

**一. 参照上图所示拓扑部署一个基于SDN的简单网络环境,南向接口采用 OpenFlow 1.3协议。(描述生成步骤,截屏相应命令,贴出生成topo的 python代码)**

**topo代码（my\_topo.py）：**

from mininet.topo import Topo

class Testtopo(Topo):

def \_\_init\_\_(self):

Topo.\_\_init\_\_(self)

S1=self.addSwitch('s1')

S2=self.addSwitch('s2')

H1=self.addHost('h1')

H2=self.addHost('h2')

H3=self.addHost('h3')

H4=self.addHost('h4')

self.addLink(S1,S2)

self.addLink(S1,H1)

self.addLink(S1,H2)

self.addLink(S2,H3)

self.addLink(S2,H4)

topos={

'topo':(lambda: Testtopo())

}

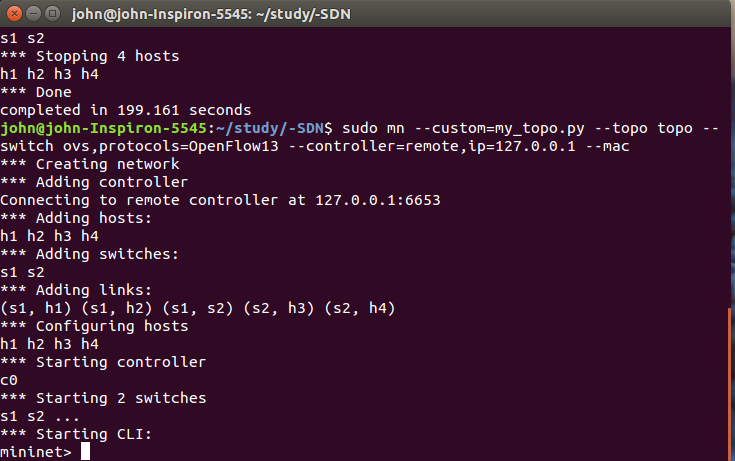
**mininet命令：**

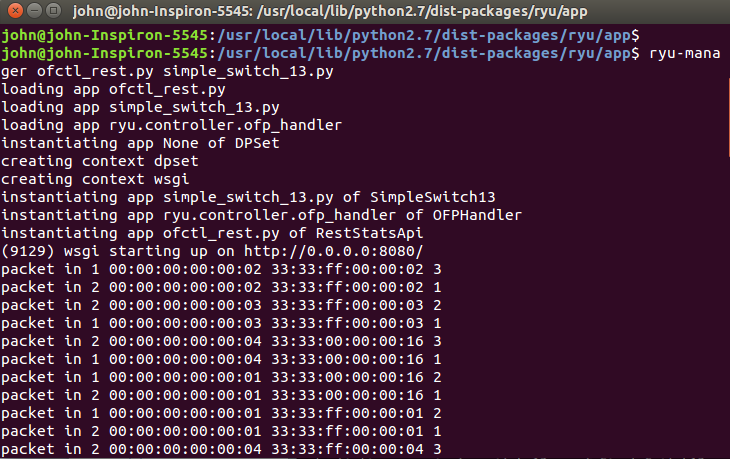
sudo mn --custom=my\_topo.py --topo topo --switch ovs,protocols=OpenFlow13 --controller=remote,ip=127.0.0.1 --mac

