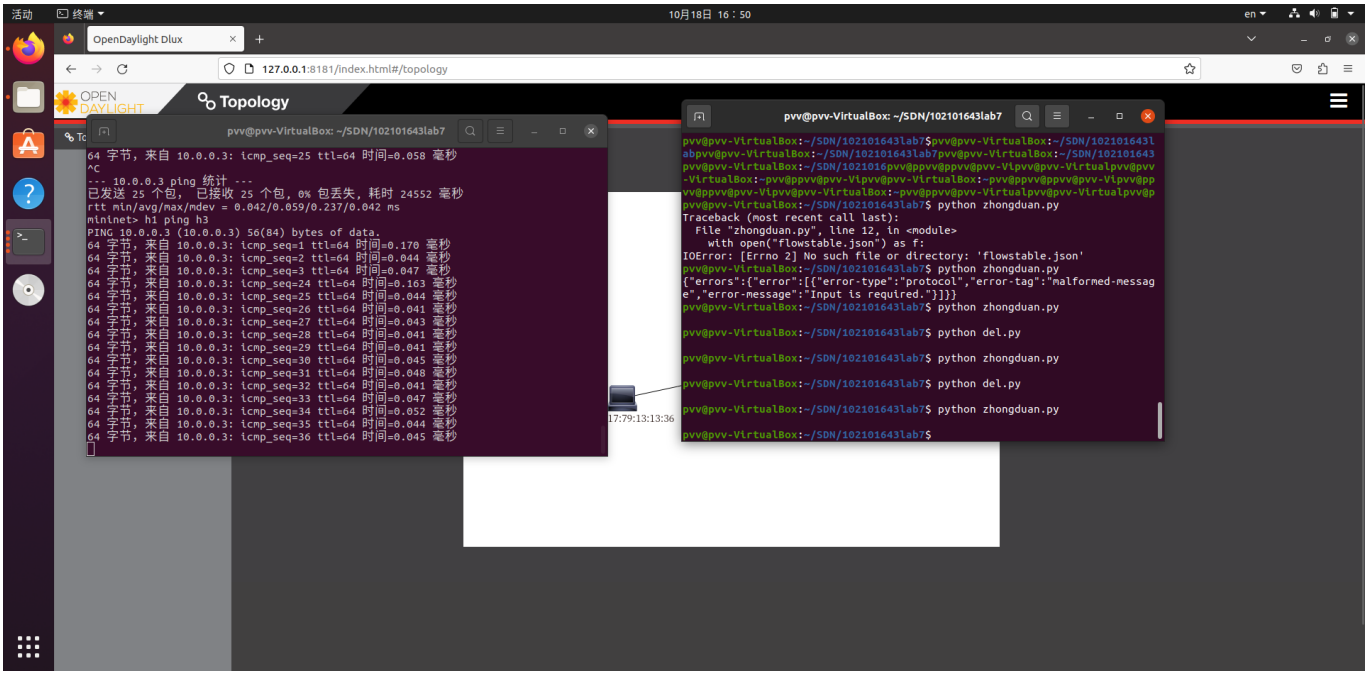
一、编程实现下发硬超时流表，实现拓扑内主机h1和h3网络中断20s

1.1 下发流表代码

```
#!/usr/bin/python
import requests
from requests.auth import HTTPBasicAuth
def http_put(url,jstr):
    url= url
    headers = {'Content-Type':'application/json'}
    resp = requests.put(url,jstr,headers=headers,auth=HTTPBasicAuth('admin',
'admin'))
    return resp

if __name__ == "__main__":
    url = 'http://127.0.0.1:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:1/flow-node-inventory:table/0/flow/1'
    with open("flowstable.json") as f:
        jstr = f.read()
```

```
    resp = http_put(url,jstr)
    print (resp.content)
```

1.2 执行结果



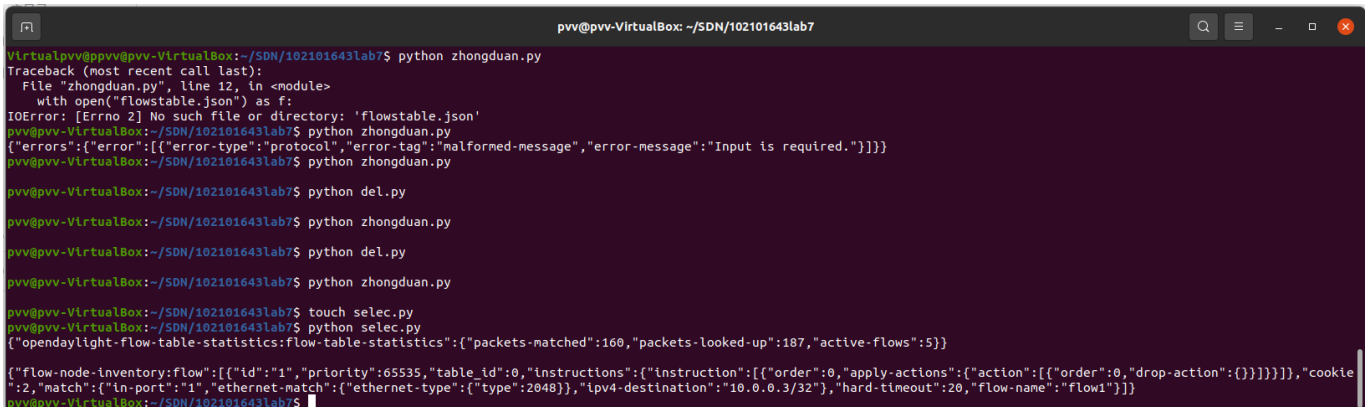
二、编程实现获取s1上实时的流表数，并通过添加和删除流表项实现更新当前交换机的实时流表数。（以硬超时为例）

