软件定义网络SDN

报告人　吴建胜

网络安全与自然语言处理：

[1]一种基于自动机的分词方法, 计算机工程与应用，2005
[2]基于机群技术的邮件过滤系统设计，微计算机信息, 2008
[3]Research in Anti-Spam Method Based on Bayesian Filtering，PACIIA2008
[4]Improvement of Chinese Spam filtering method based on Bayesian Classification，ICFCC2010
[5]New intrusion detection method based on partition ，CECN2014
[6]KDDCUP99数据集的数据分析研究， 计算机应用与软件，2014

图像处理：

[7]An effective flame Segmentation method based on OHTA color space，ISAM2012
[8]Research on Background Subtraction Based on Center-symmetric Local Binary Patterns，IMSNA2013
[9]一种基于视频的火焰检测方法， 计算机应用与软件，2014
[10]基于特征字符识别的文本图像方向判定，计算机工程与设计，2015
[11]动态阈值的Vibe运动目标检测， 计算机工程与应用，2017

其他：

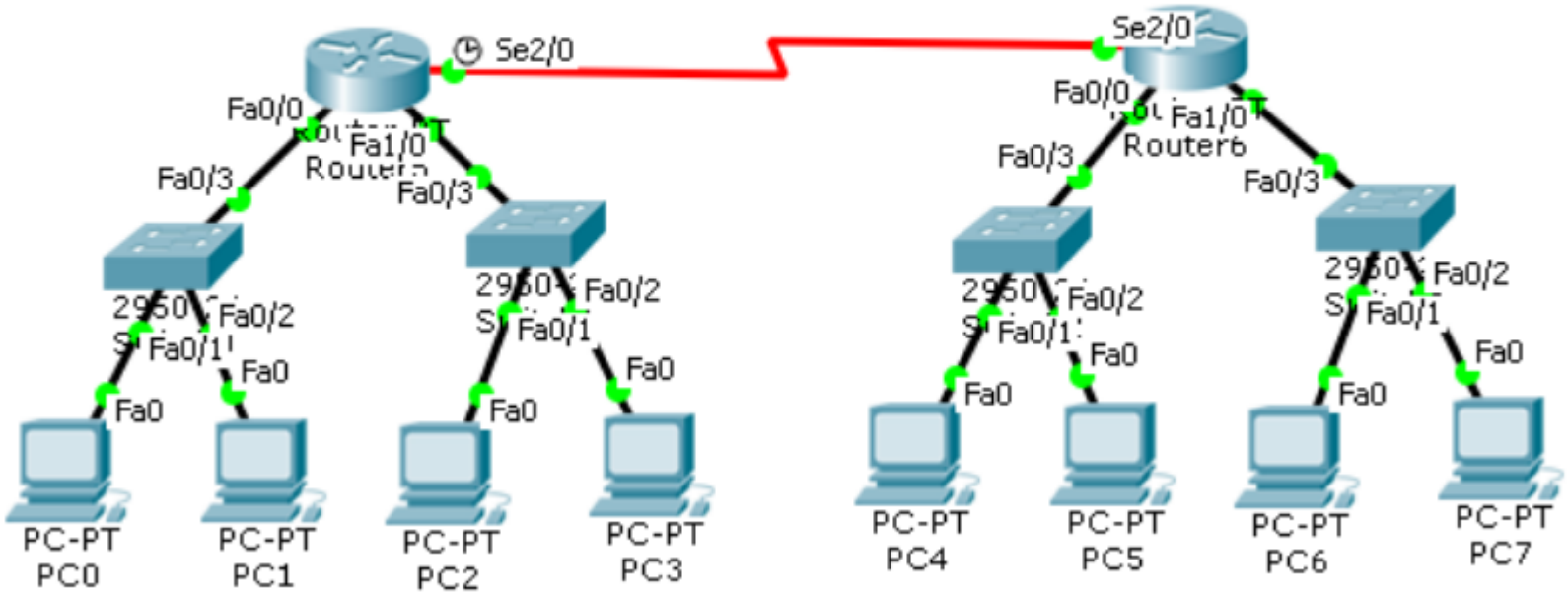
[12] 露天矿多采区协同开采资源配置优化，金属矿山，2017

目录

SDN起源

- 软件定义网络(Software Defined Network, SDN)
- 2006年，斯坦福大学，Nick McKeown教授，Clean Slate课题，提出了Openflow的思想用于校园网络的试验创新，最终目的是要重新发明英特网
- 2007年，斯坦福大学，学生Martin Casado ，网络安全与管理的项目Ethane
- 2008年，Nick教授等人提出了OpenFlow 的概念，在ACM SIGCOMM联合发表论文“OpenFlow: Enabling Innovation in Campus Networks”
- 2012年，国家“863”项目“未来网络体系结构和创新环境” 获得科技部批准（清华大学、中科院计算所、北邮、东南大学、北京大学）
- 2012，中国SDN与开放网络高峰会议，北京
  - ✓ 成立“中国SDN与开放网络专业委员会”

