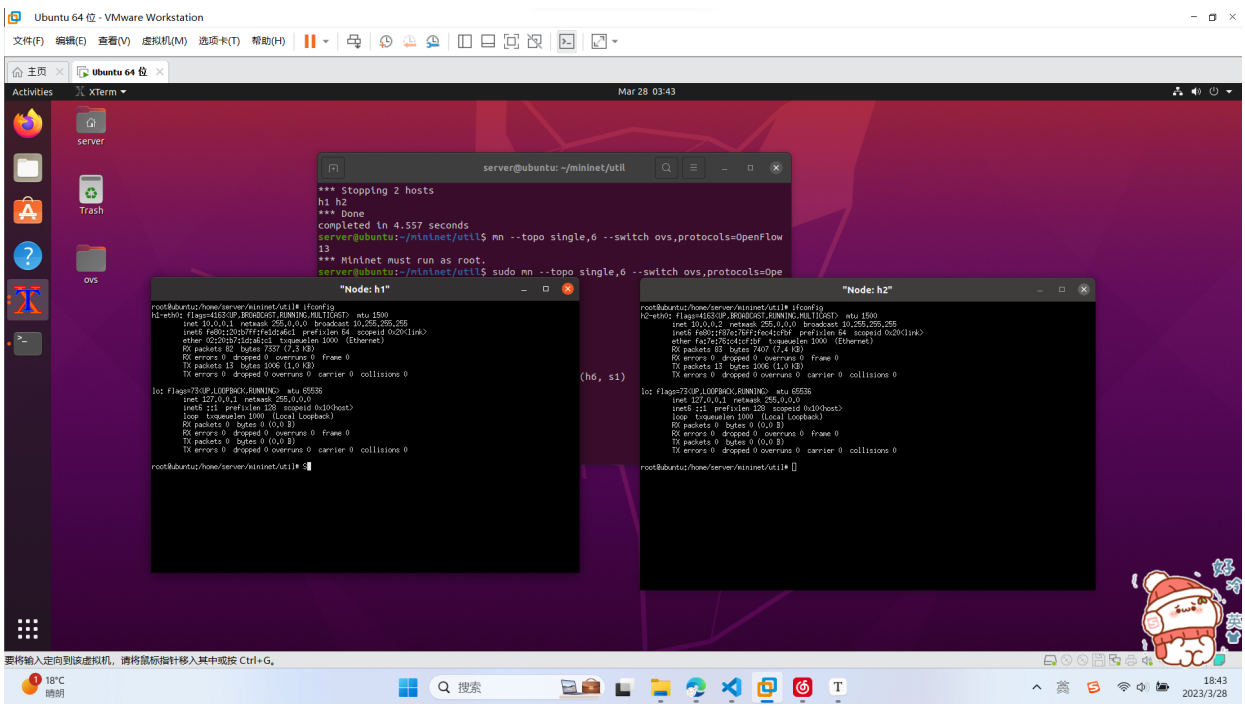
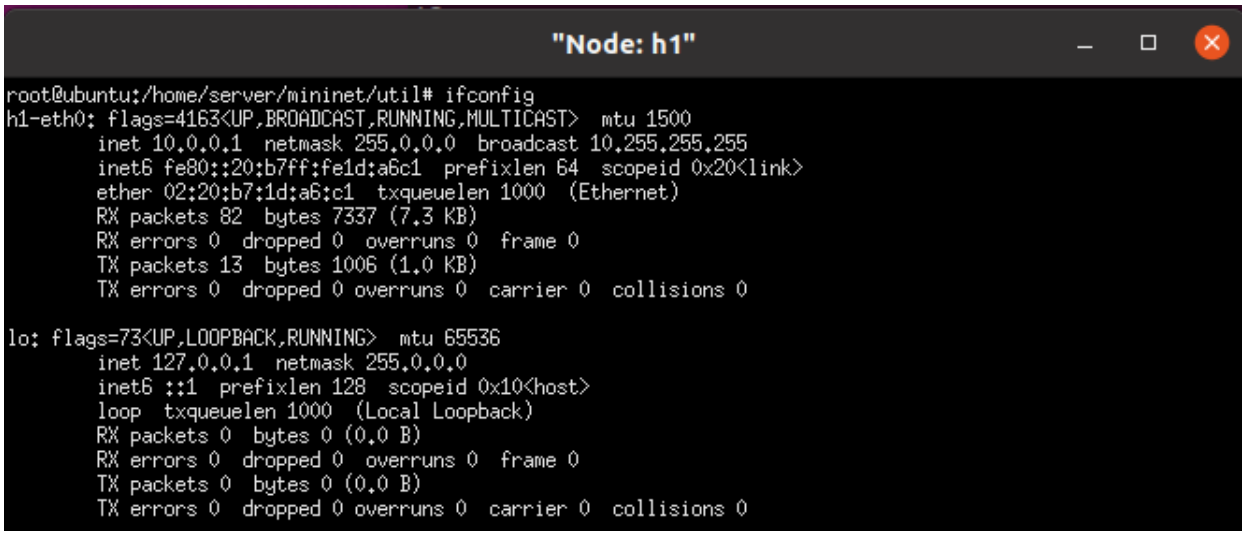
Lab3 QoS

