SDN实验---Mininet实验2（模拟多数据中心带宽实验）

补充：NameError: name 'buffer' is not defined

```
>>> import sys
>>> if sys.version_info > (3,):
...     buffer = memoryview
>>> b = buffer('yay!'.encode())
>>> len(b)
4
```

因为在Python3中buffer已经被memoryview取代了，buffer在Python2中使用，所以我们可以在文件中加入

```
import sys
if sys.version_info > (3,):
    buffer = memoryview
```

一：Mininet模拟多数据中心流量带宽实验

（一）案例目的

通过Mininet模拟搭建基于不同数据中心的网络拓扑；

掌握多数据中心网络拓扑的构建；

熟悉网络性能测试工具Iperf，根据实验测试SDN网络的性能；

通过程序生成真实网络流量。

（二）为什么使用Mininet模拟数据中心--应用价值

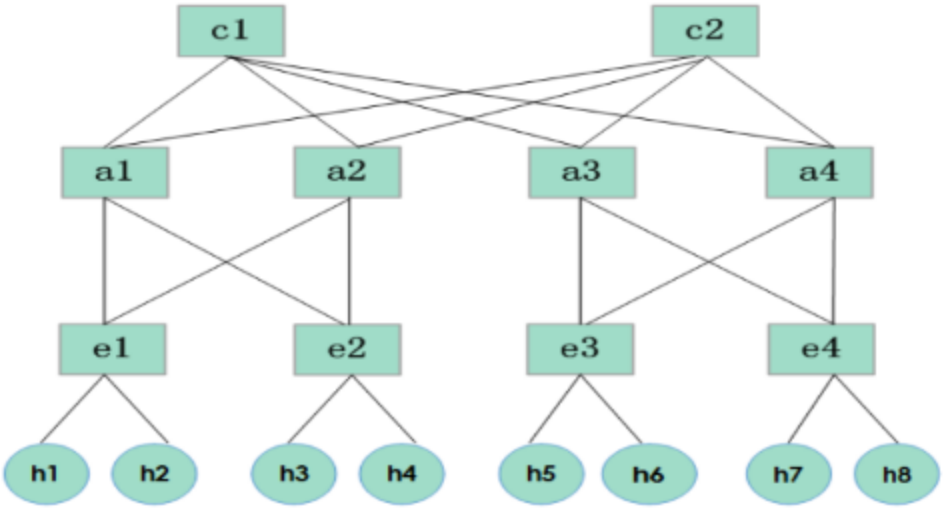
- 树状拓扑结构容错能力强
- 降低数据中心成本消耗
- 提供重新排列的全带宽无阻碍路径
- 提高带宽利用率
- 分析数据中心网络流量性能
- 为真实数据中心和仿真测试床提供有用信息

Mininet最常用的场景就是数据中心。因为Mininet可以模拟出很复杂的网络拓扑，而不需要硬件的支持，就可以搭建出不同的数据中心的拓扑。 可以为真正的数据中心网络的搭建起到模拟预测实验作用，为真实的数据中心的成本带来一定的节省。

二：数据中心网络拓扑

（一）数据中心网络拓扑结构

核心交换机：c1、c2，聚合交换机：a1-a4，边缘交换机：e1-e4，主机：h1~h8。Mininet中自带的iperf性能测试工具可以测试不同主机间通信的带宽性能质量，可以针对相同边缘交换机、相同聚合交换机不同边缘交换机、相同核心交换机不同聚合交换机下的主机进行测试。