传统网络回顾


Se2/0
Se2/0
Fa0/0
Fa1/0
Fa0/3
Fa0/3
Fa0/2
Fa0/1
Fa0
Fa0
Fa0/2
Fa0/1
Fa0
Fa0
Fa0/0
Fa1/0
Router6
Fa0/3
Fa0/3
Fa0/2
Fa0/1
Fa0
Fa0
Fa0/2
Fa0/1
Fa0
Fa0
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
PC-PT
PC4
PC-PT
PC5
PC-PT
PC6
PC-PT
PC7

网络架构

网络功能

厂商操作系统

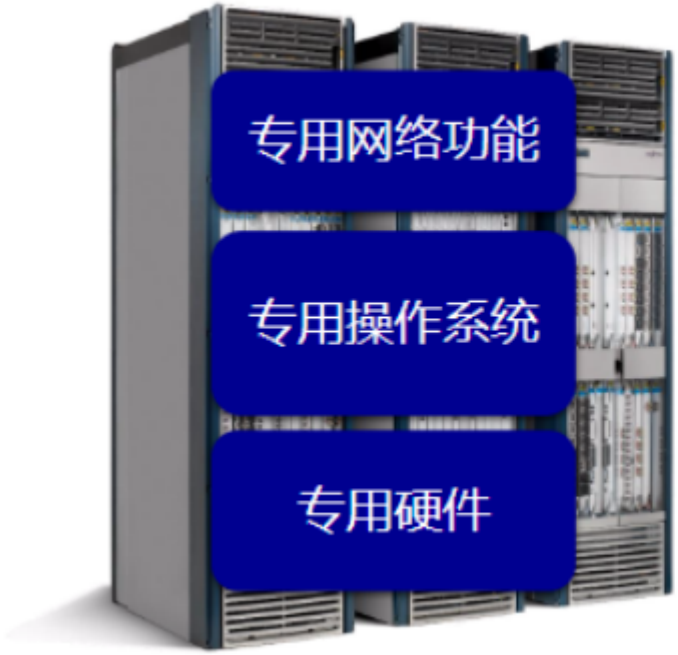
定制化硬件

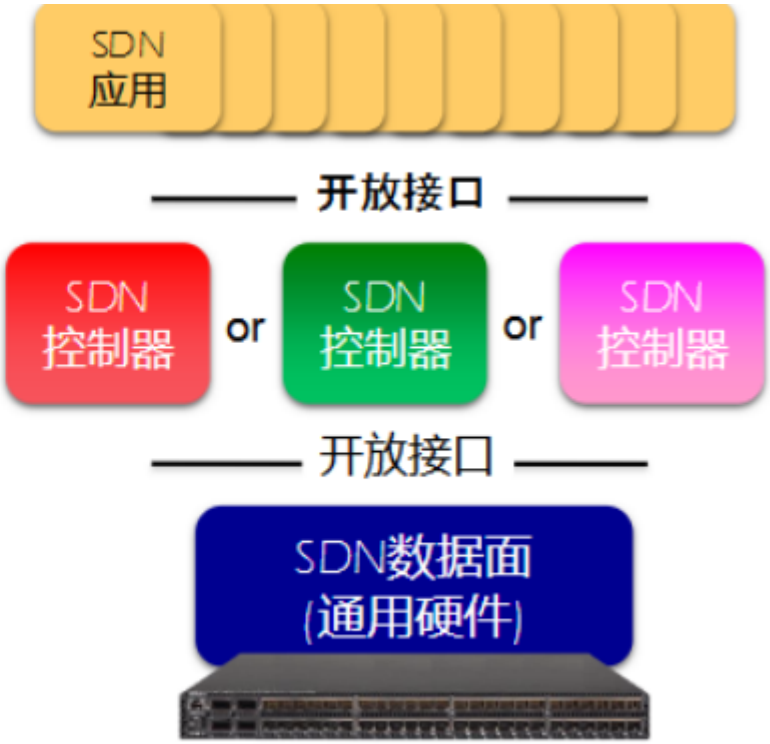
软件实现　　封闭

厂商专有　不可移植

厂商负责制的封闭系统　行业创新基本停滞

从传统网络到SDN





封闭系统

开放系统

权威SDN定义

SDN architecture **decouples** the network control and forwarding functions enabling the network control to **become directly programmable** and the underlying infrastructure to be **abstracted** for applications and network services. The **OpenFlow®** protocol is a foundational element for building SDN solutions.

ON.LAB

SDN's key attributes include: **separation of the data and control planes**; **a uniform, vendor-agnostic interface**(called OpenFlow) between control and data planes; **a logically centralized control plane** that offers a consistent, network-wide view to programmers or operators.