2.1 获取活动流表代码

```
#!/usr/bin/python
import requests
from requests.auth import HTTPBasicAuth
def http_get(url):
    url= url
    headers = {'Content-Type':'application/json'}
    resp = requests.get(url,headers=headers,auth=HTTPBasicAuth('admin', 'admin'))
    print (resp.content)
    url = 'http://127.0.0.1:8181/restconf/config/opendaylight-
inventory:nodes/node/openflow:1/flow-node-inventory:table/0/flow/1'
    resp = requests.put(url,jstr,headers=headers,auth=HTTPBasicAuth('admin',
'admin'))
    print (resp.content)
    url= url
    headers = {'Content-Type':'application/json'}
    resp = requests.get(url,headers=headers,auth=HTTPBasicAuth('admin', 'admin'))
    print (resp.content)
    return resp
```

```
if __name__ == "__main__":
    url = 'http://127.0.0.1:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:1/flow-node-inventory:table/0/opendaylight-flow-table-statistics:flow-table-statistics'

    with open("flowstable.json") as f:
        jstr = f.read()
    resp = http_get(url)
```

2.2 执行结果截图

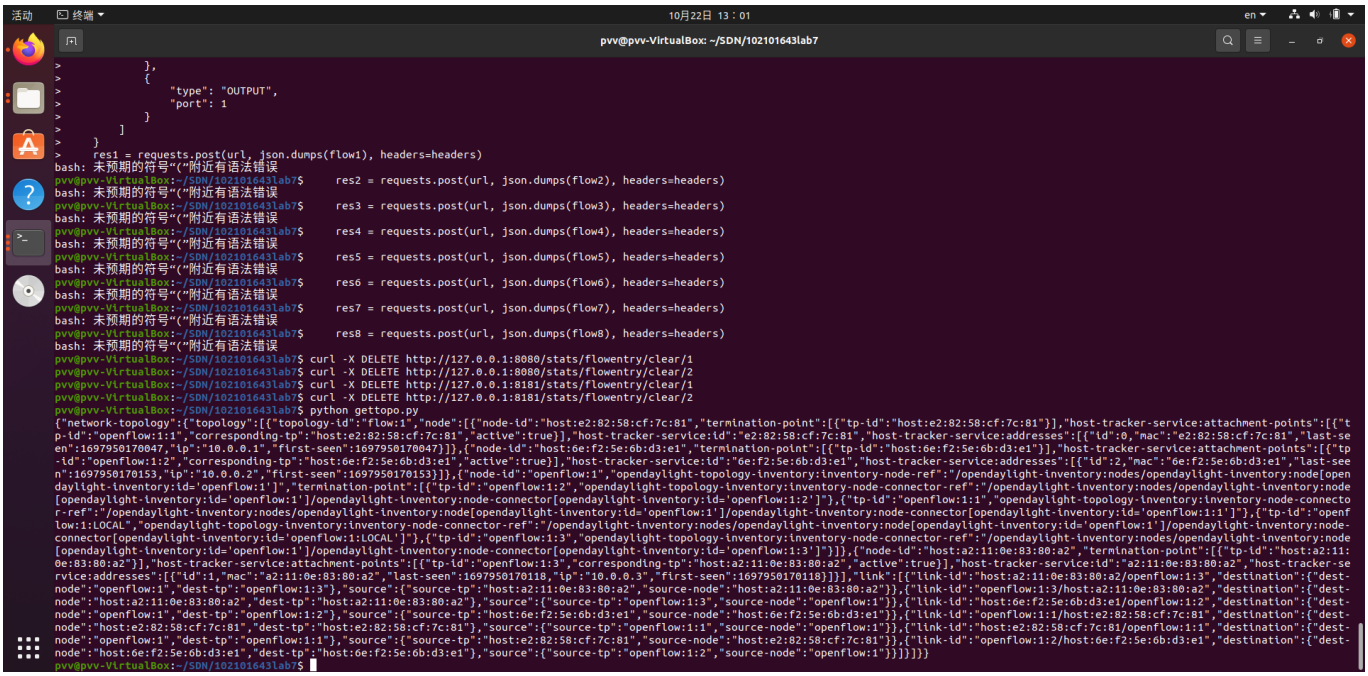


三、编程实现获取拓扑信息，包括主机、交换机和链路。

3.1 获取拓扑信息代码

```
#!/usr/bin/python
import requests
from requests.auth import HTTPBasicAuth
def http_get(url):
    url= url
    headers = {'Content-Type':'application/json'}
    resp = requests.get(url,headers=headers,auth=HTTPBasicAuth('admin', 'admin'))
    return resp

if __name__ == "__main__":
    url = 'http://127.0.0.1:8181/restconf/operational/network-topology:network-topology'
    resp = http_get(url)
    print (resp.content)
```

3.2 执行结果截图



四、编程复现基于前序VLAN实验

4.1 实现相同的VLAN配置

4.1.1 搭建拓扑

```
#!/usr/bin/env python
from mininet.topo import Topo

class MyTopo(Topo):
    def __init__(self):
        # initilaize topology
        Topo.__init__(self)

        self.addSwitch("s1")
        self.addSwitch("s2")

        self.addHost("h1")
        self.addHost("h2")
        self.addHost("h3")
        self.addHost("h4")

        self.addLink("s1", "h1")
        self.addLink("s1", "h2")
        self.addLink("s2", "h3")
        self.addLink("s2", "h4")
        self.addLink("s1", "s2")

topos = {'mytopo': (lambda: MyTopo())}
```

4.1.2 实现相同的VLAN配置

4.1.2.1 下发流表

代码

```
# ryu_vlan.py
import json

import requests

if __name__ == "__main__":
    url = 'http://127.0.0.1:8080/stats/flowentry/add'
    headers = {'Content-Type': 'application/json'}
    flow1 = {
        "dpid": 1,
        "priority": 1,
        "match":{
            "in_port": 1
        },
        "actions":[
            {
                "type": "PUSH_VLAN",
                "ethertype": 33024
            },
            {
                "type": "SET_FIELD",
                "field": "vlan_vid",
                "value": 4096
            },
            {
                "type": "OUTPUT",
                "port": 3
            }
        ]
    }
    flow2 = {
        "dpid": 1,
        "priority": 1,
        "match":{
            "in_port": 2
        },
        "actions":[
            {
                "type": "PUSH_VLAN",
                "ethertype": 33024
            },
            {
                "type": "SET_FIELD",
                "field": "vlan_vid",
                "value": 4097
            },
            {
                "type": "OUTPUT",
```

```
                "port": 3
            }
        ]
    }
    flow3 = {
        "dpid": 1,
        "priority": 1,
        "match":{
            "vlan_vid": 0
        },
        "actions":[
            {
                "type": "POP_VLAN",
                "ethertype": 33024
            },
            {
                "type": "OUTPUT",
                "port": 1
            }
        ]
    }
    flow4 = {
        "dpid": 1,
        "priority": 1,
        "match": {
            "vlan_vid": 1
        },
        "actions": [
            {
                "type": "POP_VLAN",
                "ethertype": 33024
            },
            {
                "type": "OUTPUT",
                "port": 2
            }
        ]
    }
    flow5 = {
        "dpid": 2,
        "priority": 1,
        "match": {
            "in_port": 1
        },
        "actions": [
            {
                "type": "PUSH_VLAN",
                "ethertype": 33024
            },
            {
                "type": "SET_FIELD",
                "field": "vlan_vid",
                "value": 4096
            },
```