**Ryu命令：**

ryu-manager ofctl\_rest.py simple\_switch\_13.py

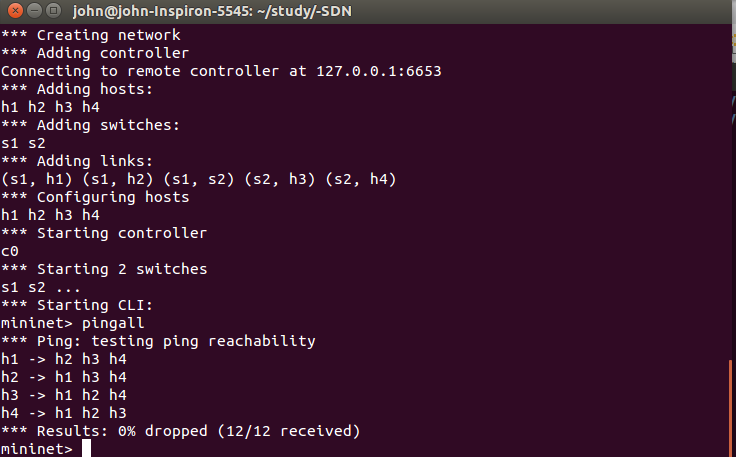
截图：





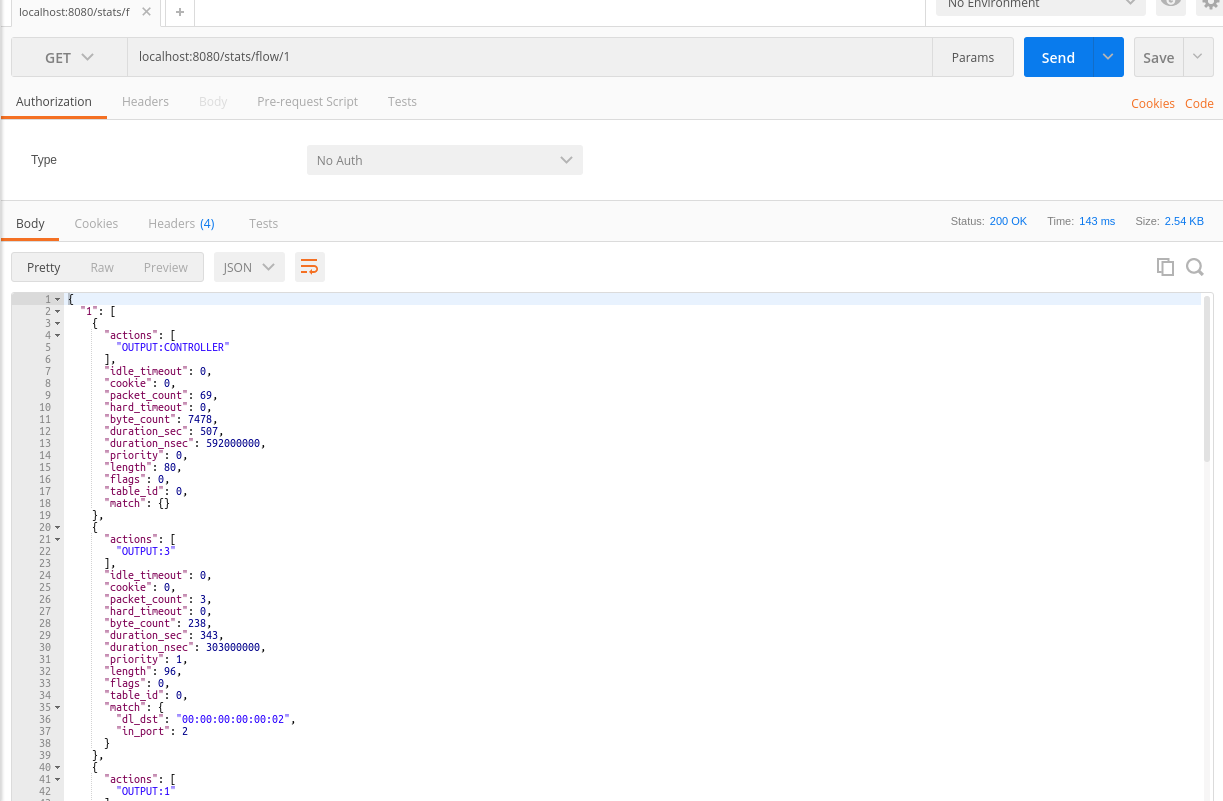
**二. H1、H2、H3、H4任意两两可互通。(给出4个节点两两互ping测试截图,每对只需ping二次即可)**

pingall:

