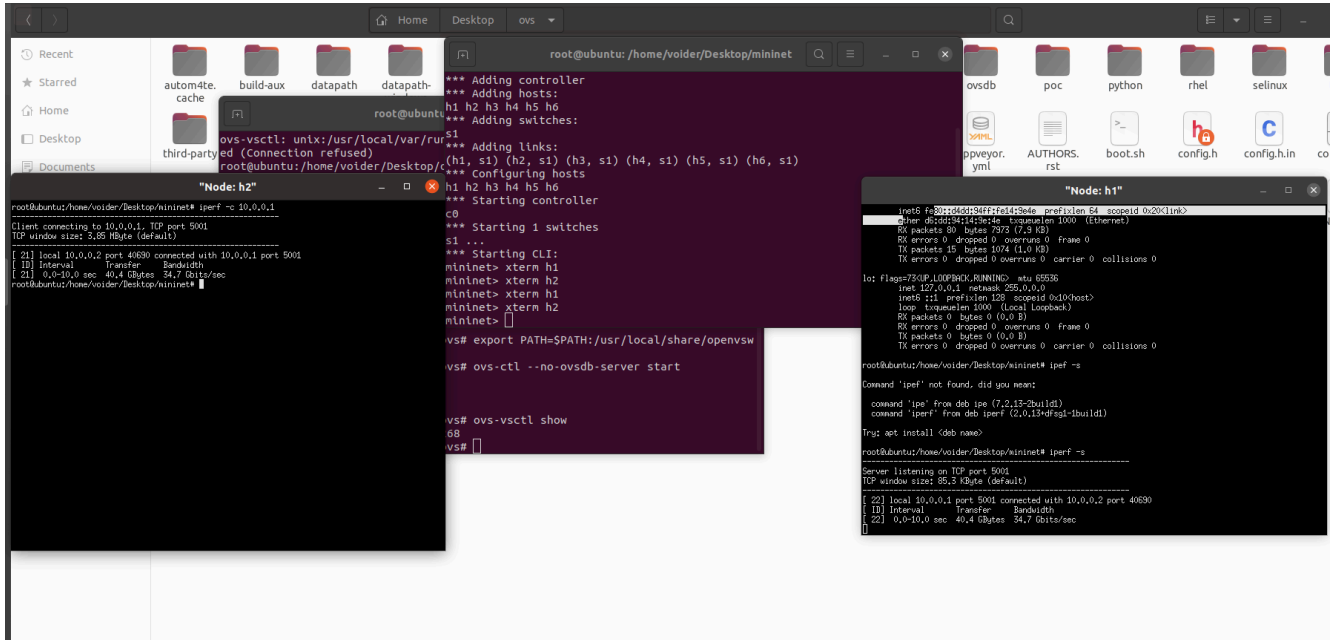


# Lab3

- Part1 搭建网络拓扑
  - 连通性测试
  - 如图



```
root@ubuntu: /home/volder/Desktop/mininet
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> xterm h1
mininet> xterm h2
mininet> xterm h2
mininet>

vs# export PATH=SPATH:/usr/local/share/openvsw
vs# ovs-ctl --no-ovsdb-server start

vs# ovs-vsctl show
vs#
```

```
root@ubuntu:/home/volder/Desktop/mininet# iperf -c 10.0.0.1
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 3,05 MByte (default)
[ 21] local 10.0.0.2 port 40690 connected with 10.0.0.1 port 5001
[ 10] interval      transfer  Bandwidth
[ 21] 0.0-10.0 sec  40.4 Gbytes  34.7 Gbits/sec
root@ubuntu:/home/volder/Desktop/mininet#
```

```
root@ubuntu:/home/volder/Desktop/mininet# iperf -s
Server listening on TCP port 5001
TCP window size: 65.5 KByte (default)
[ 22] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 40690
[ 10] interval      transfer  Bandwidth
[ 22] 0.0-10.0 sec  40.4 Gbytes  34.7 Gbits/sec
root@ubuntu:/home/volder/Desktop/mininet#
```

使用ifconfig获得node1 IP地址, 然后node2向node1发包, 进行连通性测试, 通信成功

o

```
Mar 21 00:07
"Node: h2"
root@ubuntu:/home/voider/Desktop/mininet# iperf -c 10.0.0.1
-----
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 3.85 MByte (default)
-----
[ 21] local 10.0.0.2 port 40690 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer    Bandwidth
[ 21] 0.0-10.0 sec  40.4 GBytes 34.7 Gbits/sec
root@ubuntu:/home/voider/Desktop/mininet#
```

o

```
Mar 21 00:07

"Node: h1"

root@ubuntu:/home/voider/Desktop/mininet# ifconfig
1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::d4dd:94ff:fe14:9e4e prefixlen 64 scopeid 0x20<link>
    ether d6:dd:94:14:9e:4e txqueuelen 1000 (Ethernet)
    RX packets 80 bytes 7973 (7.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15 bytes 1074 (1.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@ubuntu:/home/voider/Desktop/mininet# iperf -s

command 'iperf' not found, did you mean:

  command 'ipe' from deb ipe (7.2.13-2build1)
  command 'iperf' from deb iperf (2.0.13+dfsg1-1build1)

try: apt install <deb name>

root@ubuntu:/home/voider/Desktop/mininet# iperf -s

server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

22] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 40690
ID] Interval      Transfer      Bandwidth
22] 0.0-10.0 sec  40.4 GBytes  34.7 Gbits/sec
```