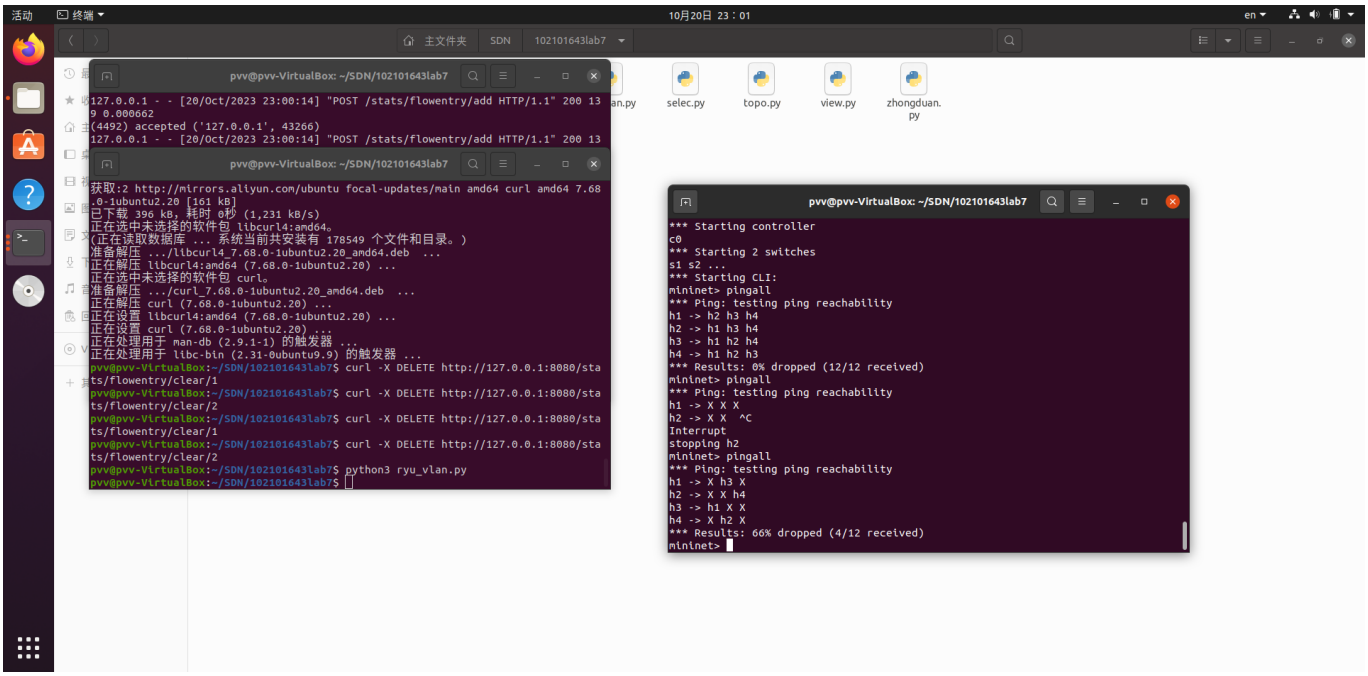
```
            {
                "type": "OUTPUT",
                "port": 3
            }
        ]
    }
    flow6 = {
        "dpid": 2,
        "priority": 1,
        "match": {
            "in_port": 2
        },
        "actions": [
            {
                "type": "PUSH_VLAN",
                "ethertype": 33024
            },
            {
                "type": "SET_FIELD",
                "field": "vlan_vid",
                "value": 4097
            },
            {
                "type": "OUTPUT",
                "port": 3
            }
        ]
    }
    flow7 = {
        "dpid": 2,
        "priority": 1,
        "match": {
            "vlan_vid": 0
        },
        "actions": [
            {
                "type": "POP_VLAN",
                "ethertype": 33024
            },
            {
                "type": "OUTPUT",
                "port": 1
            }
        ]
    }
    flow8 = {
        "dpid": 2,
        "priority": 1,
        "match": {
            "vlan_vid": 1
        },
        "actions": [
            {
                "type": "POP_VLAN",
```

```
                "ethertype": 33024
            },
            {
                "type": "OUTPUT",
                "port": 2
            }
        ]
    }
    res1 = requests.post(url, json.dumps(flow1), headers=headers)
    res2 = requests.post(url, json.dumps(flow2), headers=headers)
    res3 = requests.post(url, json.dumps(flow3), headers=headers)
    res4 = requests.post(url, json.dumps(flow4), headers=headers)
    res5 = requests.post(url, json.dumps(flow5), headers=headers)
    res6 = requests.post(url, json.dumps(flow6), headers=headers)
    res7 = requests.post(url, json.dumps(flow7), headers=headers)
    res8 = requests.post(url, json.dumps(flow8), headers=headers)
```

4.1.2.2 利用curl命令删除流表

> curl -X DELETE http://127.0.0.1:8080/stats/flowentry/clear/1 curl -X DELETE http://127.0.0.1:8080/stats/flowentry/clear/2