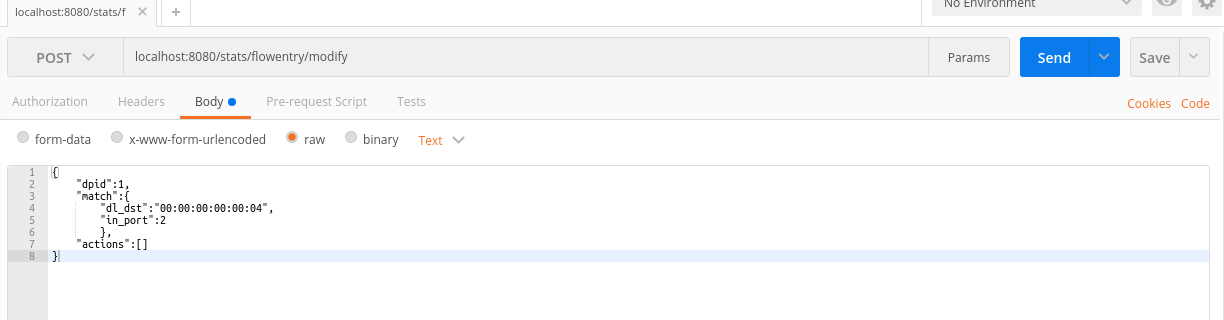


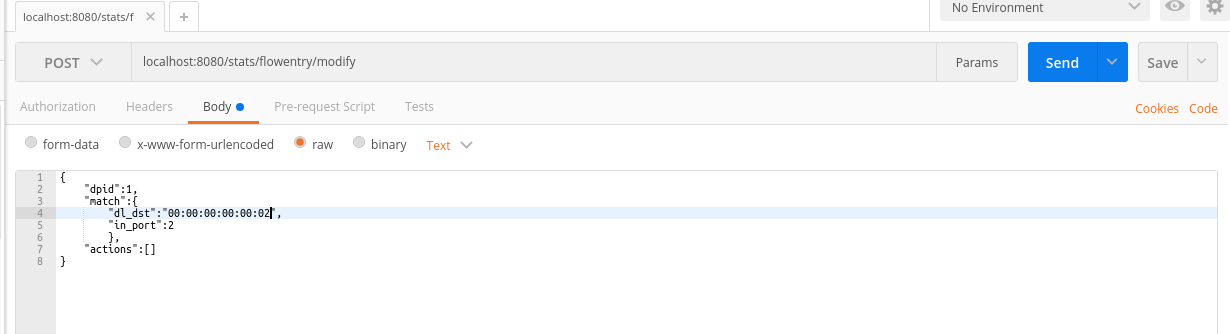
**三. 下发流表项实现H1和H2,H1和H4不能互通,并使得H1和H3可互通。(给出具体操作,查询到的流表信息和ping测试结果)**

