计算机网络 Lab5

1 problem1

我选择使用bbr控制算法进行测试。

传统的TCP控制算法往往以丢包作为网络堵塞的标志，但在现代的实际网络中，即使链路通畅也可能出现丢包，此时传统TCP算法会降低传输速率，导致带宽利用率降低。bbr算法采用主动探测的方式判断网络是否阻塞，从而可以更好利用网络带宽。

衡量网络承载能力的一个重要参数为传播时延与带宽的乘积（即bandwidth-delay product，BDP），当TCP发送窗口高于该值时，将会有数据包阻塞于链路中交换机缓存队列中；若低于该值，则无法充分利用带宽。于是，BDP的测量与计算便是bbr算法的重点。

为了计算该参数，我们需要分别测量传播时延与带宽。首先，通过将发送窗口大小限制在一个较小的值，保证网络不阻塞，此时的RTT可以近似看做链路上排除排队延时后的传播时延。随后增大传输速率，当RTT开始增加时，链路中缓存队列开始增长，此时的传输速率便是线路带宽。

2 problem2

编辑`/etc/sysctl.conf`，如下所示，启用bbr算法。

```
# for what other values do
#kernel.sysrq=438
net.core.default_qdisc=fq
net.ipv4.tcp_congestion_control=bbr
net.ipv4.tcp_congestion_control=vegas
~
-- INSERT --
```

3 problem3

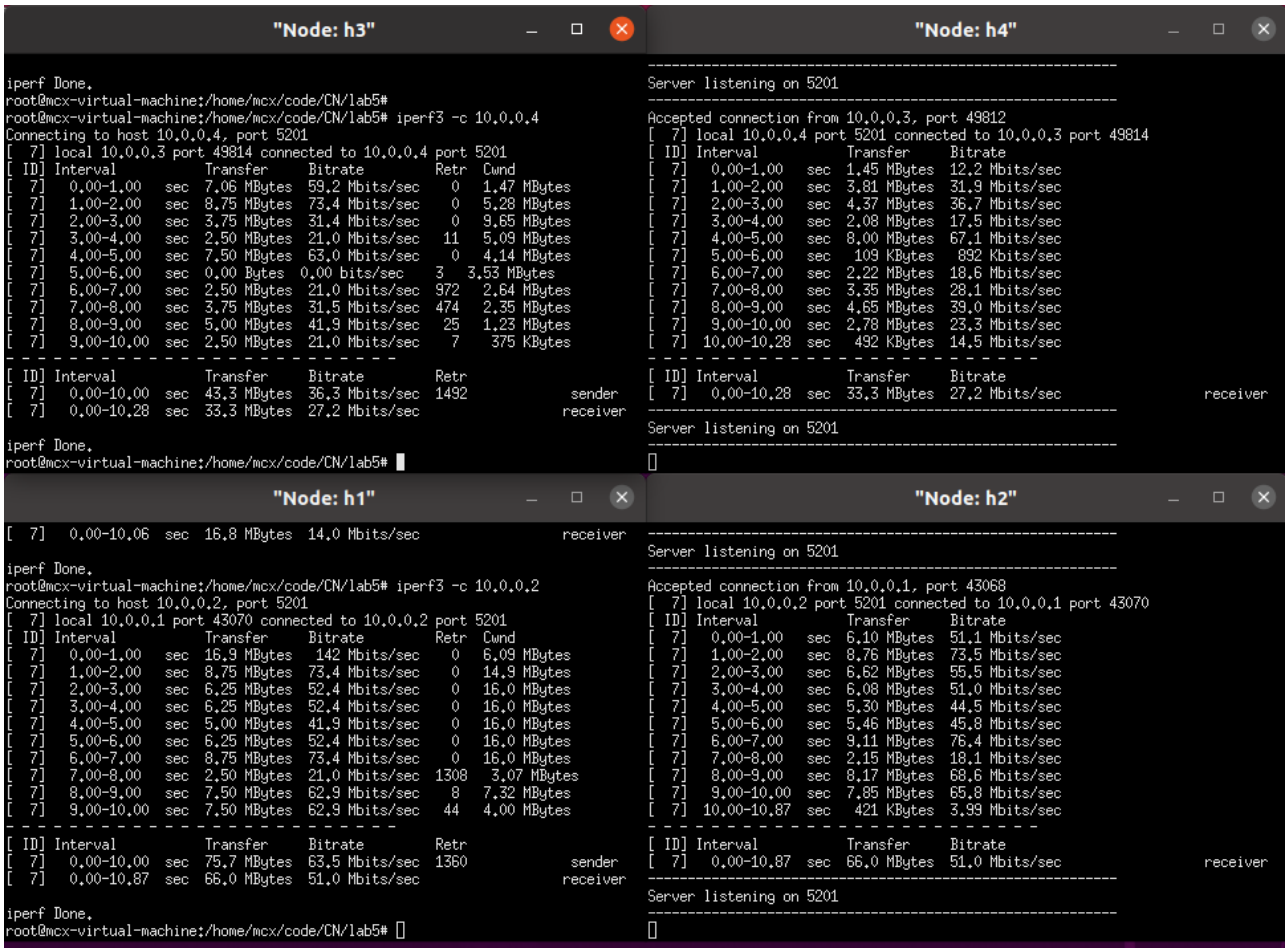
建立mininet网络（代码见`network.py`），修改s1-s2之间的链路延迟和丢包率，分别在reno和bbr算法下测试连接速度，如下表所示：

网络状态	reno	bbr
5ms,0%	97.2Mbps	78.8Mbps
20ms,0%	94.5Mbps	80.1Mbps
5ms,0.1%	30.3Mbps	76.0Mbps
20ms,0.1%	35.0Mbps	75.1Mbps
200ms,0.1%	14.4Mbps	51.5Mbps
200ms,0.2%	5.64Mbps	47.1Mbps

可以看出，reno算法受延时与丢包率的影响较大，当延迟与丢包率较高时，带宽利用率非常低。bbr不使用丢包作为阻塞标志，在网络条件较差的情况下依然能有不错的性能。

4 problem4

在上一题的mininet网络中加入主机h3、h4（代码见`network2.py`），开启终端，同时测试h1-h2、h3-h4之间的连接速度，两对连接的瓶颈为s1-s2链路。



设置s1-s2延时为20ms，丢包率为0.1%，测试结果如下：

	(h1,h2)	(h3,h4)
bbr	63.5Mbps	36.3Mbps
reno	52.0Mbps	41.9Mbps

设置s1-s2延时为200ms，丢包率为0.1%，测试结果如下：

	(h1,h2)	(h3,h4)
bbr	30.4Mbps	7.3Mbps
reno	13.6Mbps	10.5Mbps

需要说明的是，在本测试中，测试结果有一定波动，上表中数据为反复测试后较为有代表性的数据。

可以看出，reno算法在竞争状态下更能保证公平性，这是因为当网络出现瓶颈丢包时，reno算法会主动降低传输速率，使得竞争者可用带宽增加。bbr算法对公平性的保证相对较弱，这主要是因为bbr算法主动探测网络状态，在下一次重新探测前，不会降低发送窗口长度，相对其他竞争者更有优势。