执行结果



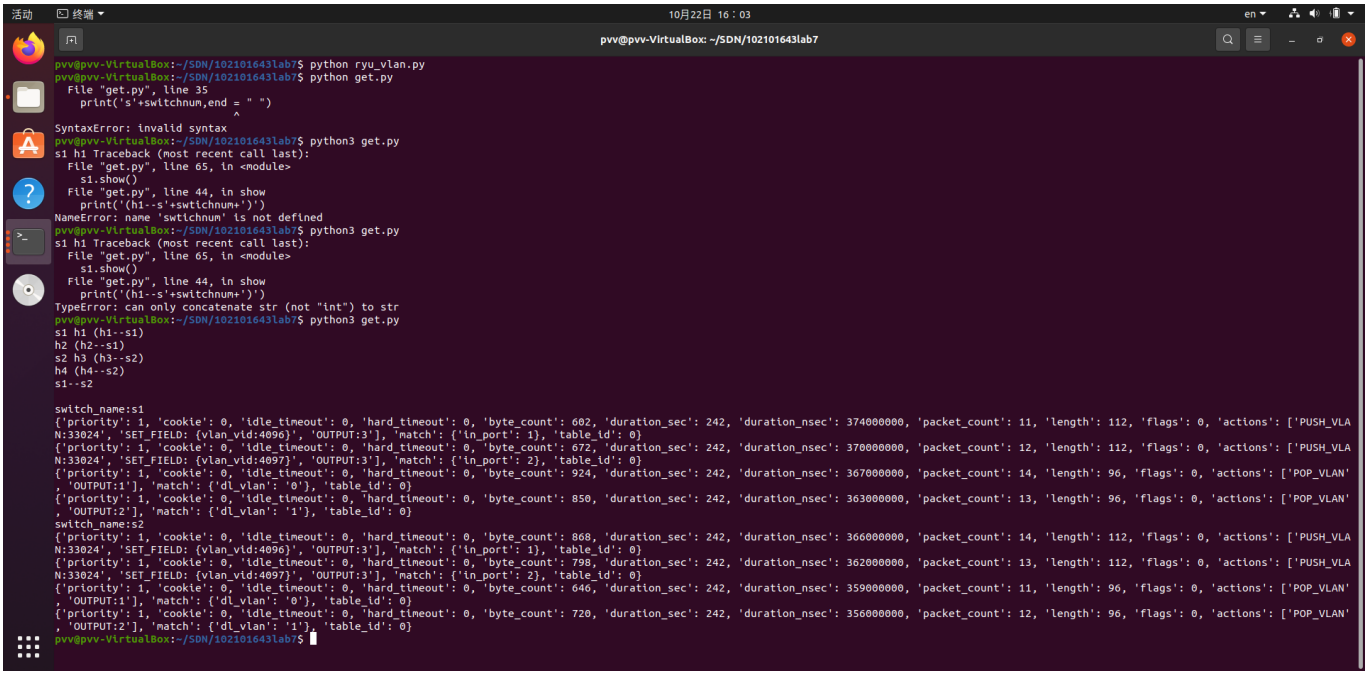
4.1.3 实现查看前序VLAN实验的拓扑信息，包括主机、交换机和链路，以及显示每台交换机的所有流表项

代码

```python
import requests
import time
import re


class GetNodes:
    def __init__(self, ip):
        self.ip = ip

    def get_switch_id(self):
        url = 'http://' + self.ip + '/stats/switches'
        re_switch_id = requests.get(url=url).json()
        switch_id_hex = []
        for i in re_switch_id:
            switch_id_hex.append(hex(i))

        return switch_id_hex

    def getflow(self):
        url = 'http://' + self.ip + '/stats/flow/%d'
        switch_list = self.get_switch_id()
        ret_flow = []
        for switch in switch_list:
            new_url = format(url % int(switch, 16))
            re_switch_flow = requests.get(url=new_url).json()
            ret_flow.append(re_switch_flow)
        return ret_flow

    def show(self):
        flow_list = self.getflow()
        for flow in flow_list:
            for dpid in flow.keys():
                dp_id = dpid
                switchnum= '{1}'.format(hex(int(dp_id)), int(dp_id))
                print('s'+switchnum,end = " ")
                switchnum = int(switchnum)
            for list_table in flow.values():
                for table in list_table:
                    string1 = str(table)
                    if re.search("'dl_vlan': '(.*?)'", string1) is not None:
                       num = re.search("'dl_vlan': '(.*?)'", string1).group(1);
                       if num == '0' and switchnum == 1:
                          print('h1',end = " ")
                          print('(h1--s'+swtichnum+')')
                       if num == '1' and switchnum == 1:
                          print('h2',end = " ")
                          print('(h2--s'+swtichnum+')')
                       if num == '0' and switchnum == 2:
                          print('h3',end = " ")
                          print('(h3--s'+swtichnum+')')
                       if num == '1' and switchnum == 2:
                          print('h4',end = " ")
                          print('(h4--s'+swtichnum+')')
```