Task1 连通性测试

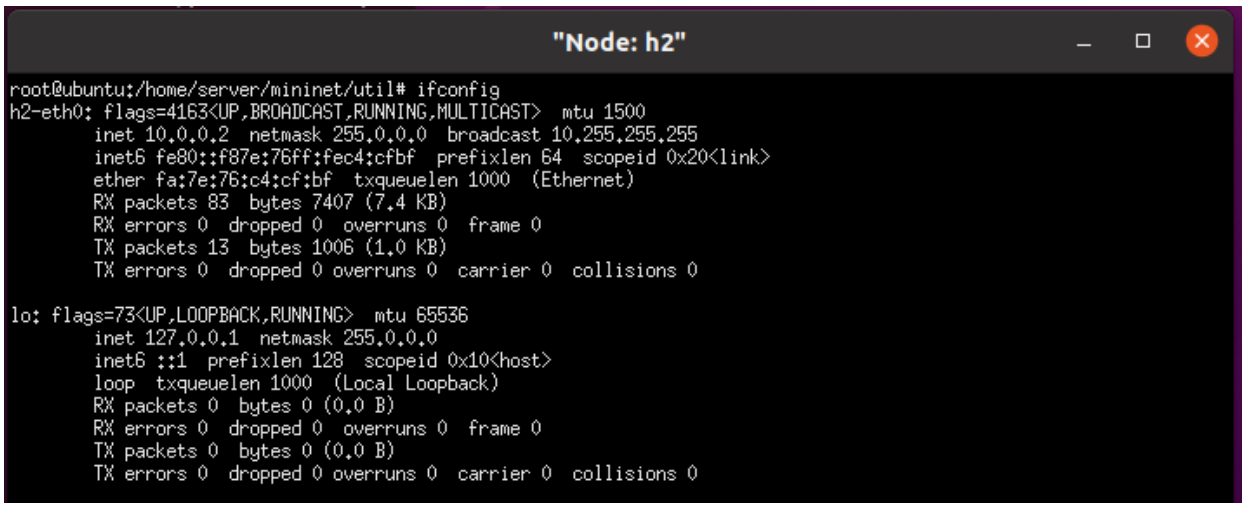
> 截图如下所示（没调虚拟机时区，所以有点怪）：



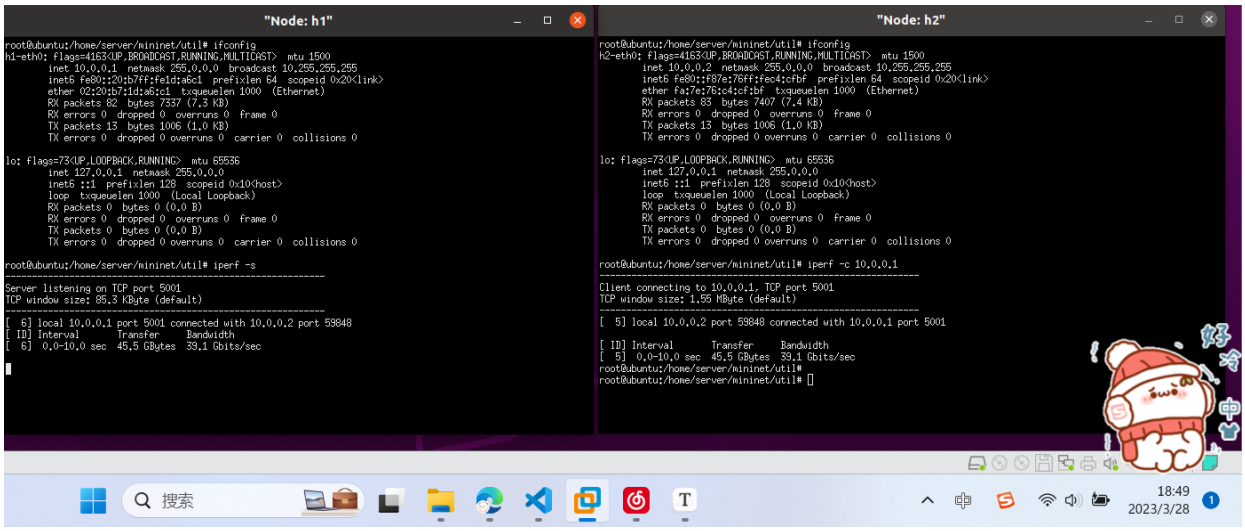
放大的第一个H1的结果如下，他的IP是10.0.0.1：



第二个H2的结果如下所示，可以看出他的IP是10.0.0.2



然后进行连通性测试，可以看出连通性正常！



Task2 三种限速方式

