Liu Yongxiang
519021910029
sam.liu@sjtu.edu.cn

1 Question 1

I choose to talk about the “bbr” TCP congestion control algorithm, which was published in 2016 by Google and dramatically improve the TCP throughput.

There are four states in the algorithm “BBR”.

When the connection is created, the algorithm uses the function like standard TCP. It exponentially increases the delivery rate, trying to fill up the network pipe as soon as possible.

When the delivery rate cannot be increased, it starts to free the network pipe exponentially and finally reaches a stable state.

After the above two process, the algorithm starts to do bandwidth probing. It tests the maximum bandwidth and uses the probable maximum bandwidth to deliver the packages in the coming six cycles. Then repeat the work again. If the bandwidth increases and the delivery rate increases at the same time, the RTT will not increase. Hence, we can know whether the bandwidth increases by increasing the delivery rate. On the contrary, if the bandwidth decreases and the delivery rate does not change, the RTT will increase and we can know the the bandwidth decreases. The bandwidth probing process is the most significant.

Besides, the algorithm will probe the delay time every 10 seconds to probe the minimum delay.

2 Question 2

```
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
Testing bandwidth between h1 and h4 under TCP bbr
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['3.67 Mbits/sec', '29.1 Mbits/sec']
29.1 Mbits/sec
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 2 switches
s1 s2
*** Stopping 2 hosts
h1 h2
*** Done
```

Figure 1: Throughput of BBR

```
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
Testing bandwidth between h1 and h4 under TCP reno
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['3.61 Mbits/sec', '7.84 Mbits/sec']
7.84 Mbits/sec
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 2 switches
s1 s2
*** Stopping 2 hosts
h1 h2
*** Done
```

Figure 2: Throughput of Reno

Figure 1 shows the throughput of TCP using ‘bbr’ algorithm while Figure 2 shows the throughput of TCP using ‘Reno’ algorithm. From the result, we can see that under the situation given by the document, ‘bbr’ algorithm performs better than ‘Reno’ algorithm.

3 Question 3

To learn the relationship between TCP throughput and bandwidth, link delay or loss rate for the above two TCP versions, I changed the bandwidth, link delay and loss rate separately to make the ablation study.

Firstly, I change the attribute of bandwidth. The link delay is set to be 5ms and the loss rate is 0. I set the bandwidth with 100, 200, and 300. For 'bbr' algorithm, the running results are 76.8Mbps, 146Mbps, and 246Mbps. While for 'Reno', the results are 94.4Mbps, 181Mbps, and 260Mbps. We can see that in 'Reno' algorithm, it can make a better use of bandwidth when the delay time and loss rate are very small.

Secondly, I change the attribute of link delay, setting the bandwidth as 100 and loss rate as 0. I set the link delay as 5ms, 100ms, 200ms. For 'bbr' algorithm, the running results are 76.8Mbps, 57.4Mbps, and 27.2Mbps. It decreases as the link delay increases and the decreasing rate becomes faster. For 'Reno' algorithm, the running results are 94.4Mbps, 67.7Mbps, 30.6Mbps. The changing trend is similar to that of 'bbr' algorithm.

Thirdly, I change the attribute of loss rate, setting the bandwidth as 100 and link delay as 5ms. For 'bbr' algorithm, when loss rate is 5%, the running result is 49.2Mbps. When the loss rate is 10%, the result decreases dramatically to only 4.15Mbps. For 'Reno' algorithm, the decreasing is much more serious. When the loss rate is only 1%, the result is only 6.01Mbps.

From the results we can see, the throughput of TCP is seriously impacted by the loss rate and 'bbr' is more robust comparing to 'Reno'. This result can explain the running results in Question 2. In Question 2, the loss rate was set to 0.1%. Since 'Reno' is much more sensitive, it decreases much more than 'bbr' and the throughput became less than 'bbr'.

4 Question 4

In this question, I create a bottleneck network for four server-client pairs. I use one switch to connect those servers with 100Mbps bandwidth links and another switch to connect those clients with the same links. Then I connect the two switch with a link having 100Mbps bandwidth.

Firstly, I only ask one pair to transmit data and get the total throughput of the structure. The total throughput is 78.4Mbps.

Later on, I ask the four pairs to send messages at the same time and get the results below. From the result we can see, each pair of client-server shares part of the bottleneck pipe but no one is more than 1/4 of the bandwidth. This test is run on 'Reno', so I suppose that 'Reno' algorithm regulate that each pair can only have at most 1/4 of the bandwidth. Therefore, the network will not meet congestion. The running results of 'bbr' are similar.

```
Client connecting to 10.0.0.1, TCP port 5201
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 54350 connected with 10.0.0.1 port 5201
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  19.8 MBytes  16.2 Mbits/sec
```

Figure 3: h1 iperf h2

```
------------------------------------------------------------
[  3] local 10.0.0.4 port 48066 connected with 10.0.0.3 port 5201
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.9 sec  22.2 MBytes  17.1 Mbits/sec
```

Figure 4: h3 iperf h4

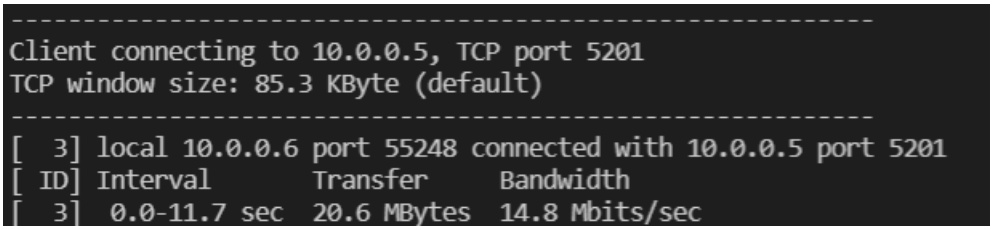

Figure 5: h5 iperf h6

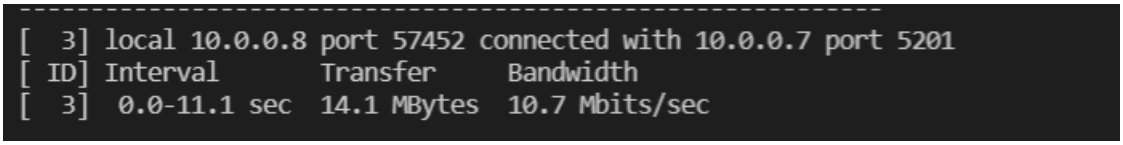

Figure 6: h7 iperf h8