交换机1的流表（localhost:8080/stats/flow/1）：

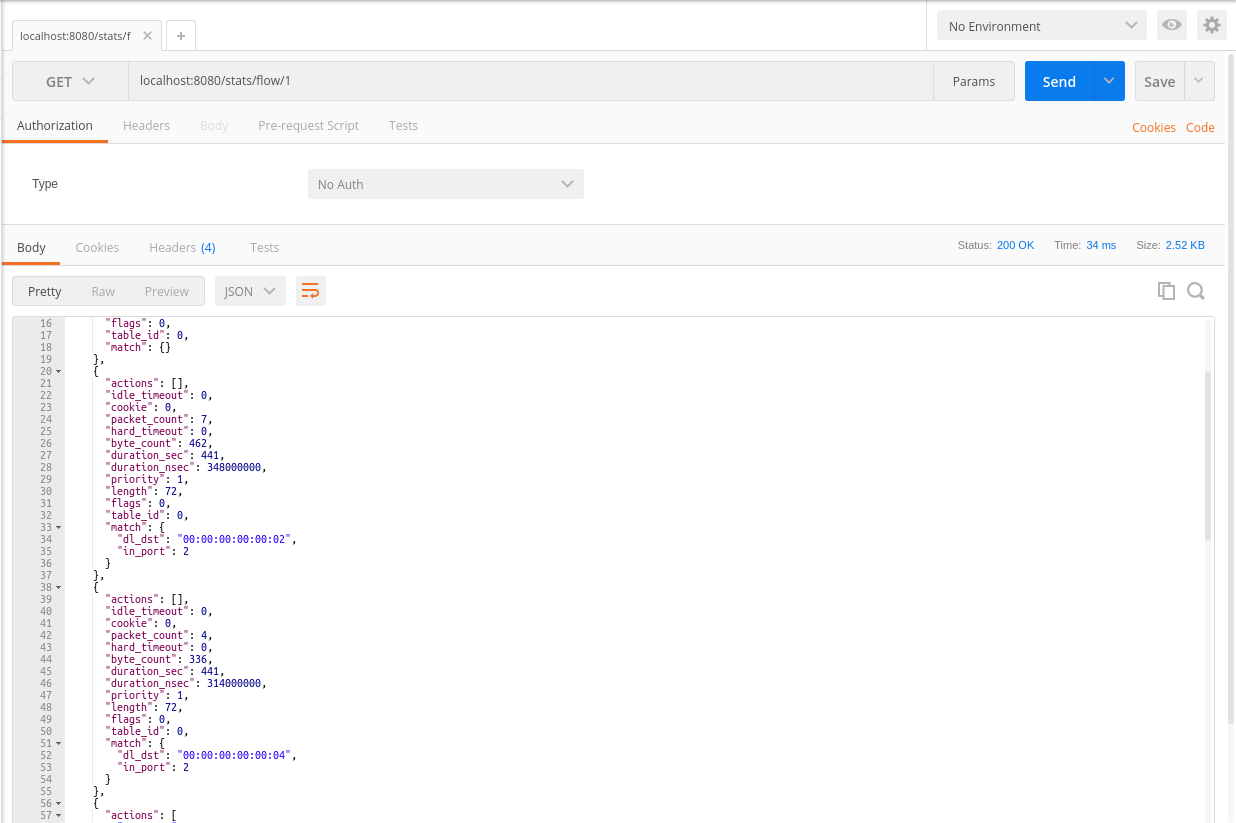


更改流表（localhost:8080/stats/flowentry/modify）：





更改后的流表（localhost:8080/stats/flow/1）：

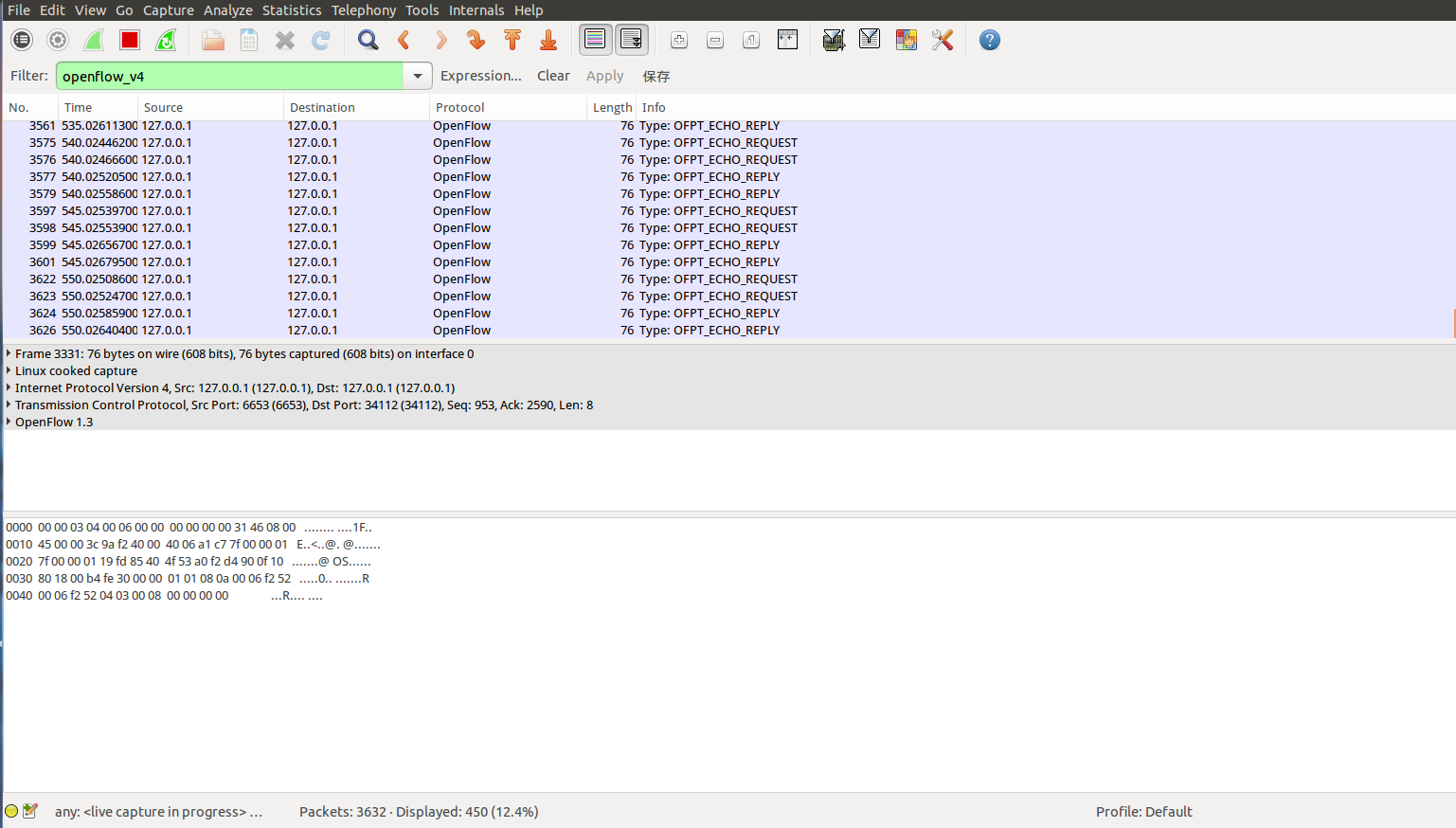