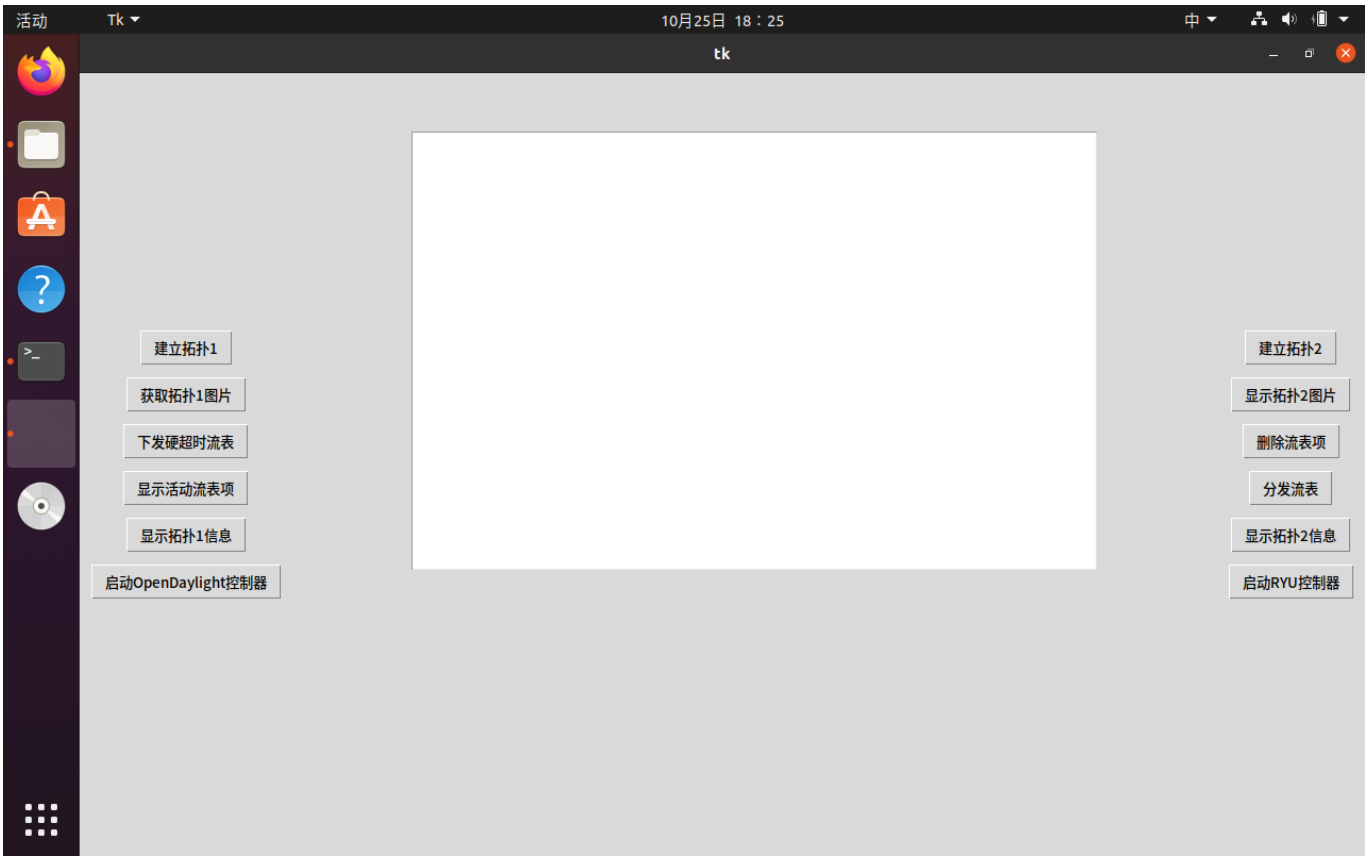
```
        print('s1--s2')
        print("")
        flow_list = self.getflow()
        for flow in flow_list:
            for dpid in flow.keys():
                dp_id = dpid
                print('switch_name:s{1}'.format(hex(int(dp_id)), int(dp_id)))
            for list_table in flow.values():
                for table in list_table:
                    print(table)
s1 = GetNodes("127.0.0.1:8080")
s1.show()
```