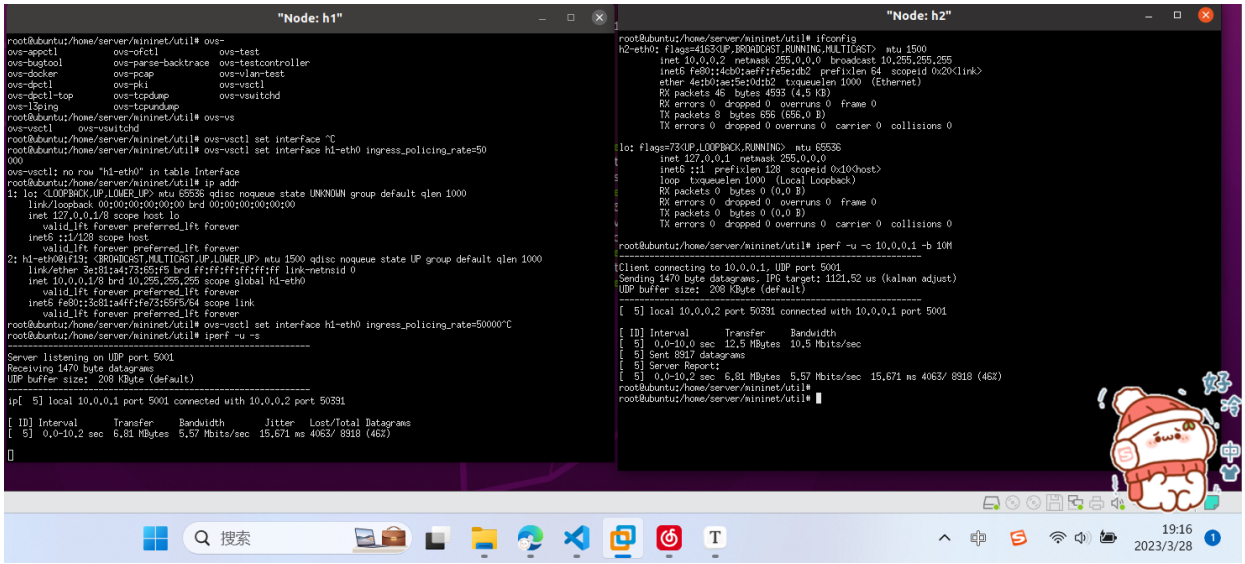
> 三种限速方法依次包括网卡限速、队列限速、Meter表限速

1、网卡限速

> - 网卡限速通过在虚拟机终端设置网卡的收包速率（不是在H1、也不是在H2里面执行，在虚拟机执行）
> - 限速原理：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。

项目	带宽	Jitter	丢包率
数值	5.57Mbits/sec	15.671ms	4063/8918(46%)

