report

Tang Peng: 517020910038

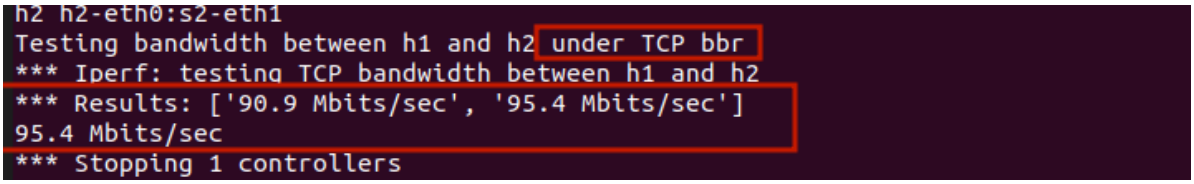
In this lab, we are going to test TCP performance of different TCP congestion control algorithms by using Mininet .

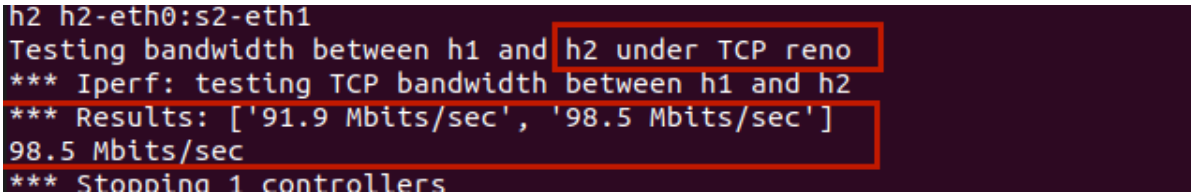
Problem 1

Here we choose TCP bbr to discuss. Most TCP algorithms use the packets loss as the signal to decline the date transmission speed, while TCP bbr uses the bottleneck bandwidth and round-trip propagation time to judge whether the network becomes congestion or not. In this way, the TCP bbr can find the network congestion earlier, which can avoid packets being blocked in the bottleneck router queue. Thus, the packets loss can be reduced largely and the throughput can be improved as well.

Problem 2

We tested the bandwidth of TCP Reno and TCP Bbr and obtained the following results. From the following figures, we can see that the throughputs of TCP Reno and TCP Bbr are almost equal.



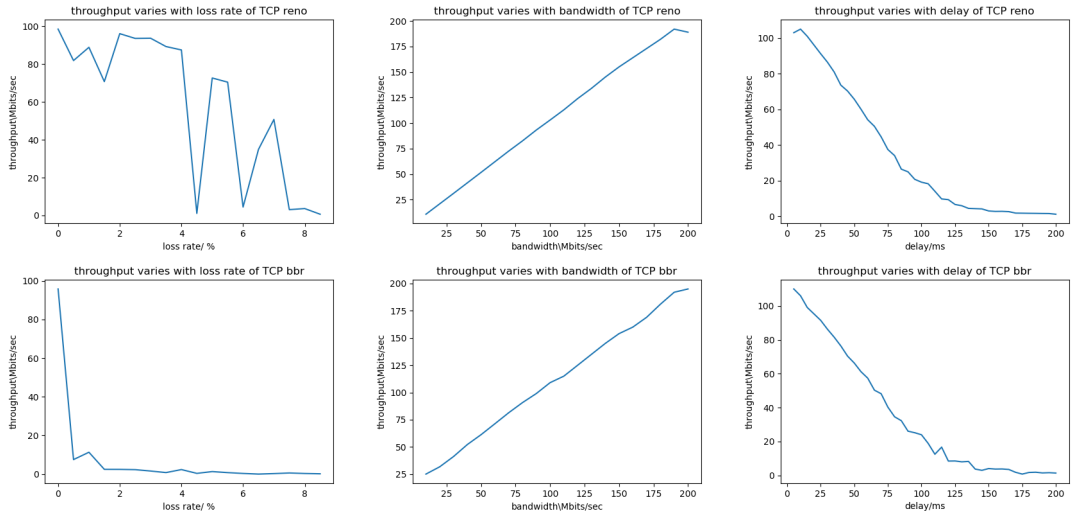


Problem 3

In this problem, we construct a network with only one pair of sender and receiver. We test the throughput with different link bandwidth/link delay/loss rate for the above two TCP versions.

- When loss rate varies, we let link bandwidth be 100 Mbits/sec and link delay be 5ms.
- When link bandwidth varies, we let link delay be 5ms and loss rate be 0.
- when loss rate varies, we let link bandwidth be 100Mbits/sec and link delay be 5ms.