- part2 准入控制

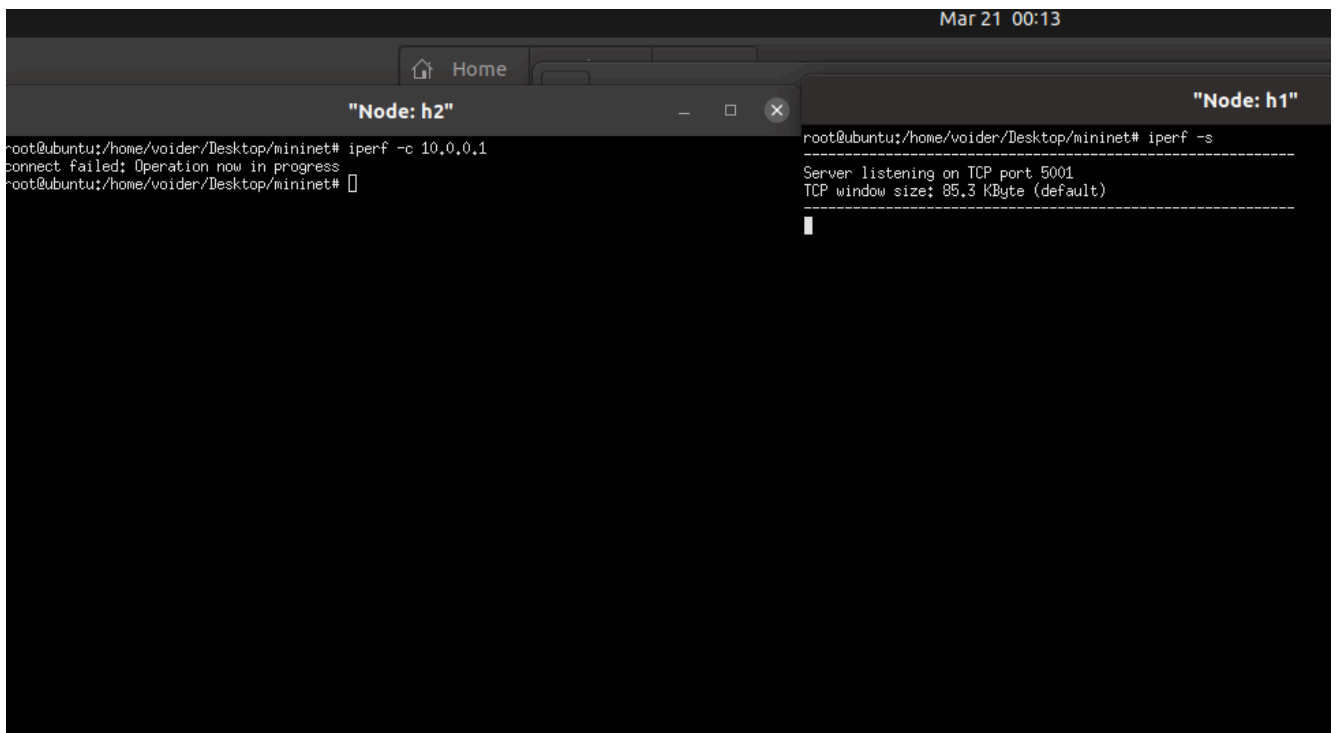
- 建立准入规则

```
Mar 21 00:12

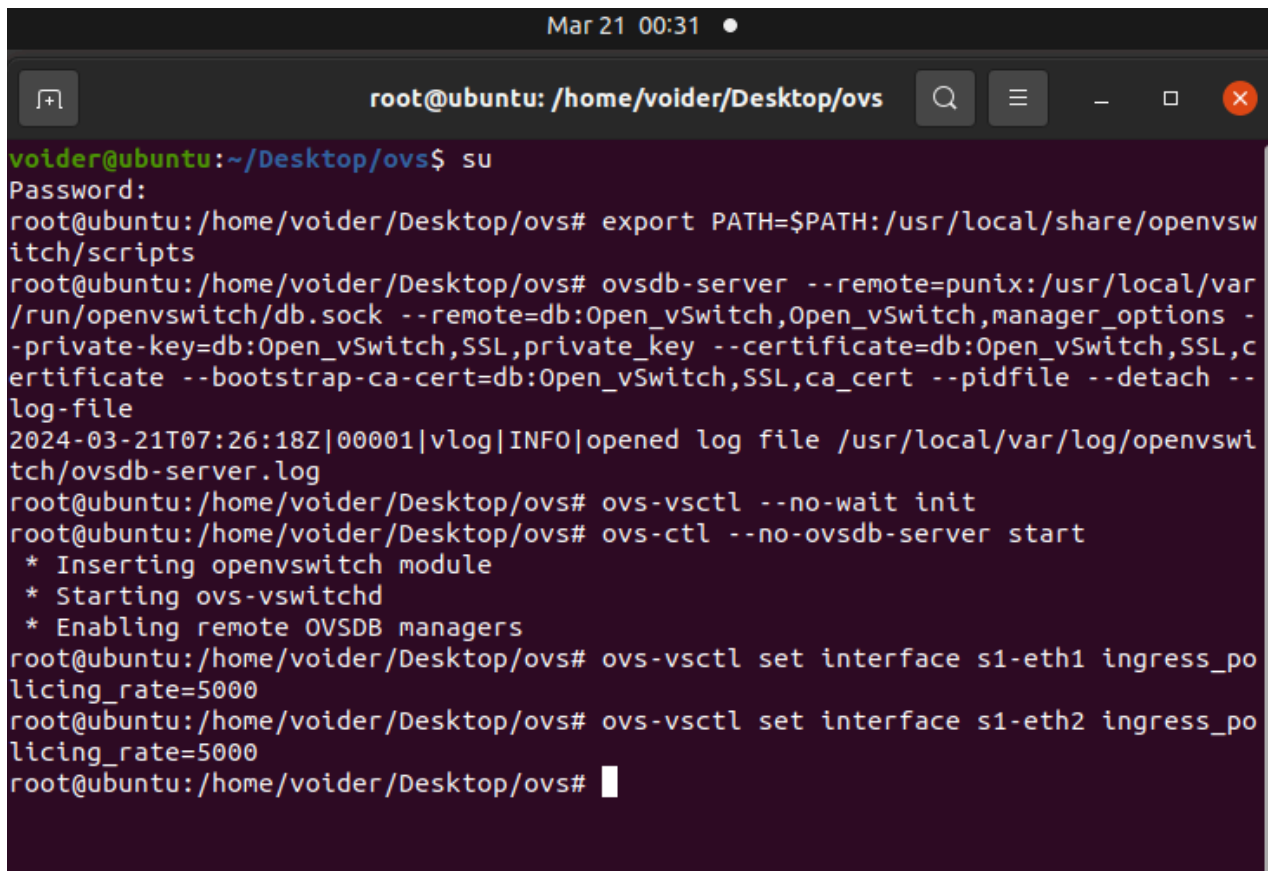
root@ubuntu: /home/voider/Desktop/ovs

root@ubuntu:/home/voider/Desktop/ovs# ovs-ofctl dump-flows s1 -O openflow13
cookie=0x0, duration=897.684s, table=0, n_packets=73, n_bytes=5234, priority=0 actions=CONTROLLER:128
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl set bridge s1 datapath_type=netdev
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl set bridge s1 protocols=OpenFlow13
root@ubuntu:/home/voider/Desktop/ovs# ovs-ofctl add-flow s1 "table=0,nw_src=10.0.0.2/24,actions=drop" -O openflow13
2024-03-21T07:11:00Z|00001|ofp_match|INFO|normalization changed ofp_match, details:
2024-03-21T07:11:00Z|00002|ofp_match|INFO| pre: nw_src=10.0.0.0/24
2024-03-21T07:11:00Z|00003|ofp_match|INFO|post:
root@ubuntu:/home/voider/Desktop/ovs# ovs-ofctl dump-flows s1 -O openflow13
cookie=0x0, duration=13.730s, table=0, n_packets=0, n_bytes=0, actions=drop
cookie=0x0, duration=43.871s, table=0, n_packets=1, n_bytes=70, priority=0 actions=CONTROLLER:128
root@ubuntu:/home/voider/Desktop/ovs#
```