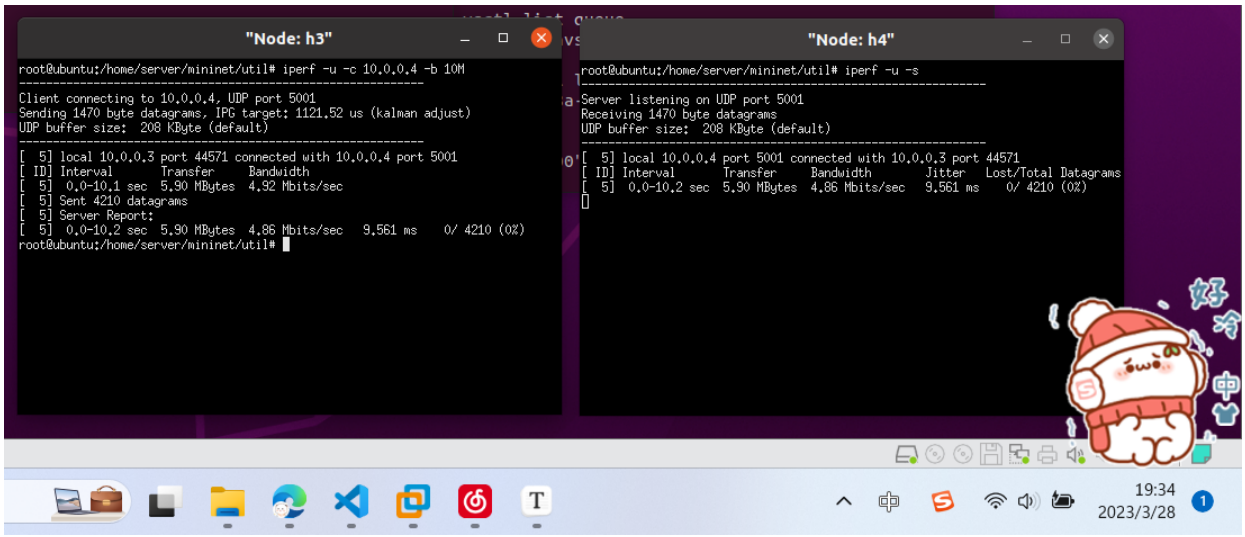
两个终端的截图如下所示，具体得到的数值如上表格所示：



2、队列限速

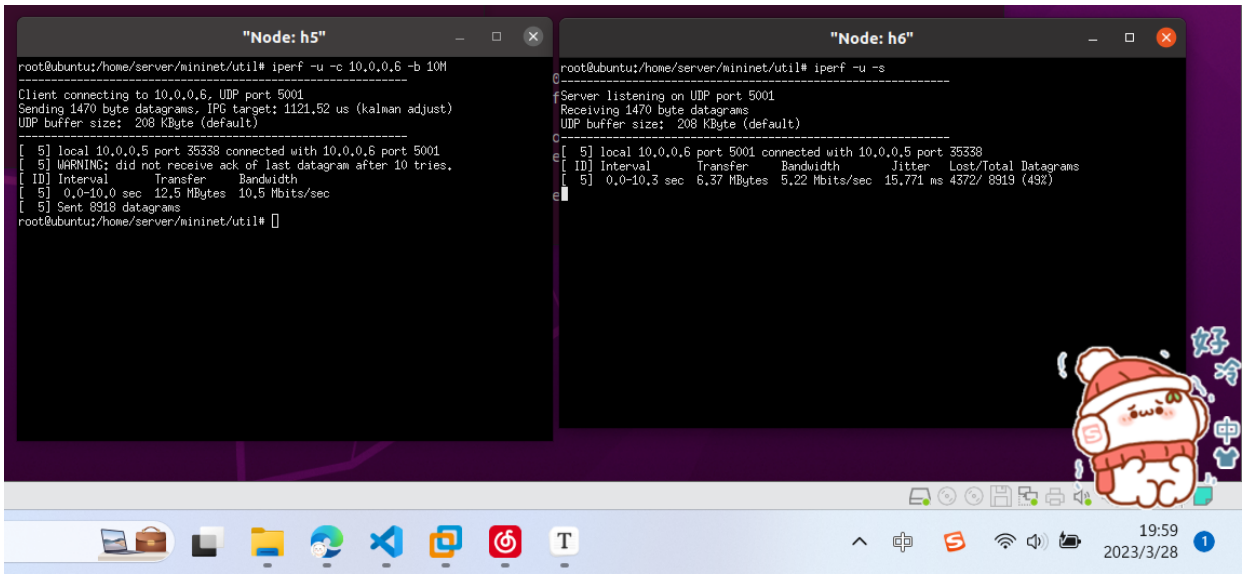
> - Linux可以将网络数据包缓存起来,然后根据用户的设置,在尽量不中断连接(如 tcp)的前提下来平 滑网络流量。内核通过某个网络接口发送数据包,它都需要按照这个接口的队列规则把数据包加入队列。
> - 实验通过h4是服务端，h3是client。

项目	带宽	Jitter	丢包率
数值	4.86Mbits/sec	9.561ms	0/4210(0%)



3、队列限速

项目	带宽	Jitter	丢包率
数值	5.22Mbits/sec	15.771ms	4372/8919(49%)