SDN解读

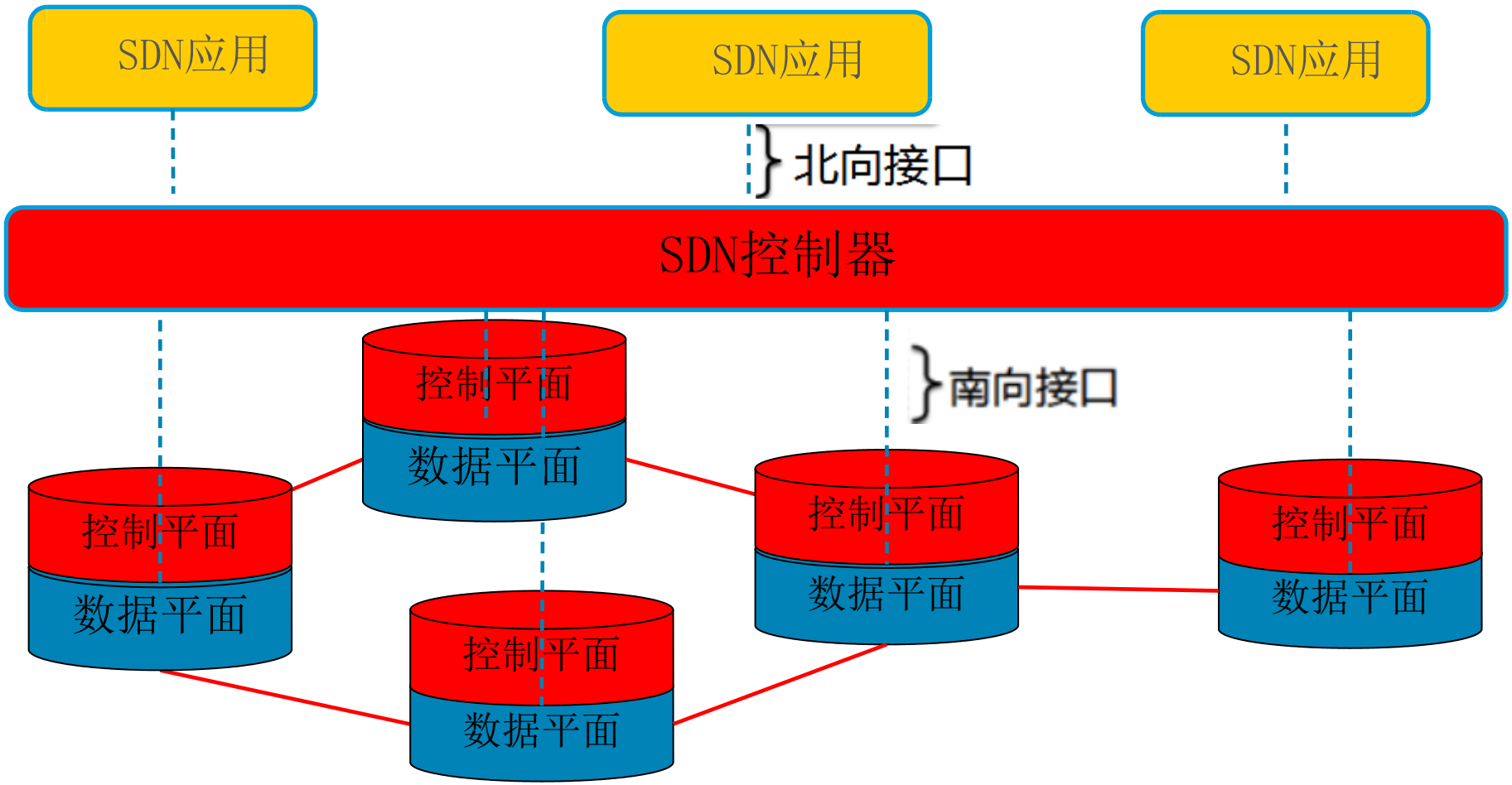
网络开放可编程

数控分离

逻辑上集中控制

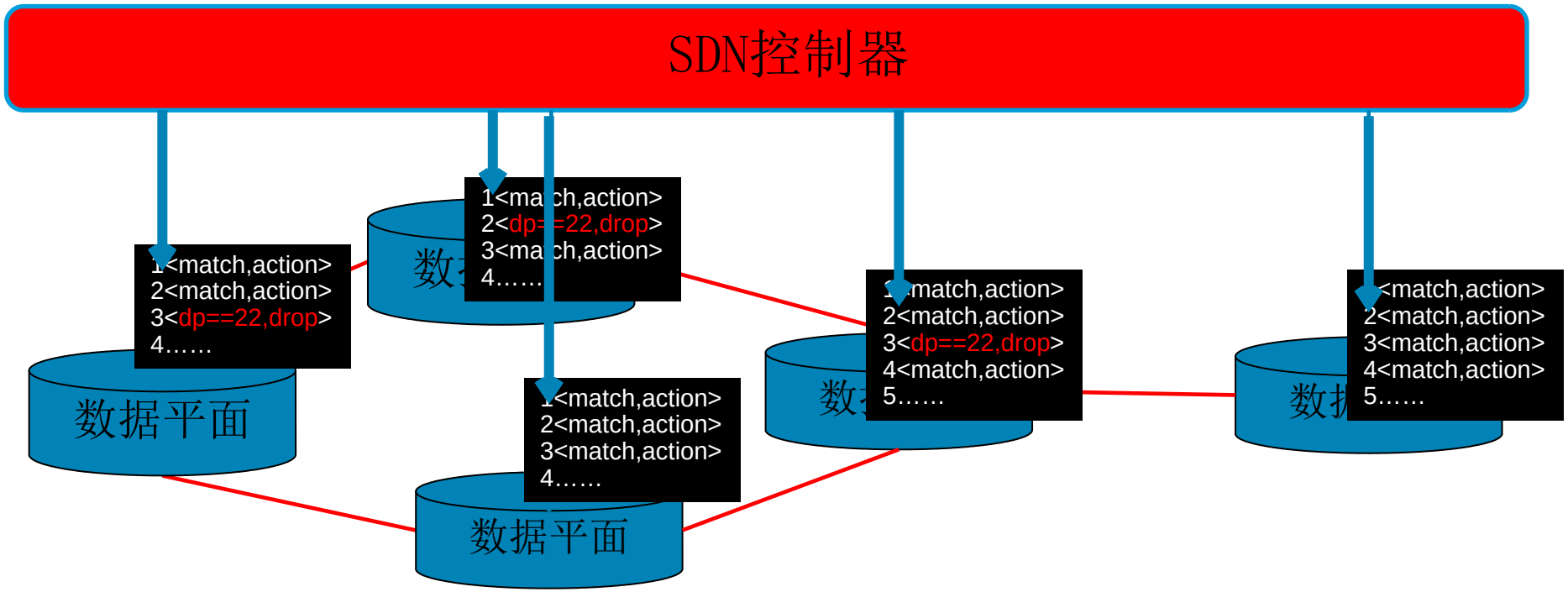
新的网络体系结构

SDN分层体系结构


SDN应用
SDN应用
SDN应用
北向接口
SDN控制器
南向接口
控制平面
数据平面
控制平面
数据平面