- node2向node1发包, 被拒收(包全部被丢弃)



- part3 三种限速方式
  - 网卡限速(Linux内核中接收数据包使用的方法叫策略(policing)用于限制网卡上接收分组(ingress)的速率，当速率超过了配置速率，就简单的把数据包丢弃。)
  - 使用ovs对node1-2网卡进行限速



- node2向node1发包后的结果

```
Mar 21 00:35
"Node: h1"
root@ubuntu:/home/voider/Desktop/mininet#
root@ubuntu:/home/voider/Desktop/mininet# iperf -u -s
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 21] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 58800
[ ID] Interval      Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[ 21] 0.0-10.2 sec  6.81 MBytes  5.57 Mbits/sec  15.670 ms 4063/ 8918 (46%)
"Node: h2"
root@ubuntu:/home/voider/Desktop/mininet# iperf -u -c 10.0.0.1 -b 10M
Client connecting to 10.0.0.1, UDP port 5001
Sending 1470 byte datagrams, IPC target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 21] local 10.0.0.2 port 58800 connected with 10.0.0.1 port 5001
[ ID] Interval      Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[ 21] 0.0-10.0 sec  12.5 MBytes  10.5 Mbits/sec
[ 21] Sent 8917 datagrams
[ 21] Server Report:
[ 21] 0.0-10.2 sec  6.81 MBytes  5.57 Mbits/sec  15.670 ms 4063/ 8918 (46%)
root@ubuntu:/home/voider/Desktop/mininet#
```