pingall：OFPT\_ECHO\_REQUEST

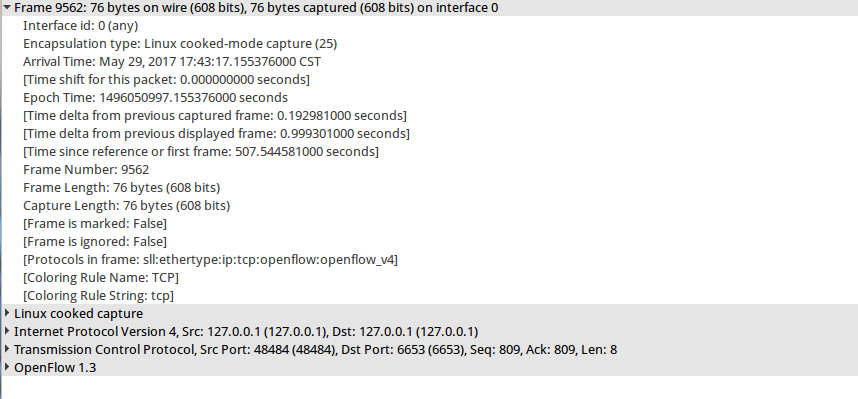


1. **结合捕获到的某些OpenFlow协议报文,分析其报文结构,并简要描述该类报文作用。(一种即可,限300字)**

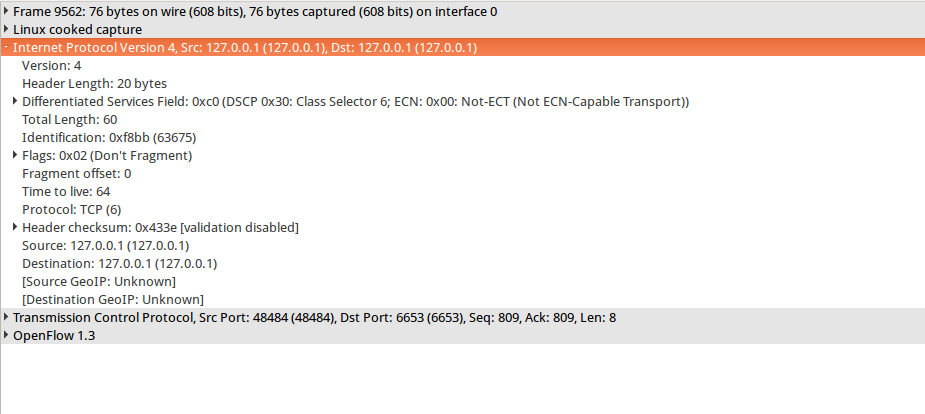
使用wireshark 抓取openflow协议报文（sudo wireshark）：