控制平面
数据平面
控制平面
数据平面
控制平面
数据平面

SDN工作原理

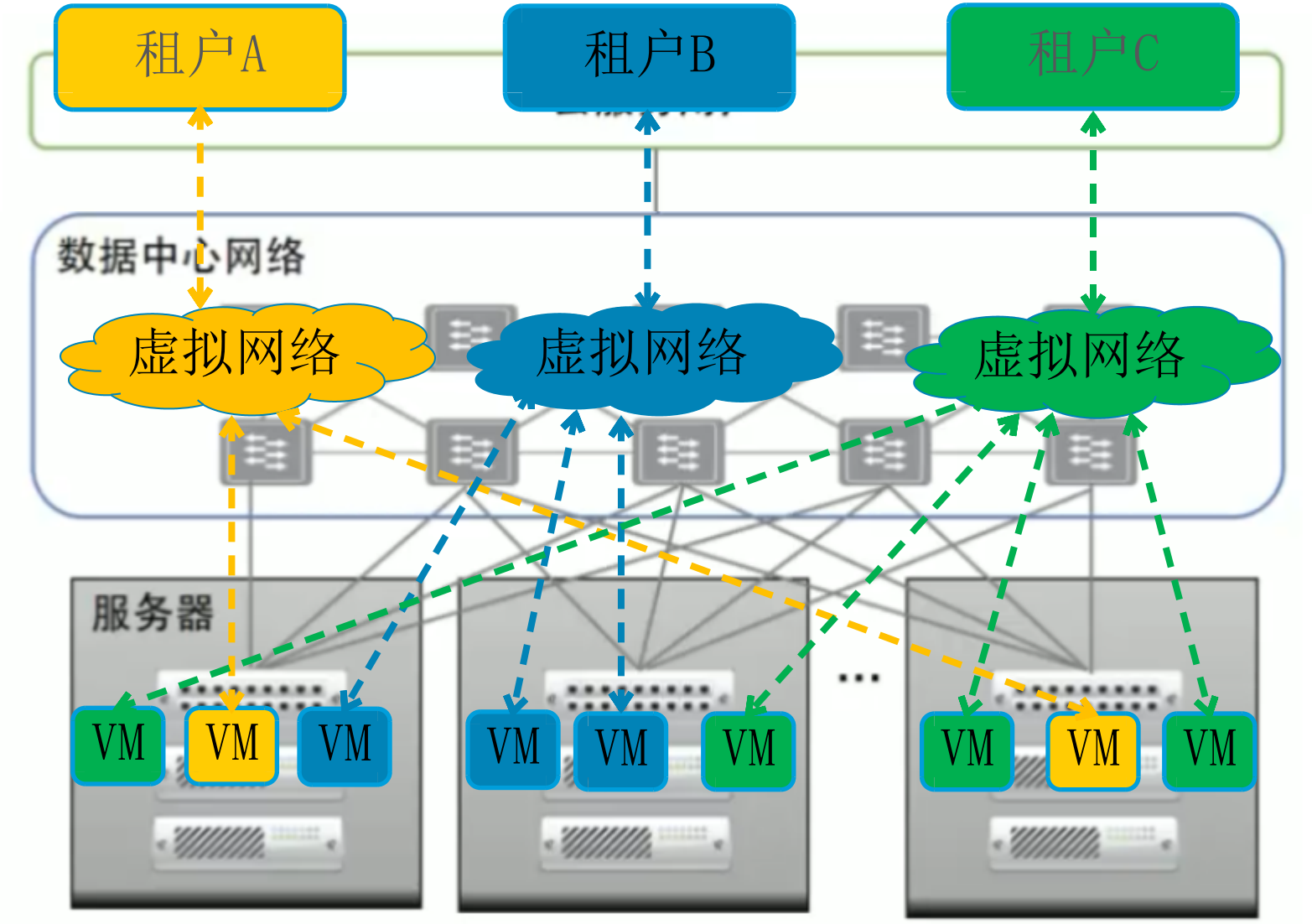
```
firewall.c
        ……
        If (pkt.tcp.dport==22)
                dropPacket(pkt);
        ……
```