By running the code in *problem3.py* file, we can get the following results.

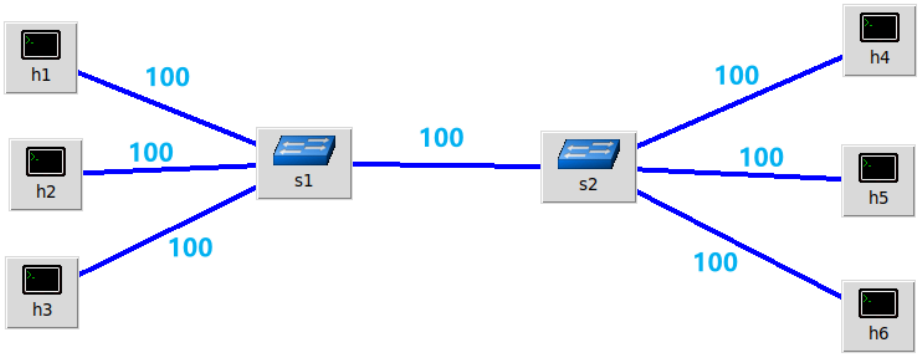


- From above figure, when the delay and bandwidth don't change, we can see that the TCP bbr can be largely influenced by the loss rate. When loss rate reaches 2%, the throughput is almost 0. But for TCP reno, the influence of loss rate is smaller than TCP bbr.
- When loss rate is 0 and delay is 5ms, the throughput almost linearly increases with the bandwidth increases. What's more, the TCP bbr has a bit better performance than TCP reno with the same bandwidth.
- When delay increases, the throughput of TCP bbr and TCP reno vary in same trend.

Problem 4

We construct a network with a bottleneck link shared by three pairs of senders and receivers. The bandwidth of the bottleneck link is 100Mbits/sec. We test the throughput of the two TCP versions in two cases.

- The following figure shows the network we construct. The number on the link means the bandwidth(Mbits/sec) of this link. Obviously, the bottleneck link is the link between **s1** and **s2**. We test the thought between **h1** and **h4**, **h2** and **h5**, **h3** and **h6**. In this case, every pair has the same bandwidth.

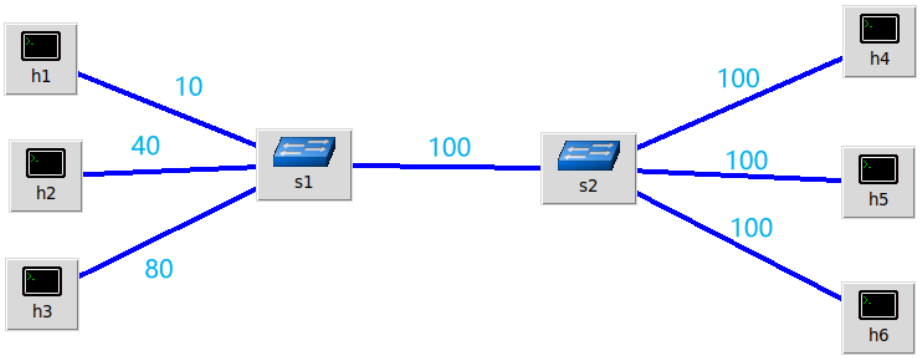


By running the code in *problem4.py* file and setting the corresponding bandwidth, we get the following results.

throughput	TCP Reno	TCP Bbr
h1---h4	40.1 Mbits/sec	29.1 Mbits/sec
h2---h5	39.9 Mbits/sec	31.0 Mbits/sec
h3---h6	38.9 Mbits/sec	30.3 Mbits/sec

From the table, we can see that the throughput of three pairs are almost same in both TCP Reno and TCP Bbr, which means the bottleneck link's bandwidth is divided equally among them.

- We can adjust the bandwidth of these links and we get the following network.



In this network, the bandwidth of three pairs are different. By running the code in *problem4.py* file and setting the corresponding bandwidth, we get the following results.

throughput	TCP Reno	TCP Bbr
h1---h4	9.64 Mbits/sec	9.30 Mbits/sec
h2---h5	36.5 Mbits/sec	33.9 Mbits/sec
h3---h6	48.6 Mbits/sec	46.6 Mbits/sec

From the table, we can see that the pair with smaller bandwidth will be satisfied.

With two case, we can see that the two TCP versions use the max-min fairness algorithm.