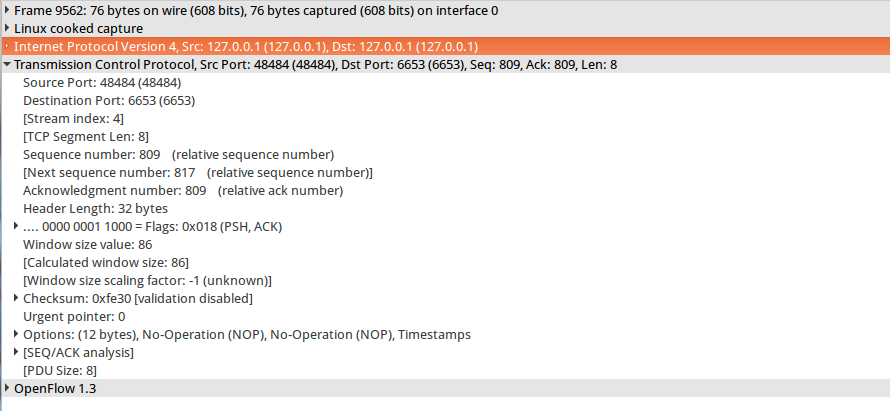
Openflow1.3 报文：



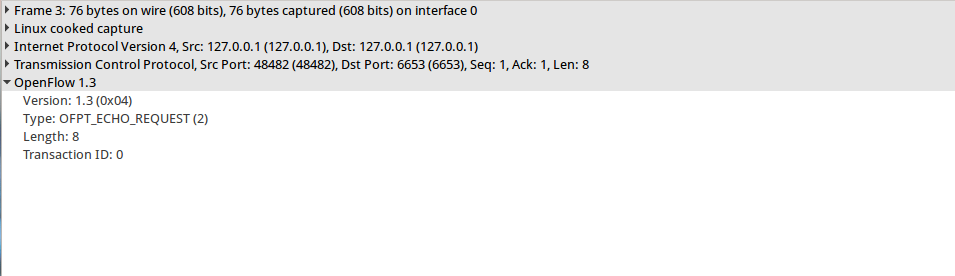
上图中的Frame就是数据链路层的帧。其中，包含着的信息有interface，arrival time，epoch time，帧号（frame number），帧长度（frame length），捕获时间（capture time），协议（TCP）等。



上图中展开的是Internet协议，其中包含信息有源IP，目的IP，协议（TCP），Header Length等。



上图中展开部分是Transmission Control Protocol。其中包含信息有源端口号，目的端口号，Sequence number，Ack number，Header Length等。



上图中展开的Openflow1.3列表中version：1.3表示openflow版本； Type：OPFT\_ECHO\_RESQUEST(2)表示消息类型是OFTP\_ECHO\_RESQUEST(2)；Length：8 表示报文长度；Transaction：0 表示报文类型编号。

这条openflow1.3的报文类型是OPFT\_ECHO\_RESQUEST(2)。它的作用是当没有其他的数据包进行交换时，controller会定期的循环给sw发送OFPT\_ECHO\_REQUEST，以查询连接状态，确保连接通畅。

OFPT\_ECHO 分为 OFPT\_ECHO\_REQUEST 和 OFPT\_ECHO\_REPLY 两种对称信息，此次实验分析的是OFPT\_ECHO\_REQUEST 类型的信息，是在没有数据包交换的情况下，controller向交换机发送的。

1. **思考在同样网络拓扑下如何在传统网络中实现要求3,并分析与SDN实现之间的差别。**

在同样的网络拓扑下，可以通过基于VLAN（虚拟局域网，Virtual Local Area Network）技术的方法实现。

VLAN是对连接到的第二层交换机端口的网络用户的逻辑分段，不受网络用户的物理位置限制而根据用户需求进行网络分段。一个VLAN可以在一个交换机或者跨交换机实现。VLAN可以根据网络用户的位置、作用、部门或者根据网络用户所使用的应用程序和协议来进行分组。

使用VLAN只需要将h1与h2，h4置于不同分组中，而与h3置于相同分组中即可实现要求三中，h1与h2，h4 ping不同，与h3可以ping通。而SDN实现要求三是通过更改交换机的流表项，从而改变网络功能。SDN（软件定义网络，Software Defined Network）是通过openflow技术，将网络设备控制面与数据面分离开来，从而实现网络流量的灵活性，SDN无须依赖底层网络设备（路由器、交换机、防火墙），屏蔽了来自底层网络设备的差异。而控制权是完全开放的，用户可以自定义任何想实现的网络路由和传输规则策略，从而更加灵活和智能。而VLAN是对连接到第二层交换机端口的网络用户进行逻辑分段，不受网络用户的物理位置限制，而根据用户需求进行分段。

VLAN与SDN的实现差别是VLAN只局限于同一二层网络，而SDN技术可以消除传统的二层网络限制。VLAN在数量上也会影响多租户业务规模，而SDN可以利用叠加在三层网络之上的虚拟网络传递二层数据包。