▪ 结果:

发送/收到(bytes)	丢包率(%)	Jitter(ms)	发送/接收带宽(Mbits/sec)
12.5M/6.81M	4063/ 8918 (46%)	15.670	10.5/5.57

◦ 队列限速(Linux可以将网络数据包缓存起来,然后根据用户的设置,在尽量不中断连接(如tcp)的前提下平滑网络流量。内核通过某个网络接口发送数据包,它都需要按照这个接口的队列规则把数据包加入队列)

▪ 为node4网卡创建队列, 指定最大速率

```
Mar 21 01:12
root@ubuntu: /home/voider/Desktop/ovs
voider@ubuntu: ~/Desktop/mininet
s1-eth6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::b876:27ff:fe2d:d779 prefixlen 64 scopeid 0x20<link>
    ether ba:76:27:2d:d7:79 txqueuelen 1000 (Ethernet)
    RX packets 9  bytes 726 (726.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 52  bytes 5033 (5.0 KB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

voider@ubuntu:~/Desktop/mininet$ ovs-vsctl set port s1-eth4 qos=@newqos -- --id=@newqos create
qos type=linux-htb queues=0=@q0 -- --id=@q0 create queue other-config:max-rate=5000000
ovs-vsctl: unix:/usr/local/var/run/openvswitch/db.sock: database connection failed (Permission
denied)
voider@ubuntu:~/Desktop/mininet$ sudo ovs-vsctl set port s1-eth4 qos=@newqos -- --id=@newqos c
reate qos type=linux-htb queues=0=@q0 -- --id=@q0 create queue other-config:max-rate=5000000
[sudo] password for voider:
034fcc19-ab3d-4d50-930d-1ffc72c0b003
388bcd7-d9aa-4cb6-9485-07bba4687d32
voider@ubuntu:~/Desktop/mininet$ ovs-vsctl list qos
ovs-vsctl: unix:/usr/local/var/run/openvswitch/db.sock: database connection failed (Permission
denied)
voider@ubuntu:~/Desktop/mininet$ sudo ovs-vsctl list qos
 _uuid          : 034fcc19-ab3d-4d50-930d-1ffc72c0b003
external_ids    : {}
other_config    : {}
queues          : {0=388bcd7-d9aa-4cb6-9485-07bba4687d32}
type            : linux-htb
voider@ubuntu:~/Desktop/mininet$ sudo ovs-vsctl list queue
 _uuid          : 388bcd7-d9aa-4cb6-9485-07bba4687d32
dscp            : []
external_ids    : {}
other_config    : {max-rate="5000000"}
```

▪ node3向node4发送包

```

XTerm Mar 21 01:15
"Node: h3"
root@ubuntu:/home/voider/Desktop/mininet# iperf -u -c 10.0.0.4 -b 10M
Client connecting to 10.0.0.4, UDP port 5001
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)
-----
[ 21] local 10.0.0.3 port 60551 connected with 10.0.0.4 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 21] 0.0-10.1 sec  5.90 MBytes  4.92 Mbits/sec
[ 21] Sent 4210 datagrams
[ 21] Server Report:
[ 21] 0.0-10.2 sec  5.90 MBytes  4.86 Mbits/sec    9.543 ms    0/ 4210 (0%)
root@ubuntu:/home/voider/Desktop/mininet#

"Node: h4"
RX packets 92 bytes 8047 (8.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 14 bytes 1076 (1.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@ubuntu:/home/voider/Desktop/mininet# iperf -u -s
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[ 21] local 10.0.0.4 port 5001 connected with 10.0.0.3 port 60551
[ ID] Interval      Transfer      Bandwidth      Jitter    Lost/Total Datagrams
[ 21] 0.0-10.2 sec  5.90 MBytes  4.86 Mbits/sec  9.544 ms    0/ 4210 (0%)

```

▪ 结果

发送/接收(bytes)	丢包率(%)	Jitter(ms)	发送/接收带宽(Mbbits/sec)
5.90M/5.90M	0/4210 (0%)	9.544ms	4.92/4.86

○ Meter表限速

▪ 先按要求进行配置

```

Mar 21 01:27
root@ubuntu: /home/voider/Desktop/ovs
speed: 10000 Mbps now, 0 Mbps max
2(s1-eth2): addr:8e:ea:2d:bf:17:c4
config: 0
state: LIVE
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
3(s1-eth3): addr:42:54:17:b1:3d:04
config: 0
state: LIVE
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
4(s1-eth4): addr:3a:87:e7:0c:ee:9f
config: 0
state: LIVE
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
5(s1-eth5): addr:66:8f:bf:16:54:52
config: 0
state: LIVE
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
6(s1-eth6): addr:ba:76:27:2d:d7:79
config: 0
state: LIVE
current: 10GB-FD COPPER
speed: 10000 Mbps now, 0 Mbps max
LOCAL(s1): addr:06:fb:98:d4:8f:44
config: PORT_DOWN
state: LINK_DOWN
current: 10MB-FD COPPER
speed: 10 Mbps now, 0 Mbps max
OFP1_GET_CONFIG_REPLY (OF1.3) (xid=0x9): frags=normal miss_send_len=0
root@ubuntu:/home/voider/Desktop/ovs# ovs-ofctl add-flow s1 in_port=5,action=meter:1,output:6 -O openflow13
root@ubuntu:/home/voider/Desktop/ovs# ovs-ofctl dump-flows s1 -O openflow13
cookie=0x0, duration=8.778s, table=0, n_packets=0, n_bytes=0, in_port="s1-eth5" actions=meter:1,output:"s1-eth6"
cookie=0x0, duration=58.634s, table=0, n_packets=0, n_bytes=0, priority=0 actions=CONTROLLER:128
root@ubuntu:/home/voider/Desktop/ovs#