5、问题回答一

> 解释下面的命令含义：
>
> ```
> $ ovs-ofctl add-flow s1 in_port=5,action=meter:1,output:6 -O openflow13
> $ ovs-ofctl dump-flows s1 -O openflow13
> ```

这两个指令含义如下：

- 第一个指令是下发转发的流表
  - `add-flow` 添加一个s1的流
  - `in_port=5` 代表数据包进入的端口是“s1-eth5”

    - 然后action代表数据包交给meter表处理，我们之前再meter表里面设置了超过 5M 的数据包丢弃，所以这里为了实现速度限制，交给了meter表
    - `output:6` 代表数据包进入
    - `-O` 指明了 OpenFlow 的版本 13
- 第二个指令用来查看s1表里面的条目
    - `-O` 指明了 OpenFlow 的版本 13
    - `dump-flows` 表示要输出 `s1` 流表的条目。

```
server@ubuntu:~/Desktop$ sudo ovs-ofctl dump-flows s1 -O openflow13
 cookie=0x0, duration=875.017s, table=0, n_packets=53573, n_bytes=80987676,
in_port="s1-eth5" actions=meter:1,output:"s1-eth6"
 cookie=0x0, duration=922.169s, table=0, n_packets=68, n_bytes=86674,
priority=0 actions=CONTROLLER:128
```

6、问题回答二

> 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。

	带宽 Mbits/sec	**抖动 ms**	**丢包率**
网卡限速	5.57Mbits/sec	15.671ms	4063/8918(46%)
队列限速	4.86Mbits/sec	9.561ms	0/4210(0%)
Meter 表限速	5.22Mbits/sec	15.771ms	4372/8919(49%)

- 从限速带宽效果来看，我们的限速的目标是5Mb/s，网卡限速、Meter表限速的最终结果都超过了5Mb/s，但是队列限速的结果为4.86
- 从丢包率来看：队列限速效果最好，丢包率为0。这取决于他的原理，将网络数据包缓存起来,然后根据用戶的设置,在尽量不中断连接(如 tcp)的前提下来平滑网络流量。回避了丢包问题。网卡限速和Meter 表限速是当速率超过了配置速率，就简单的把数据包丢弃，所以丢包率严重。
- 从抖动的数据来看，Meter的抖动最高。这可能和ovs交换的流表控制能力有关。交换机中数据流计数，动作的执行、流表的匹配可能影响其控制力度。相比较于硬件的交换机，可能软件实现的虚拟的交换机对流表的控制还是有若干的限制。