存在线路冗余（多条链路可达），容错能力强-----胖树拓扑

（二）实现网络拓扑---按照结构实现，代码不唯一

Python | 复制代码

```python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import RemoteController
from mininet.link import TCLink
from mininet.util import dumpNodeConnections

class MyTopo(Topo):

    def __init__(self):
        super(MyTopo,self).__init__()

        #Marking the number of switch for per level
        L1 = 2;
        L2 = L1*2
        L3 = L2

        #Starting create the switch
        c = []    #core switch
        a = []    #aggregate switch
        e = []    #edge switch

        #notice: switch label is a special data structure
        for i in range(L1):
            c_sw = self.addSwitch('c{}'.format(i+1))    #label from 1 to n,not start with 0
            c.append(c_sw)

        for i in range(L2):
            a_sw = self.addSwitch('a{}'.format(L1+i+1))
            a.append(a_sw)

        for i in range(L3):
            e_sw = self.addSwitch('e{}'.format(L1+L2+i+1))
            e.append(e_sw)

        #Starting create the link between switchs
        #first the first level and second level link
        for i in range(L1):
            c_sw = c[i]
            for j in range(L2):
                self.addLink(c_sw,a[j])

        #second the second level and third level link
        for i in range(L2):
            self.addLink(a[i],e[i])
```

```
            if not i%2:
                self.addLink(a[i],e[i+1])
            else:
                self.addLink(a[i],e[i-1])

        #Starting create the host and create link between switchs and hosts
        for i in range(L3):
            for j in range(2):
                hs = self.addHost('h{}'.format(i*2+j+1))
                self.addLink(e[i],hs)



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

（三）使用Mininet测试

Shell | 复制代码

```
sudo mn --custom ./data_center_topo.py --topo=mytopo --controller=remote
```

```
njzy@njzy-Inspiron-5493:/opt/mininet/experiment/day02_flowtable$ sudo mn --custom ./data_center_topo.py --topo=mytopo --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
a3 a4 a5 a6 c1 c2 e7 e8 e9 e10
*** Adding links:
(a3, e7) (a3, e8) (a4, e7) (a4, e8) (a5, e9) (a5, e10) (a6, e9) (a6, e10) (c1, a3) (c1, a4) (c1, a5) (c1, a6) (c2, a3) (c2, a4) (c2, a5) (c2, a6)
 (e7, h1) (e7, h2) (e8, h3) (e8, h4) (e9, h5) (e9, h6) (e10, h7) (e10, h8)
```

三：流量模拟

（一）为什么需要流量模拟

网络性能评估中一个巨大的挑战就是如何生成真实的网络流量，可以通过程序来创造人工的网络流量，通过建立测试环境来模拟真实的状况。此应用主要以数据中心网络为目标场景，在mininet仿真环境中尽可能地还原数据中心内部的真实流量情况。

（二）流量随机模型在Mininet中的应用

流量随机模型：主机向在网络中的另一任意主机以等概率发送数据包。

使用mininet中的iperf工具在网络中生成UDP流量，iperf客户端传送数据流到iperf的服务端，由服务端接收并记录相关信息。

我们需要实现的是将批处理流的自定义命令添加到mininet中，在mininet中使用此自定义命令，实现上述功能。

四：自定义命令拓展实现---为流量模拟做准备

在mininet中进行自定义命令功能拓展主要分为4步：

- 修改mininet/net.py
- 修改mininet/cli.py
- 修改bin/mn
- 重新安装Mininet核心文件：
  ~/mininet/util/install.sh -n

（一）修改net.py

Python | 复制代码

```python
def iperf_single( self, hosts=None, udpBw='10M',period=60,port=5001):
        """Run iperf between two hosts using UDP.
           hosts: list of hosts; if None, uses first and last hosts
           returns: results two-element array of server and client speeds
        """
        if not hosts:
            return
        else:
            assert len(hosts) == 2    #这段代码我们要求一定要有两个参数，即下面的client,和server

        client, server = hosts
        filename = client.name[1:]+'.out'
        output('*** Iperf:testing bandwidth between ')
        output('%s and %s\n'%(client.name,server.name))    #这里在Mininet交互界面显示提示信息
        iperfArgs = 'iperf -u '
        bwArgs = '-b '+udpBw+' '    #设置命令和参数，这是要在client和server上执行的
        print("***start server***")
        server.cmd(iperfArgs+'-s -i 1'+' > /home/njzy/temp_log/'+filename+'&')    #服务器端执行指令，并且将返回的信息存放在文件中
                                                    #注意：对应我们存放日志的目录，一定是真实存在的
        print("***start client***")
        client.cmd(iperfArgs+'-t '+str(period)+' -c '+server.IP()+' '+bwArgs    #客户端执行指令，并且将返回的信息存放在文件中
                    +' > /home/njzy/temp_log/'+'client'+filename+'&')

    def iperfMulti(self,bw,period=60):
        base_port = 5001
        server_list = []
        client_list = [h for h in self.hosts]
        host_list = []
        host_list = [h for h in self.hosts]    #收集所有主机信息

        cli_outs = []
        ser_outs = []