```

▪ question1 尝试理解Line19,20两条指令，指出每条指令的具体工作是

# 什么，并逐个分析其中各个参数的具体含义

- 第一条指令向交换机s1中添加流表项
  - add-flow s1 在s1中添加
  - in\_port=5 表示这个流表项匹配进入s1的第5端口(s1-eth5)的数据包
  - action=meter:1 指定匹配到的数据包应用Meter1的规则
  - output:6 匹配到的数据包将被发送到交换机的第6端口(s1-eth6)
  - O openflow13: 指定使用 OpenFlow 1.3版本协议进行操作
- 第二条用于显示交换机s1上的所有流表项
  - dump-flows 用于导出交换机上的流表项
  - s1 指定要查看的交换机的名称
  - O openflow13 指定使用OpenFlow 1.3版本协议进行操作
- 发包测试

```
root@ubuntu: /home/voider/Desktop/ovs
"Node: h5"
h5-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.5 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::1963:feff:fe25:4593 prefixlen 64 scopeid 0x20<link>
    ether 82:e5:fe:3a:3c:3 txqueuelen 1000 (Ethernet)
    RX packets 104 bytes 8897 (8.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15 bytes 1146 (1.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@ubuntu:/home/voider/Desktop/mininet# ethtool -K h5-eth0 tx off
Actual changes:
tx-checksumming: off
tx-checksum-ip-generic: off
tx-checksum-udp: off
tcp-segmentation-offload: off
    tx-tcp-segmentation: off [requested on]
    tx-tcp-receive-segmentation: off [requested on]
    tx-tcp-merge-segmentation: off [requested on]
    tx-tcp6-segmentation: off [requested on]
root@ubuntu:/home/voider/Desktop/mininet# iperf -u -c 10.0.0.6 -b 10M
Client connecting to 10.0.0.6, UDP port 5001
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size: 208 KByte (default)

[ 21] local 10.0.0.5 port 43601 connected with 10.0.0.6 port 43601
[ 21] WARNING: did not receive ack of last datagram after 10 tries.
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[ 21] 0.0-10.0 sec 12.5 MBytes 5.35 Mbits/sec 0.026 ms 4371/ 8917 (49%)
root@ubuntu:/home/voider/Desktop/mininet#

"Node: h6"
root@ubuntu:/home/voider/Desktop/mininet# ifconfig
h6-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.6 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::1b6e:45ff:fe4:d5ce prefixlen 64 scopeid 0x20<link>
    ether ba:6e:45:e4:d5:ce txqueuelen 1000 (Ethernet)
    RX packets 105 bytes 8893 (8.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 1216 (1.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@ubuntu:/home/voider/Desktop/mininet# iperf -u -s
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 21] local 10.0.0.6 port 5001 connected with 10.0.0.5 port 43601
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[ 21] 0.0-10.0 sec 6.37 MBytes 5.35 Mbits/sec 0.026 ms 4371/ 8917 (49%)
```

发送/接收(bytes)	丢包率(%)	Jitter(ms)	发送/接收带宽(Mbits/Sec)
12.5M/6.37M	4371/8917 (49%)	0.026	10.5/5.35

- Question2到这里，你已经完成了三种限速方式的实验，并获得了三组测试数据，请你就三组数据中的带宽、抖动和丢包率等参数，对三种限速方式进行横向比较，并适当地分析原因
  - 接收端数据对比