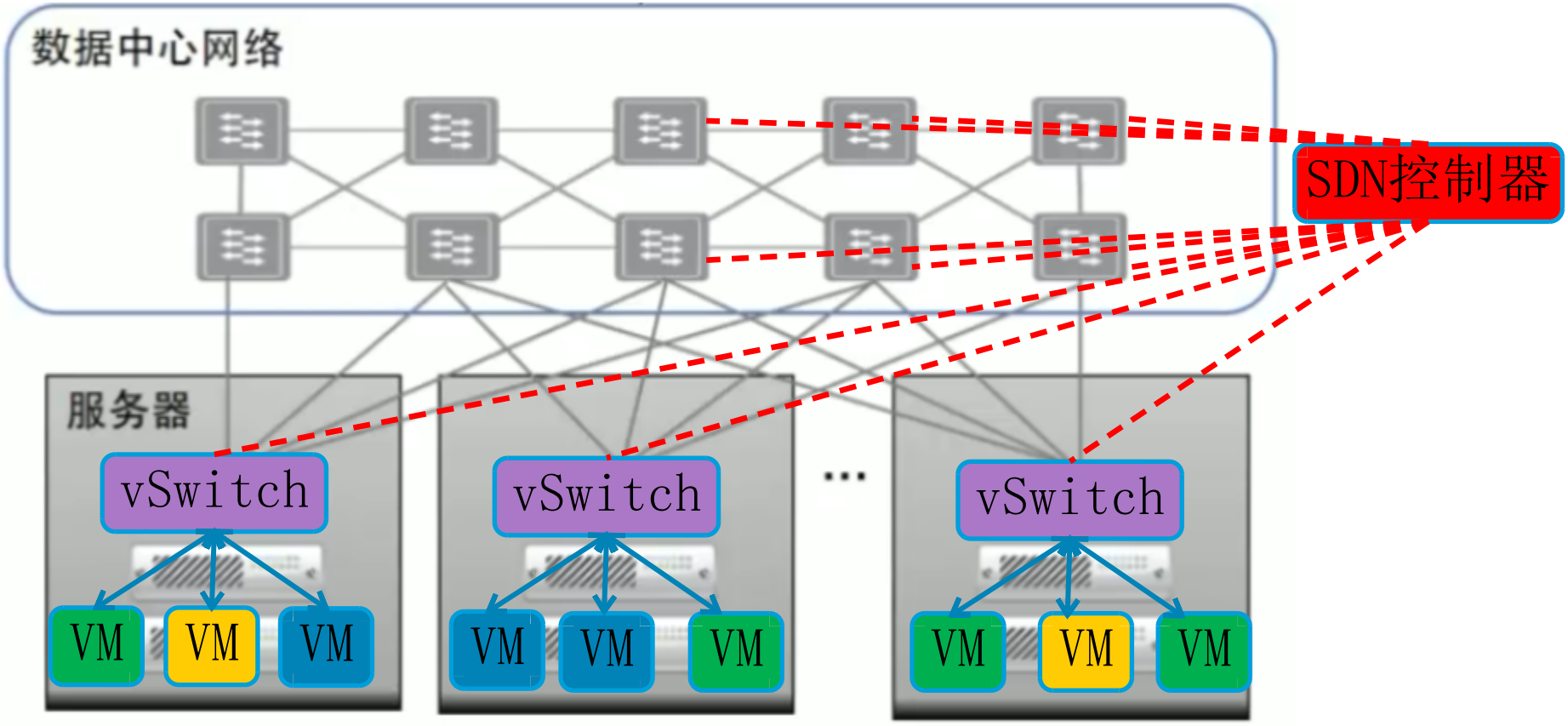
开源SDN控制器

	NOX (2008)	Ryu (2012)	Floodlight (2012.1)	ONOS (2014.12)	ODL (2013.12)
系统架构	Centralized multi-threaded	Centralized multi-threaded	Centralized	Distributed	Distributed
北向接口	C++ API	REST API	RESTful API/Java API	RESTful API	**REST/RESTCONF/Java**
编程语言	C++/Python	Python	Java	Java	Java
管理	--	CLI	Web UI	Web GUI/CLI	Web GUI/CLI
南向接口	OpenFlow	Full	OpenFlow	Full	Full
OpenStack支持	No	Yes	Yes	--	Yes
OpenFlow支持	v1.0	**v1. {0,2,3,4,5 } OF++**	V1. {0,3 }	v1.0	v1. {0,3 }
一致性	No	No	No	Strong	Weak
容错能力	No	No	No	Yes	No
学习曲线	Medium	**Easy**	Medium	Medium	Hard
License	GPLv3	Apache 2.0	Apache	Apache 2.0	EPL v1.0
社区活跃度	28 / 6	2460 / 57	2511 / 53	4453 / 55	**6384 / 60**