        _len = len(host_list)
        for i in xrange(0,_len):    #按照主机数目进行循环，每次选择一台主机，作为客户端
            client = host_list[i]
            server = client

```

```
            while (server==client):    #如果客户端和服务端是同一台主机，那么我们随机从主机中国选择一台新的主机作为服务端，直到找到一台与客户端不同的主机，用来做服务端
                server = random.choice(host_list)

            server_list.append(server)
            self.iperf_single(hosts=[client,server],udpBw=bw,period=period,port=base_port)    #客户端和服务端进行带宽测试
            sleep(.05)
            base_port += 1    #更换端口号，做到随机

        sleep(period)
        print("test has done")    #结束，打印提示信息
```

（二）修改cli.py将iperfmulti命令在CLI类中注册

Python | 复制代码

```python
def do_iperfmulti( self, line ):
        """
        Multi iperf UDP test between nodes
        """
        args = line.split()
        if len(args) == 1:
            udpBw = args[0]
            self.mn.iperfMulti(udpBw)
        elif len(args) == 2:
            udpBw = args[0]
            period = args[1]
            self.mn.iperfMulti(udpBw,float(period))
        else:
            error( 'invalid number of args: iperfMulti udpBw period\n '+
                    'udpBw examples:1M 120\n' )
```

```
    def do_iperfudp( self, line ):

    def do_iperfmulti( self, line ):
        """
        Multi iperf UDP test between nodes
        """
        args = line.split()
        if len(args) == 1:
            udpBw = args[0]
            self.mn.iperfMulti(udpBw)
        elif len(args) == 2:
            udpBw = args[0]
            period = args[1]
            self.mn.iperfMulti(udpBw,float(period))
        else:
            error( 'invalid number of args: iperfMulti udpBw period\n '+
                   'udpBw examples:1M 120\n' )

    def do_intfs( self, _line ):

    def do_dump( self, _line ):
```

（三） 在mininet/bin/mn文件中加入iperfmulti可执行命令

Python | 复制代码

```
TESTS = { name: True
          for name in ( 'pingall', 'pingpair', 'iperf', 'iperfudp','iperfmu
lti' ) }
```

Python | 复制代码

```
# Map to alternate spellings of Mininet() methods
ALTSPELLING = { 'pingall': 'pingAll', 'pingpair': 'pingPair',
                'iperfudp': 'iperfUdp','iperfmulti': 'iperfMulti' }
```

（四）重新编译mininet---因为我们修改了mininet内核文件

进入mininet/util目录，重新编译安装mininet。

#~/mininet/util/install.sh –n

因为我们已经安装OpenFlow协议和openvswitch，所以不需要再加3V

重新创建网络，如mn，输入iperf，可用table补全iperfmulti，从而可使用iperfmulti进行流量随机模型的测试。

Shell | 复制代码