限速方式	接收带宽(Mbits/Sec)	丢包率(%)	Jitter(ms)
网卡限速	5.57	46%	15.670
队列限速	4.86	0	9.544



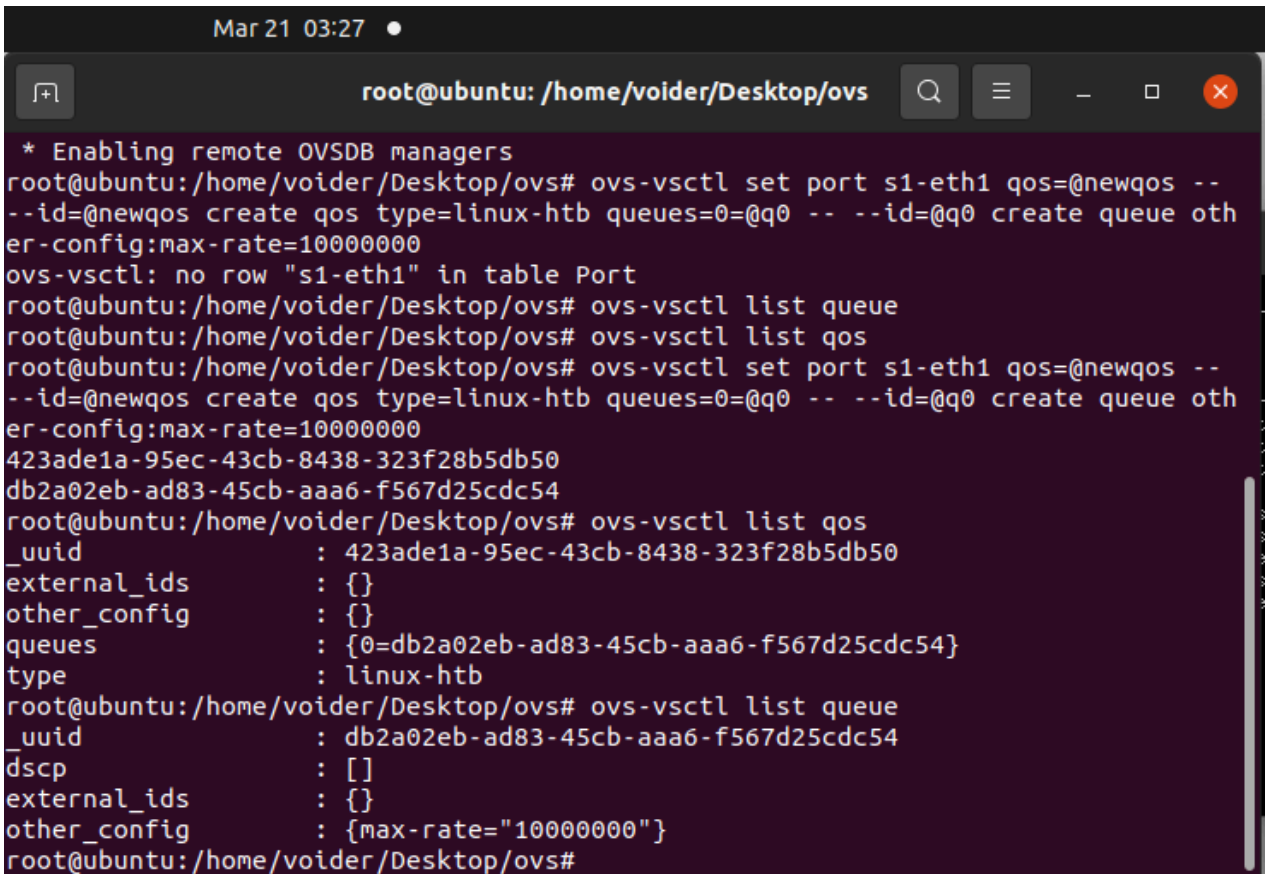
限速方式	接收带宽(Mbits/Sec)	丢包率(%)	Jitter(ms)
Meter表限速	5.35	49%	0.026

- 队列限速带宽限制效果好于网卡限速和Meter表限速: 队列限制确保数据包以有序和可预测的方式进行处理和传输——在发送端就已经做好了限制, 网卡限速与Meter表限速对于突发流量可能会出现带宽超出限制的情况
- 由于维护数据缓存, 队列限速会把来不及发送的数据缓存在队列里, 不会简单地丢弃, 另外两种策略都会丢弃数据包
- Meter表限速抖动最小: Meter表可以根据实时流量情况调整其策略, 以更好地适应网络负载的变化。网卡在硬件上固定速率, 方式粗放; 队列限制也会出现队列拥塞的情况, 或者由于调度的问题, 产生发送速率的波动。