多租户网络需求

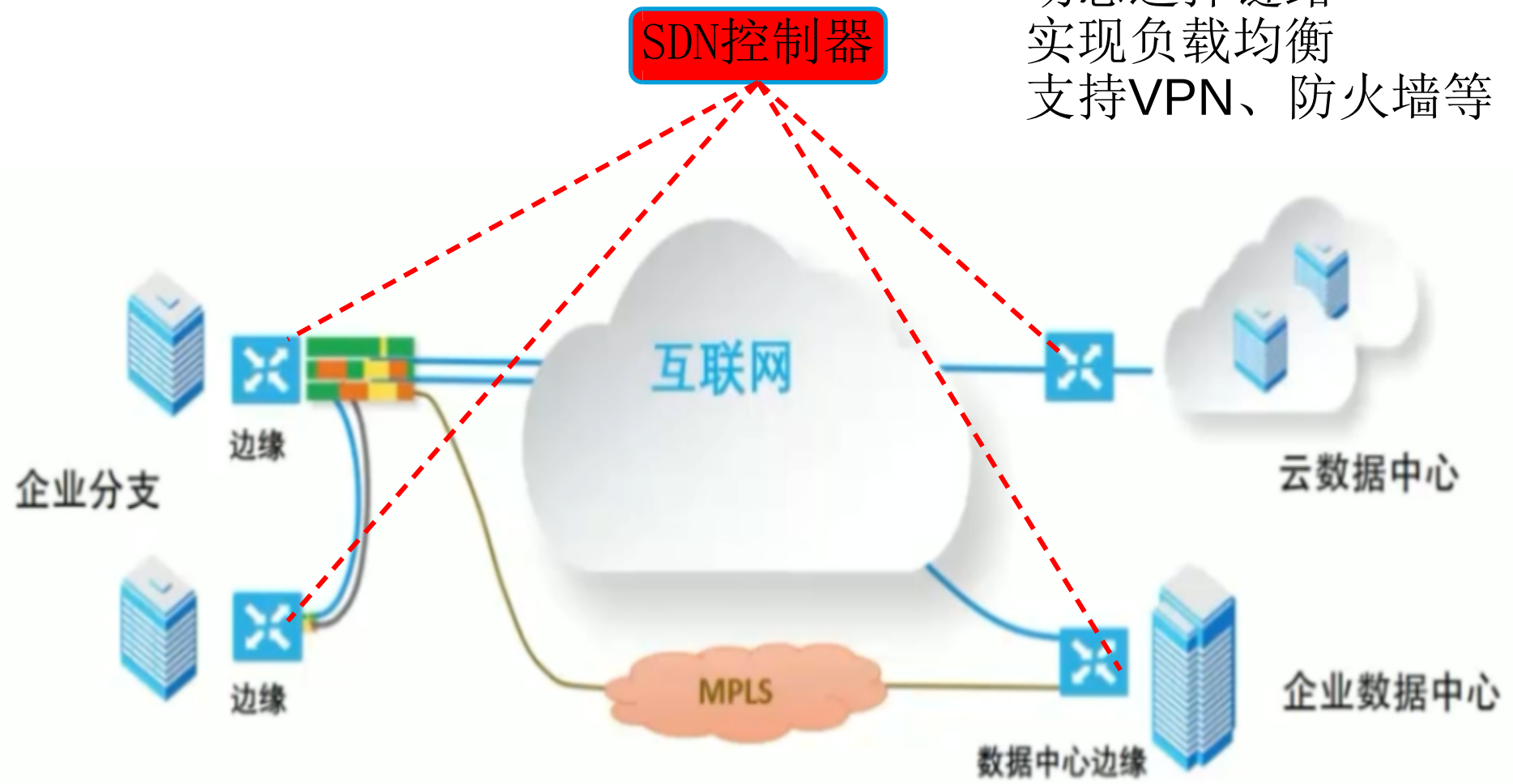


SDN在数据中心网络的应用

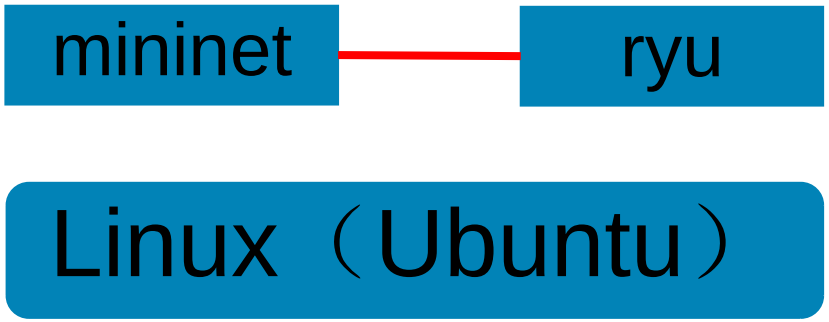


软件定义广域网——SDWAN

动态选择链路
实现负载均衡
支持VPN、防火墙等



实验环境搭建



mininet：网络仿真器
定义虚拟网络
网络测试

Ryu：SDN控制器
定义转发规则

Python

随机流量模拟

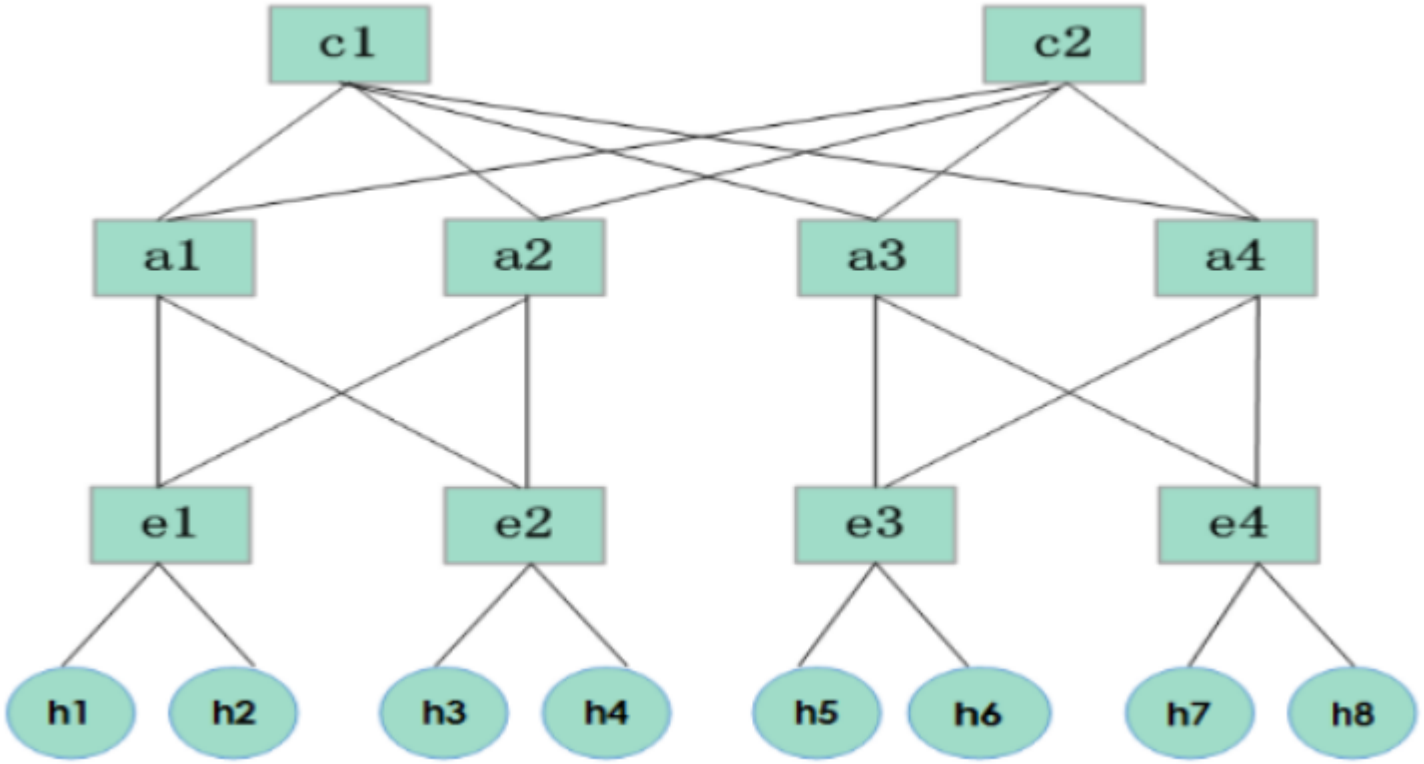
- 定义网络

核心交换

汇聚交换

边缘交换

主机



随机流量模拟

创建网络拓扑脚本 fattree.py

```
#!/usr/bin/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):
                Topo.__init__(self)
                L1 = 2
                L2 = L1 * 2
                L3 = L2
                c = []
                a = []
                e = []
```

随机流量模拟

创建网络拓扑脚本 fattree.py

```
for i in range( L1 ):
        sw=self.addSwitch('c{}'.format(i+1))
        c.append(sw)

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

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

随机流量模拟

创建网络拓扑脚本 fattree.py

```
for i in range( L1 ):
        sw1 = c[i]
        for sw2 in a[i/2::L1/2]:
                self.addLink(sw2,sw1)

for i in range(0, L2, 2):
        for  sw1 in a[i:i+2]:
                for sw2 in e[i:i+2]:
                        self.addLink(sw2,sw1)
```

随机流量模拟

创建网络拓扑脚本 fattree.py

```
        count =1
        for sw1 in e:
                for i in range(2):
                        host = self.addHost('h{}'.format( count ))
                        self.addLink( sw1,host)
                        count += 1

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

随机流量模拟