执行结果

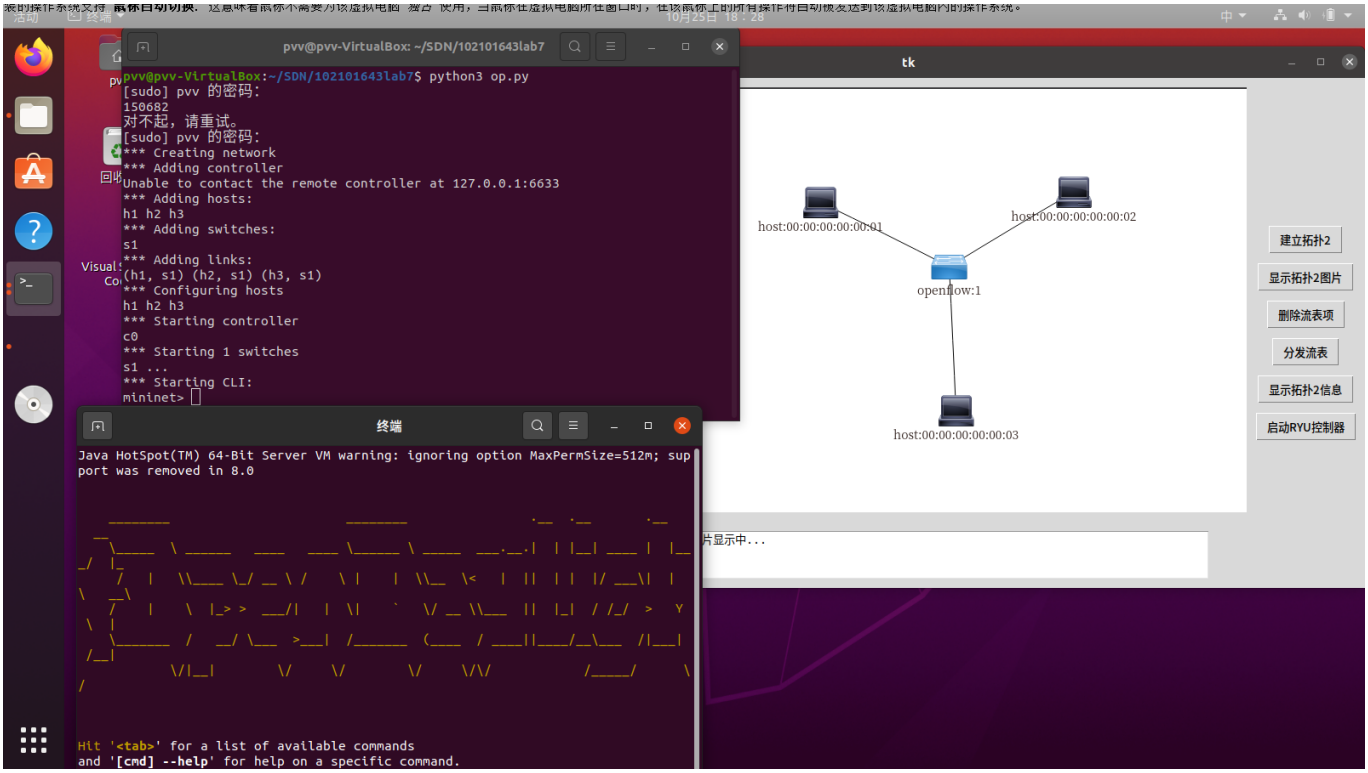


进阶要求

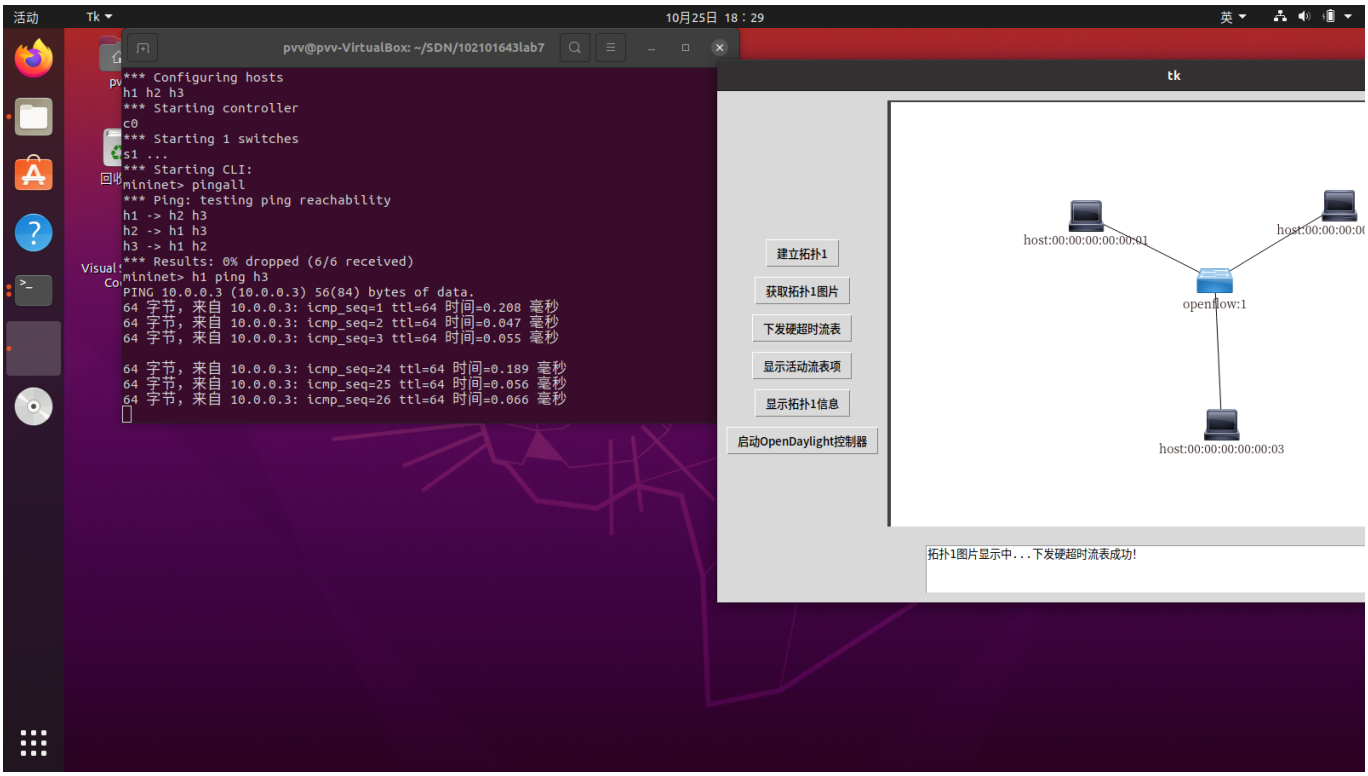
前端页面



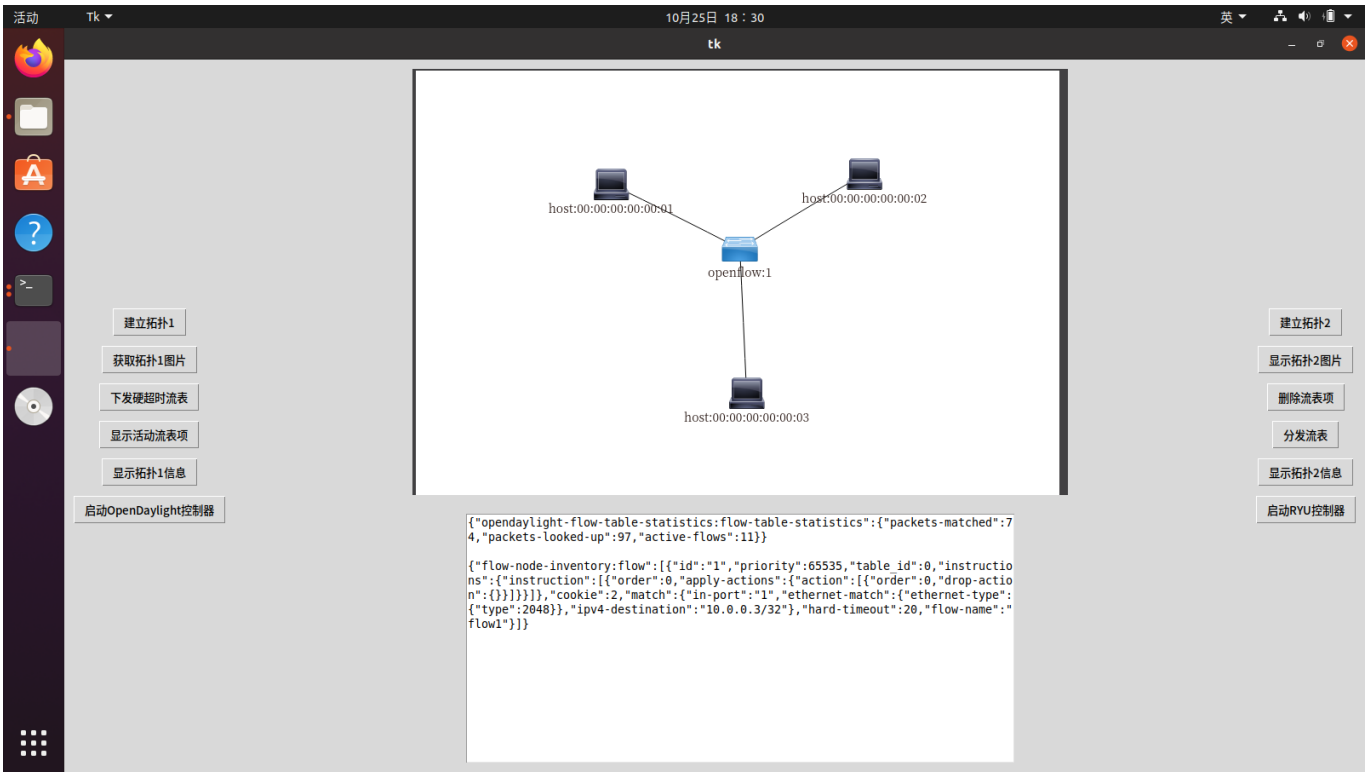
启动ODL并建立拓扑结果



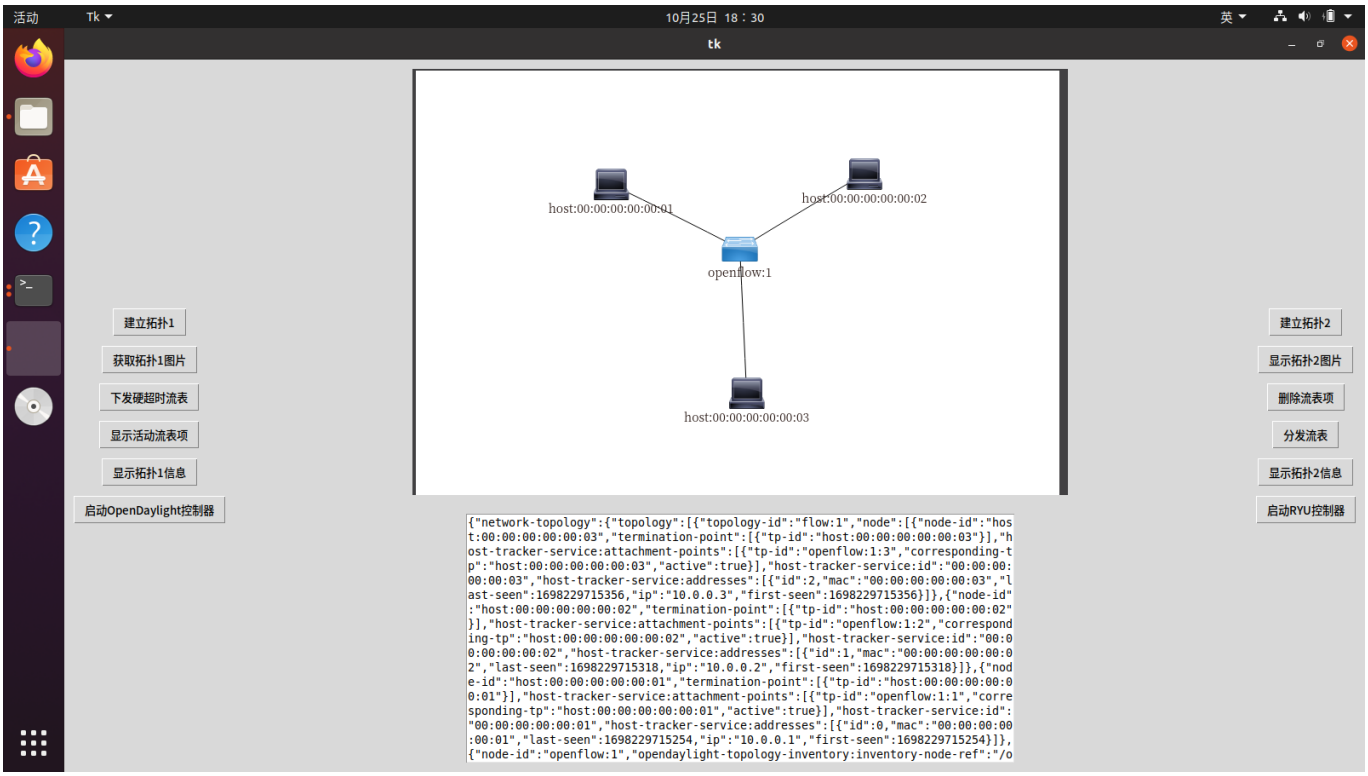
下发硬超时流表



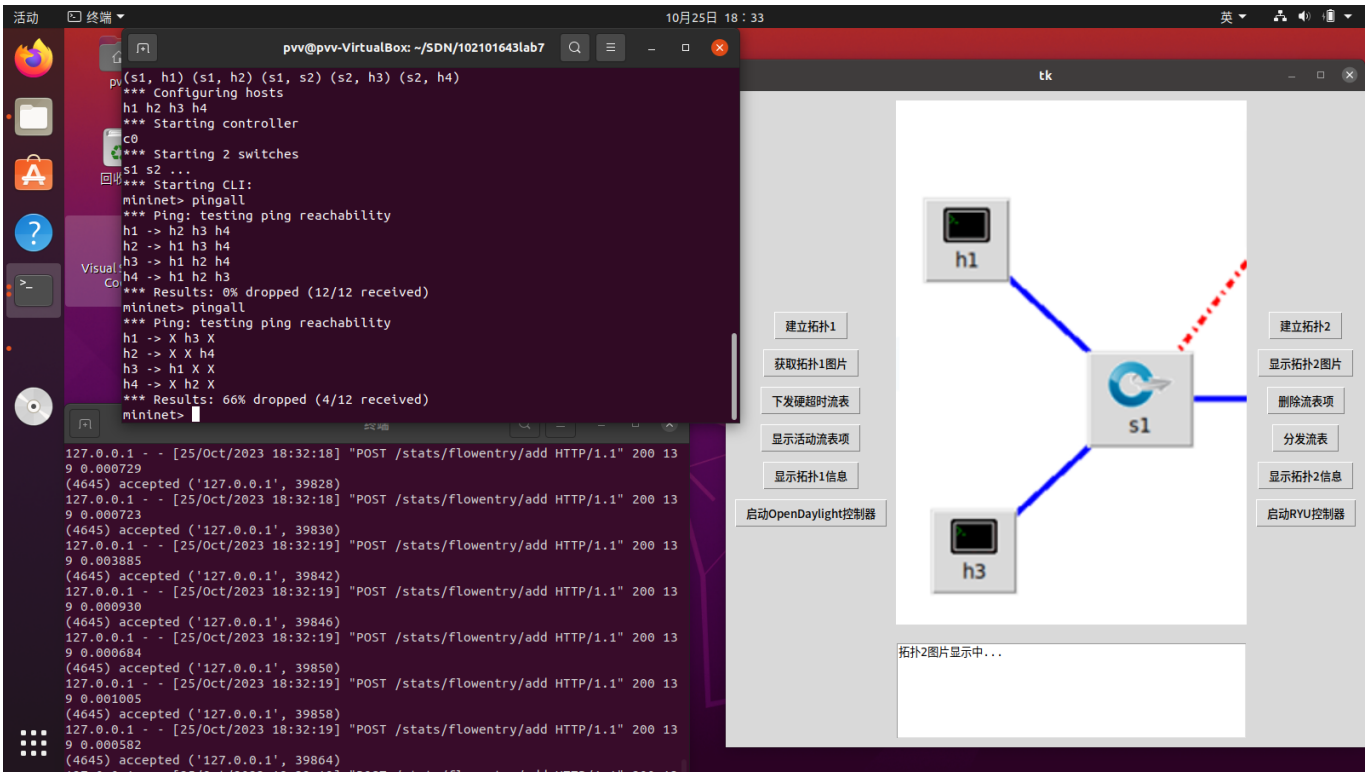
显示活动流表项



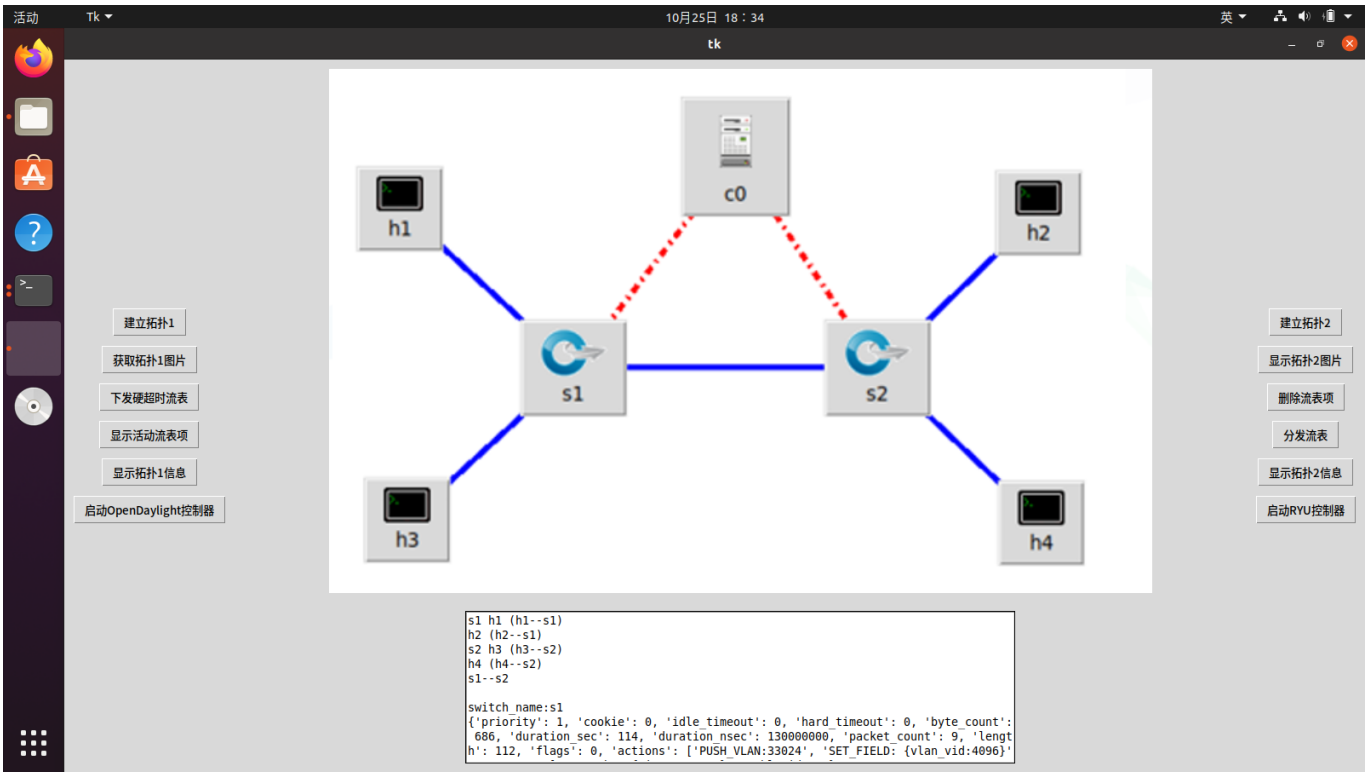
显示拓扑信息



实现与前序VLAN划分一样的拓扑