#### • Part4 拓展与应用

##### ◦ 场景模拟

- 为s1-eth1队列限速10Mbits/sec



```

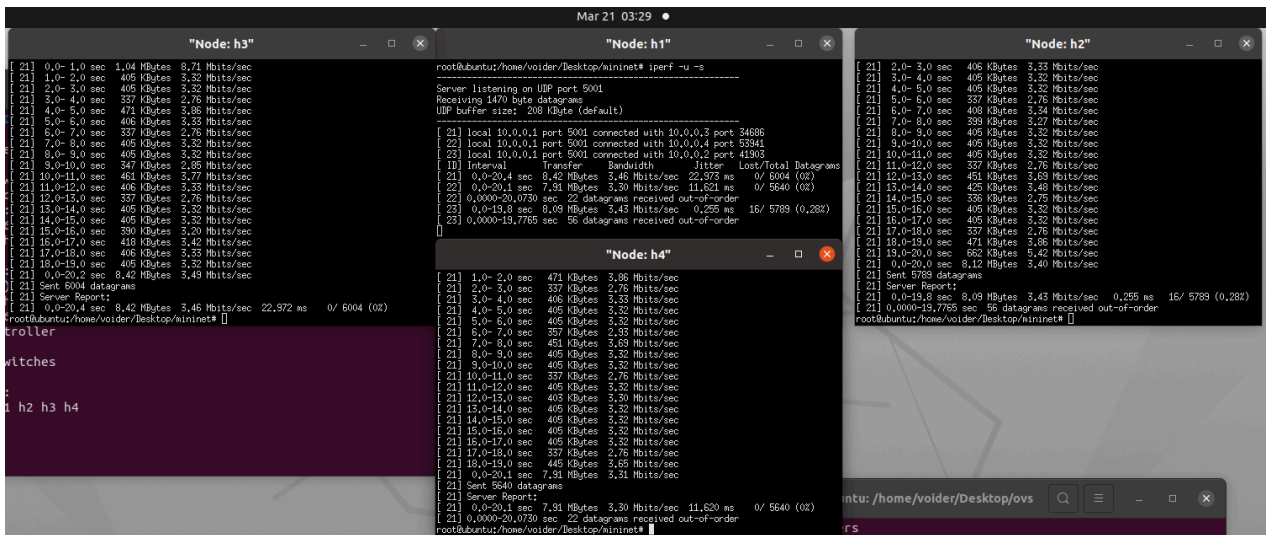
Mar 21 03:27
root@ubuntu: /home/voider/Desktop/ovs

* Enabling remote OVSDB managers
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl set port s1-eth1 qos=@newqos --
--id=@newqos create qos type=linux-htb queues=0=@q0 -- --id=@q0 create queue oth
er-config:max-rate=10000000
ovs-vsctl: no row "s1-eth1" in table Port
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl list queue
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl list qos
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl set port s1-eth1 qos=@newqos --
--id=@newqos create qos type=linux-htb queues=0=@q0 -- --id=@q0 create queue oth
er-config:max-rate=10000000
423ade1a-95ec-43cb-8438-323f28b5db50
db2a02eb-ad83-45cb-aaa6-f567d25cdc54
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl list qos
 _uuid          : 423ade1a-95ec-43cb-8438-323f28b5db50
external_ids    : {}
other_config    : {}
queues         : {0=db2a02eb-ad83-45cb-aaa6-f567d25cdc54}
type            : linux-htb
root@ubuntu:/home/voider/Desktop/ovs# ovs-vsctl list queue
 _uuid          : db2a02eb-ad83-45cb-aaa6-f567d25cdc54
dscp            : []
external_ids    : {}
other_config    : {max-rate="10000000"}
root@ubuntu:/home/voider/Desktop/ovs#

```

- node2-4同时发送 稳定后平均带宽为3.43 3.46 3.30 Mbits/sec





- 三者速率几乎相同, 可见三者是平等关系, 且三者带宽之和约等于node1限制的带宽——node1(Server)带宽被完全占用, 平均分配给三个client

#### ◦ Qos设计

- 利用队列设置中的min-rate保证node2-3的速率 同时也使用max-rate选项保证node4带宽尽量多
- 首先设置node1最大接收带宽为10Mbits/sec 然后创建两个限速队列 通过流表配置到node2-3
- 运行指令
- 

```
ovs-vsctl set port s1-eth1 qos=@newqos -- --id=@newqos create qos \
type=linux-htb other-config:max-rate=10000000 \
other-config:min-rate=10000000 queues=1=@q1,2=@q2 -- \
--id=@q1 create queue other-config:min-rate=5100000 \
other-config:max-rate=5400000 -- \
--id=@q2 create queue other-config:min-rate=3100000 \
other-config:max-rate=3400000

ovs-ofctl add-flow s1 in_port=2,action=set_queue:1,output:1 -O openflow13
ovs-ofctl add-flow s1 in_port=3,action=set_queue:2,output:1 -O openflow13
// 将两个最小值设置比5M 3M稍大 如果正好为这两个值的话会出现不能保证带宽的情况
```

- 运行结果 能够满足要求

## "Node: h4"

```

[ 21] 0.0- 1.0 sec 446 KBytes 3.66 Mbits/sec
[ 21] 1.0- 2.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 2.0- 3.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 3.0- 4.0 sec 141 KBytes 1.15 Mbits/sec
[ 21] 4.0- 5.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 5.0- 6.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 6.0- 7.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 7.0- 8.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 8.0- 9.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 9.0-10.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 10.0-11.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 11.0-12.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 12.0-13.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 13.0-14.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 14.0-15.0 sec 141 KBytes 1.15 Mbits/sec
[ 21] 15.0-16.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 16.0-17.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 17.0-18.0 sec 142 KBytes 1.16 Mbits/sec
[ 21] 18.0-19.0 sec 144 KBytes 1.18 Mbits/sec
[ 21] 0.0-20.1 sec 3.24 MBytes 1.35 Mbits/sec
[ 21] Sent 2310 datagrams
[ 21] Server Report:
[ 21] 0.0-20.2 sec 3.24 MBytes 1.35 Mbits/sec 7.172 ms 0/ 2310 (0%)
root@ubuntu:/home/voider/Desktop/mininet#