```
sudo mn
iperfmulti    后面会自动补全
```

五：进行网络测试

（一）开始Ryu---为了防止广播风暴，使用生成树协议《重点注意协议文件选择》

Shell | 复制代码

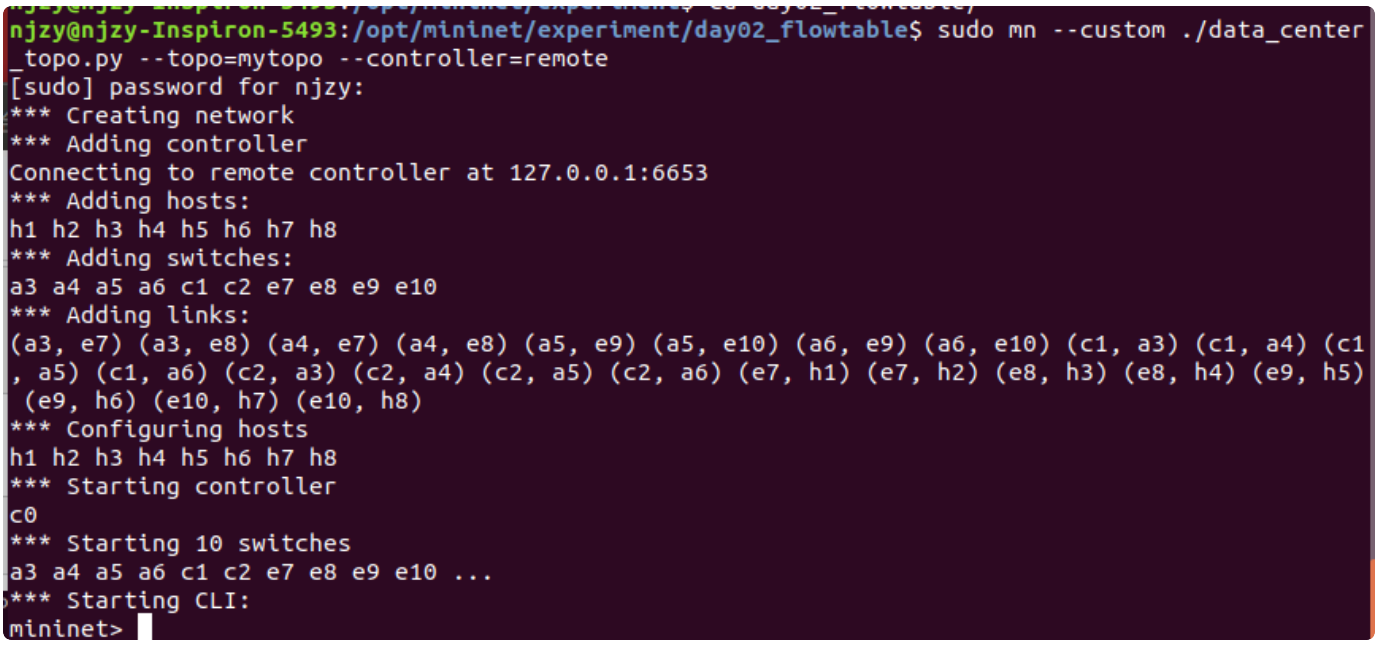
```
ryu-manager simple_switch_stp_13.py
！注意：是使用simple_switch_stp_13协议，不要使用simple_switch_stp文件，不然会出问题
```

```
njzy@njzy-Inspiron-5493:~/CODE/python/SDN_Controller/ryu/ryu/app$ ryu-manager si
mple_switch_stp_13.py
loading app simple_switch_stp_13.py
loading app ryu.controller.ofp_handler
instantiating app None of Stp
creating context stplib
instantiating app simple_switch_stp_13.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
[STP][INFO] dpid=0000000000000005: Join as stp bridge.
[STP][INFO] dpid=0000000000000005: [port=4] DESIGNATED_PORT     / LISTEN
[STP][INFO] dpid=0000000000000005: [port=1] DESIGNATED_PORT     / LISTEN
[STP][INFO] dpid=0000000000000005: [port=2] DESIGNATED_PORT     / LISTEN
[STP][INFO] dpid=0000000000000006: Join as stp bridge.
[STP][INFO] dpid=0000000000000005: [port=3] DESIGNATED_PORT     / LISTEN
```

（二）Mininet启动网络拓扑

Shell | 复制代码

```
sudo mn --custom ./data_center_topo.py --topo=mytopo --controller=remote
```



（三）使用iperf命令，进行TCP带宽测试

注意：在测试之前先多ping几次<h1 ping h2>（找到可以ping通），使得网络拓扑结构提前存在控制器中，不然容易直接退出

```
mininet> h1 ping h2 -c 4
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3062ms
pipe 4
mininet> h1 ping h2 -c 4
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=9.45 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.079 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.055 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.055/2.414/9.453/4.063 ms
```

1.同一交换机内部的主机间连通性及通信带宽测试

Shell | 复制代码