在net.py中添加自定义命令iperf_single

```
def iperf_single( self ,hosts=None, udpBw='10M', period=60, port=5001):
        if not hosts:
                return
        else:
                assert len(hosts) ==2
        client,server=hosts
        filename=client.name[1:]+'.out'
        output('*** Iperf:testing bandwidth between ')
        output("%s and %s\n" % (client.name, server.name))
        iperfArgs = 'iperf -u '
        bwArgs = '-b '+udpBw +' '
        print "*** start server ****"
        server.cmd( iperfArgs + '-s -i 1' + '>/home/wjs/log/' + filename + '&')
        print "***start client ***"
        client.cmd(
                iperfArgs + '-t ' +str(period) + ' -c '+server.IP() + ' ' +bwArgs
                +' > /home/wjs/log' + 'client' + filename +'&')
```

随机流量模拟

在net.py中添加自定义命令iperfMulti

```
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]

        cli_outs = []
        ser_outs = []
        _len = len(host_list)
        for i in xrange(0, _len):
                client = host_list[i]
                server = client
                while(server == client ):
                        server = random.choice(host_list)
                server_list.append(server)
                self.iperf_single(hosts = [client,server], udpBw=bw,
                                        period=period, port=base_port)
                sleep(.05)
                base_port +=1
        sleep(period)
        print "test has done"
