Report for lab5

邬心远

519021910604

> 1 Pick one TCP congestion control algorithm other than TCP Reno, and explain how it works.

Vegas:

- when to resend:

  When receiving duplicate ACKs, Vegas check the time receiving and the time labeled, if the difference between them is larger than the *timeout interval*, then it send back data pack immediately instead of waiting for a third ACK. After receiving the ACK of resending package, Vegas reduce CongWin to $\frac{3}{4}$

  And when receiving first or second ACK after resending package, Vegas will check the time interval to see if it's larger than *timeout interval*, and resend if it is.

- how to get CongWin
  - expected throughput = RevWin/BaseRTT, (BaseRTT refers to the minimum value of RTT, usually the first round RTT)
  - actual throughput = RevWin/RTT
  - *diff* = (expected throughput - expected throughput)*BaseRTT

  Define threshold *a* and *b*

  - if *diff*<*a*, increase CongWin by 1
  - if *a*<=*diff*<=*b*, do nothing to CongWin
  - if *diff*>*b*, decrease CongWin by 1
- Slow start

  At starting, CongWin increase exponentially at every two ACKs, and when diff is larger than a, CongWin begins to increase linearly.

> 2. Enable TCP Reno and your selected TCP congestion control algorithm, and test them in Mininet.

result using reno:

```
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: ['3.54 Mbits/sec', '7.14 Mbits/sec']
7.14 Mbits/sec
```

result using vegas:

```
Testing bandwidth between h1 and h4 under TCP vegas
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['3.61 Mbits/sec', '6.13 Mbits/sec']
6.13 Mbits/sec
```

> 3 Construct a network with only one pair of sender and receiver. Study how TCP throughput varies with respect to link bandwidth/link delay/loss rate for the above two TCP versions.

	case1	case2	case3	case4
bandwidth(delay=5ms,loss=0.1%)	10Mbit/s	20Mbit/s	50Mbit/s	100Mbit/s
throughput(Mbit/s)	9.28	12.9	16.4	18.1
delay(bandwidth=50Mbit,loss=0.1%)	1ms	5ms	20ms	50ms
throughput(Mbit/s)	20.1	16.1	6.34	3.29
loss(bandwidth=50Mbit,delay=5ms)	0.1%	0.5%	1%	5%
throughput(Mbit/s)	14.5	9.23	5.38	1.90

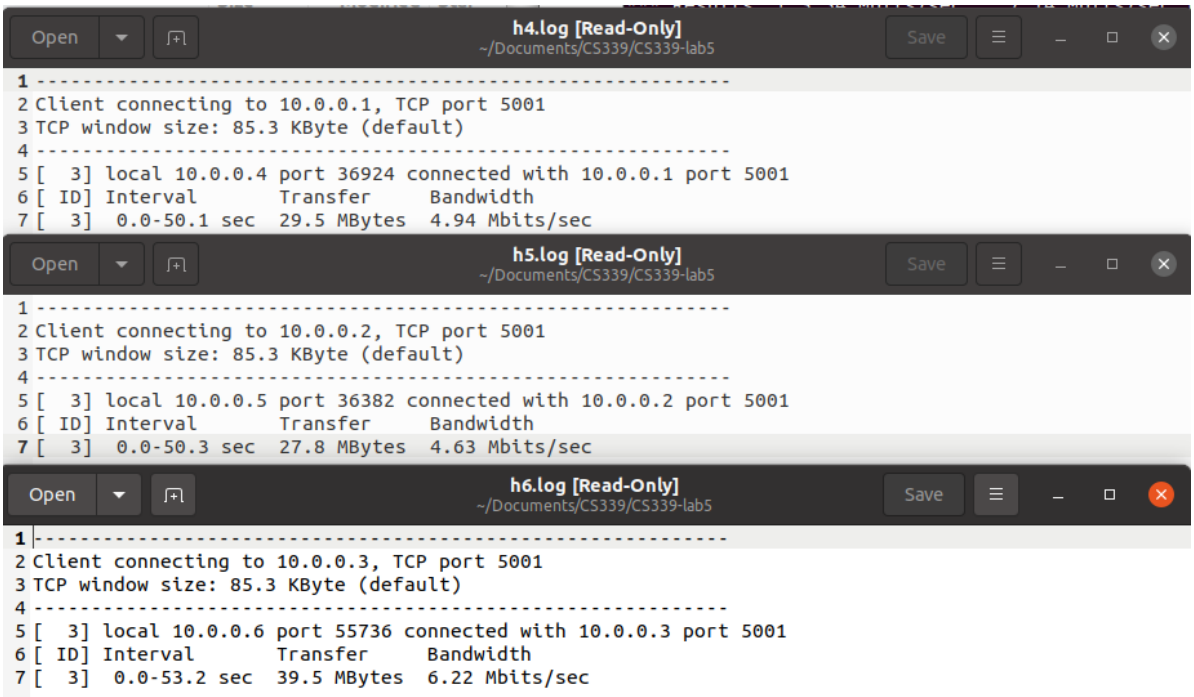
Especially, when changing bandwidth, I changed all three links in the net, and when changing delay and loss rate, I only change the link between two switchers.

We can see that as bandwidth increase, the throughput doesn't grow linearly, instead, it grow very slowly because of the delay and loss.

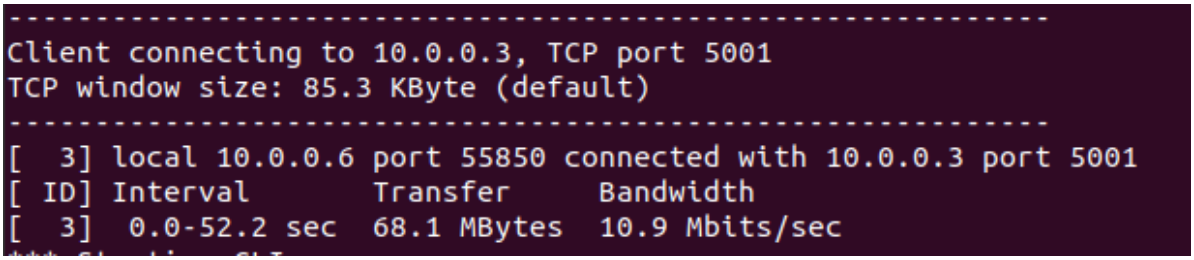
And as expected, the throughput drop as delay and loss rate increase.

> 4 Construct a network with a bottleneck link shared by multiple pairs of senders and receivers. Study how these sender-receiver pairs share the bottleneck link.

In this part, I set three pairs linked through same switchers, and running `iperf` simultaneously, and get the result as follows:



And running `iperf` on a single host:



We can see that the use rate of three pairs is larger than the single pair, I guess this may because of that the time of resending and waiting for ACKs is utilized.