```
1  iperf h1 h2
```

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['10.2 Gbits/sec', '11.8 Gbits/sec']
```

2.相同汇聚交换机下不同机架的主机间测试

Shell | 复制代码

```
1  iperf h1 h3
```

```
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['53.9 Gbits/sec', '53.9 Gbits/sec']
```

3.相同核心交换机不同汇聚交换机下的主机间测试

Shell 复制代码

```
iperf h1 h5
```

```
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['52.7 Gbits/sec', '52.7 Gbits/sec']
```

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['68.5 Gbits/sec', '68.5 Gbits/sec']
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['56.5 Gbits/sec', '56.6 Gbits/sec']
mininet> iperf h1 h5
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['47.9 Gbits/sec', '48.0 Gbits/sec']
```

（四）使用iperfmulti命令，进行UDP带宽测试

Shell 复制代码

```
iperfmulti 0.025M
```

```
mininet> iperfmulti 0.025M
*** Iperf:testing bandwidth between h1 and h6
***start server***
***start client***
*** Iperf:testing bandwidth between h2 and h4
***start server***
***start client***
*** Iperf:testing bandwidth between h3 and h4
***start server***
***start client***
*** Iperf:testing bandwidth between h4 and h3
***start server***
***start client***
*** Iperf:testing bandwidth between h5 and h7
***start server***
***start client***
*** Iperf:testing bandwidth between h6 and h7
***start server***
***start client***
*** Iperf:testing bandwidth between h7 and h4
***start server***
***start client***
*** Iperf:testing bandwidth between h8 and h3
***start server***
***start client***
test has done
mininet>
```

查看流量日志

```
njzy@njzy-Inspiron-5493:~/temp_log$ ls
1.out  3.out  5.out  7.out  client1.out  client3.out  client5.out  client7.out
2.out  4.out  6.out  8.out  client2.out  client4.out  client6.out  client8.out
njzy@njzy-Inspiron-5493:~/temp_log$ pwd
/home/njzy/temp_log
```

```
njzy@njzy-Inspiron-5493:~/temp_log$ cat client1.out
------------------------------------------------------------
Client connecting to 10.0.0.5, UDP port 5001
Sending 1470 byte datagrams, IPG target: 448615.24 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 39708 connected with 10.0.0.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-60.6 sec   194 KBytes  26.2 Kbits/sec
[  3] Sent 135 datagrams
[  3] Server Report:
[  3]  0.0-60.6 sec   194 KBytes  26.2 Kbits/sec   0.000 ms    0/  135 (0%)
njzy@njzy-Inspiron-5493:~/temp_log$ cat 1.out
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.5 port 5001 connected with 10.0.0.1 port 39708
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec  4.31 KBytes  35.3 Kbits/sec   0.708 ms    0/    3 (0%)
[  3]  1.0- 2.0 sec  2.87 KBytes  23.5 Kbits/sec   0.623 ms    0/    2 (0%)
[  3]  2.0- 3.0 sec  2.87 KBytes  23.5 Kbits/sec   0.548 ms    0/    2 (0%)
[  3]  3.0- 4.0 sec  2.87 KBytes  23.5 Kbits/sec   0.482 ms    0/    2 (0%)
[  3]  4.0- 5.0 sec  4.31 KBytes  35.3 Kbits/sec   0.397 ms    0/    3 (0%)
[  3]  5.0- 6.0 sec  2.87 KBytes  23.5 Kbits/sec   0.349 ms    0/    2 (0%)
[  3]  6.0- 7.0 sec  2.87 KBytes  23.5 Kbits/sec   0.308 ms    0/    2 (0%)
[  3]  7.0- 8.0 sec  2.87 KBytes  23.5 Kbits/sec   0.271 ms    0/    2 (0%)
[  3]  8.0- 9.0 sec  4.31 KBytes  35.3 Kbits/sec   0.224 ms    0/    3 (0%)
[  3]  9.0-10.0 sec  2.87 KBytes  23.5 Kbits/sec   0.197 ms    0/    2 (0%)
[  3] 10.0-11.0 sec  2.87 KBytes  23.5 Kbits/sec   0.174 ms    0/    2 (0%)
```