```

## "Node: h3"

```

[ 21] 0.0- 1.0 sec 500 KBytes 4.09 Mbits/sec
[ 21] 1.0- 2.0 sec 434 KBytes 3.55 Mbits/sec
[ 21] 2.0- 3.0 sec 368 KBytes 3.01 Mbits/sec
[ 21] 3.0- 4.0 sec 396 KBytes 3.25 Mbits/sec
[ 21] 4.0- 5.0 sec 435 KBytes 3.56 Mbits/sec
[ 21] 5.0- 6.0 sec 368 KBytes 3.01 Mbits/sec
[ 21] 6.0- 7.0 sec 462 KBytes 3.79 Mbits/sec
[ 21] 7.0- 8.0 sec 366 KBytes 3.00 Mbits/sec
[ 21] 8.0- 9.0 sec 393 KBytes 3.22 Mbits/sec
[ 21] 9.0-10.0 sec 438 KBytes 3.59 Mbits/sec
[ 21] 10.0-11.0 sec 366 KBytes 3.00 Mbits/sec
[ 21] 11.0-12.0 sec 438 KBytes 3.59 Mbits/sec
[ 21] 12.0-13.0 sec 392 KBytes 3.21 Mbits/sec
[ 21] 13.0-14.0 sec 366 KBytes 3.00 Mbits/sec
[ 21] 14.0-15.0 sec 464 KBytes 3.80 Mbits/sec
[ 21] 15.0-16.0 sec 368 KBytes 3.01 Mbits/sec
[ 21] 16.0-17.0 sec 434 KBytes 3.55 Mbits/sec
[ 21] 17.0-18.0 sec 396 KBytes 3.25 Mbits/sec
[ 21] 18.0-19.0 sec 365 KBytes 2.99 Mbits/sec
[ 21] 0.0-20.1 sec 7.99 MBytes 3.33 Mbits/sec
[ 21] Sent 5697 datagrams
[ 21] Server Report:
[ 21] 0.0-20.3 sec 7.99 MBytes 3.31 Mbits/sec 14.060 ms 0/ 5697 (0%)
root@ubuntu:/home/voider/Desktop/mininet#

```

## "Node: h2"

```

[ 21] 0.0- 1.0 sec 745 KBytes 6.10 Mbits/sec
[ 21] 1.0- 2.0 sec 670 KBytes 5.49 Mbits/sec
[ 21] 2.0- 3.0 sec 606 KBytes 4.96 Mbits/sec
[ 21] 3.0- 4.0 sec 673 KBytes 5.52 Mbits/sec
[ 21] 4.0- 5.0 sec 603 KBytes 4.94 Mbits/sec
[ 21] 5.0- 6.0 sec 670 KBytes 5.49 Mbits/sec
[ 21] 6.0- 7.0 sec 604 KBytes 4.95 Mbits/sec
[ 21] 7.0- 8.0 sec 672 KBytes 5.50 Mbits/sec
[ 21] 8.0- 9.0 sec 604 KBytes 4.95 Mbits/sec
[ 21] 9.0-10.0 sec 673 KBytes 5.52 Mbits/sec
[ 21] 10.0-11.0 sec 606 KBytes 4.96 Mbits/sec
[ 21] 11.0-12.0 sec 672 KBytes 5.50 Mbits/sec
[ 21] 12.0-13.0 sec 604 KBytes 4.95 Mbits/sec
[ 21] 13.0-14.0 sec 670 KBytes 5.49 Mbits/sec
[ 21] 14.0-15.0 sec 670 KBytes 5.49 Mbits/sec
[ 21] 15.0-16.0 sec 606 KBytes 4.96 Mbits/sec
[ 21] 16.0-17.0 sec 672 KBytes 5.50 Mbits/sec
[ 21] 17.0-18.0 sec 630 KBytes 5.16 Mbits/sec
[ 21] 18.0-19.0 sec 610 KBytes 5.00 Mbits/sec
[ 21] 0.0-20.0 sec 12.6 MBytes 5.28 Mbits/sec
[ 21] Sent 8989 datagrams
[ 21] Server Report:
[ 21] 0.0-20.1 sec 12.6 MBytes 5.25 Mbits/sec 58.519 ms 0/ 8989 (0%)
root@ubuntu:/home/voider/Desktop/mininet#

```

## "Node: h1"

```

root@ubuntu:/home/voider/Desktop/mininet# iperf -u -s
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 208 KByte (default)

[ 21] local 10.0.0.1 port 5001 connected with 10.0.0.2 port 39630
[ 22] local 10.0.0.1 port 5001 connected with 10.0.0.4 port 55760
[ 23] local 10.0.0.1 port 5001 connected with 10.0.0.3 port 50399
[ ID] Interval      Transfer      Bandwidth      Jitter    Lost/Total Datagrams
[ 21] 0.0-20.1 sec 12.6 MBytes  5.25 Mbits/sec 58.520 ms  0/ 8989 (0%)
[ 22] 0.0-20.2 sec 3.24 MBytes  1.35 Mbits/sec 7.172 ms  0/ 2310 (0%)
[ 23] 0.0-20.3 sec 7.99 MBytes  3.31 Mbits/sec 14.060 ms  0/ 5697 (0%)

```