```

随机流量模拟

在cli.py中注册自定义命令iperfMulti

```
def do_iperfmulti( self, line):
        args = line.split()
        if len(args) == 1:
                udpBw = args[ 0 ]
                self.mn.iperfMulti(udpBw)
        elif len(args) ==2:
                udpBw = args[ 0 ]
                period = args[ 1 ]
                err = False
                self.mn.iperfMulti(udpBw, float(period))
        else:
                error('invalid number of args: iperfmulti udpBw period\n' +
                                        'udpBw examples:1M 120\n')
```

随机流量模拟

在mn中加入可执行命令iperfmulti

```
TESTS = { name: True
        for name in ( 'pingall', 'pingpair', 'iperf', 'iperfudp','iperfmulti' ) }
CLI = None  # Set below if needed
# Locally defined tests
def allTest( net ):
    "Run ping and iperf tests"
    net.waitConnected()
    net.start()
    net.ping()
    net.iperf()
def nullTest( _net ):
    "Null 'test' (does nothing)"
    pass
TESTS.update( all=allTest, none=nullTest, build=nullTest )
# Map to alternate spellings of Mininet() methods
ALTSPELLING = { 'pingall': 'pingAll', 'pingpair': 'pingPair',
                'iperfudp': 'iperfUdp','iperfmulti':'iperfMulti' }
```

随机流量模拟

重新编译mininet

启动ryu控制器
#ryu-manager simple_switch_stp.py

启动mininet，生成测试拓扑结构
#mn –custom fattree.py --topo mytopo
　　--controller=remote,ip=127.0.0.1,port=6633

流量测试
iperf h1 h2
iperf h1 h3
iperf h1 h5
iperfmulti 0.025m

查看日志文件

Thank You!

2018年05月12日