Task3 实际场景

一、清除旧的规则

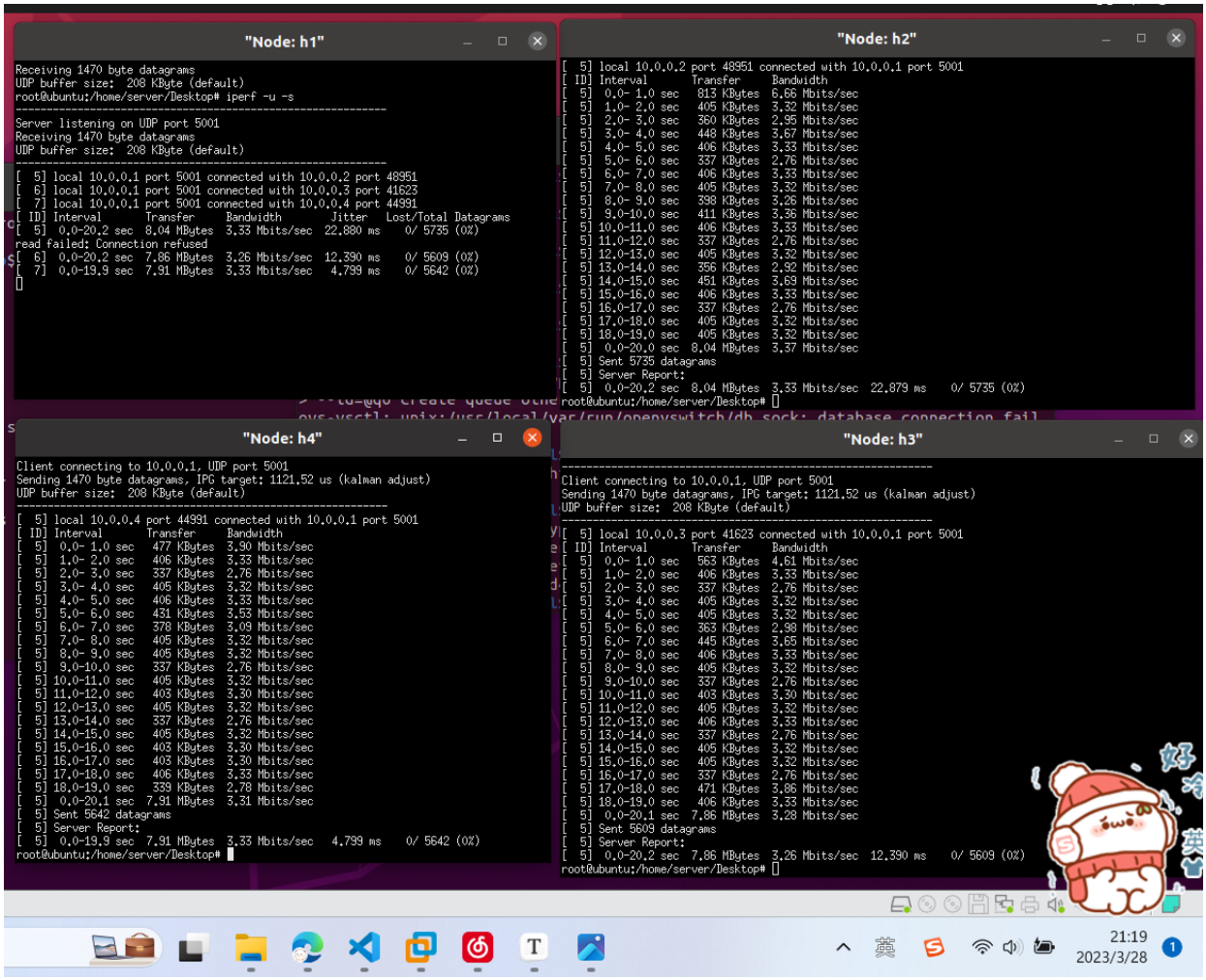
```
ovs-vsctl clear port s1-eth1 qos
# ... 同样对于  s1-eth2  s1-eth3  s1-eth4
ovs-vsctl -- --all destroy qos -- --all destroy queue
ovs-vsctl list qos
ovs-vsctl list queue
```

二、限速发送端10M

- 通过执行队列限速来控制速度

```
sudo ovs-vsctl set port s1-eth1 qos=@newqos -- \
--id=@newqos create qos type=linux-htb queues=0=@q0 -- \
--id=@q0 create queue other-config:max-rate=10000000
```

- 限速之后，h2-4同时开始，可以发现速度最终趋近于三个平衡，大约速率都在3Mb/s左右。



三、实际应用

- 网卡限速可以用于控制Server带宽，但是不太适合客户端的控制，所以经过考虑，我选择使用Meter表来控制三个客户端的带宽
- 根据说明，H2要5Mb及以上，H3要3Mb及以上，h4在保证h2和h3的前提下尽量多
- 那我们就做一个折中，H2控制在5-5.5Mb，H3控制在3-3.5Mb，H4大概就在1Mb左右
- 所以我们在发送端做一个流控
- 我发现如果min-rate恰恰设置为5M、3M的时候，反而有时候速度会稍微低于要求值，为了保证质量，我稍微提高了一些。

```
sudo ovs-vsctl set port s1-eth1 qos=@qos1 -- --id=@qos1 create qos type=linux-htb
queues=2=@q2,3=@q3,4=@q4 -- \
--id=@q2 create queue other-config:min-rate=5200000 other-config:max-rate=5800000 -- \
--id=@q3 create queue other-config:min-rate=3200000 other-config:max-rate=3800000 -- \
--id=@q4 create queue other-config:min-rate=0 other-config:max-rate=1800000
```

然后下发到流表中，分别指定不同队列分配给不同 Client：

```
sudo ovs-ofctl add-flow s1 in_port=2,action=set_queue:2,output:1 -O openflow13
sudo ovs-ofctl add-flow s1 in_port=3,action=set_queue:3,output:1 -O openflow13
sudo ovs-ofctl add-flow s1 in_port=4,action=set_queue:4,output:1 -O openflow13
```

实验的结果也非常的理想：

- h2的速度达到5.5以上，甚至接近6M，保证游戏可以玩的舒服
- h3的速度保证3.5以上，下载的游戏速度也得到保证了
- h4的带宽也可以达到最大化的利用，大约1.多的M，带宽利用非常充分