显示拓扑信息



GitHub链接

https://github.com/Paradox354/SDNlab7.git

个人阶段总结

实验难度：很难

本次实验的基本要求比较难，在查找ODL控制器的API遇到了很大的麻烦，还好网上有历年的作业可以参考，看了他们使用的API后，再根据自己的一些后端知识还是能解决一些简单的request问题的。在基础要求第三步的时候，要求返回拓扑信息，这个在历年作业上是找不到的，我只能通过查阅ODL控制器的API文档来寻找，最后还是找到了那个接口，但是有点遗憾，因为时间问题，我没来得及对请求回来的数据进行处理，不然的话显示的拓扑信息会更加友好。在进行第四步的时候，我换成了Ryu控制器，但是Ryu控制器我一直觉得有个bug，无法可视化拓扑，明明已经连接上了，也已经pingall了，但是可视化页面就是没显示拓扑。在第四步显示拓扑的时候我就有对数据进行处理了，这样子看起来会比较友好（毕竟只需要在往届代码的基础上修改一下就行）。51_Open_lab对我这次实验的帮助是很大的。 在建立前端页面方面，我采用了python的GUI框架，主要原因还是因为VScode的环境还没配置好，所以就用现成的python环境了，当时在进行前端调用建立拓扑的时候还是遇到了一个问题，就是程序会卡在那个进程运行不了了，后来仔细想想才明白，建立完拓扑后就进入了mininet的命令行页面，所以应该再开一个线程运行，另外还得屏蔽cirl+c组合，否则对mininet的操作会对主进程造成影响。