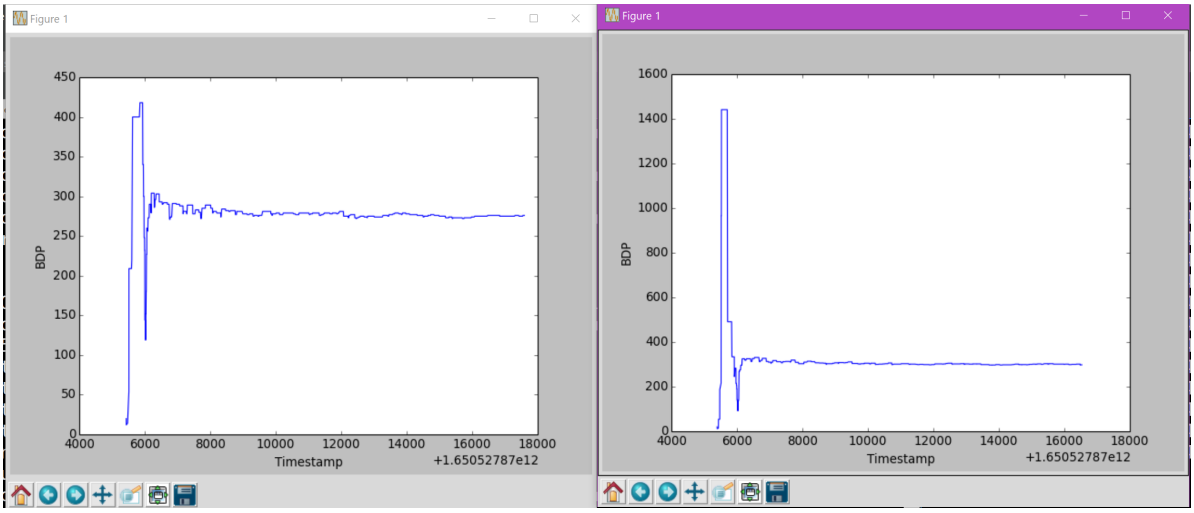
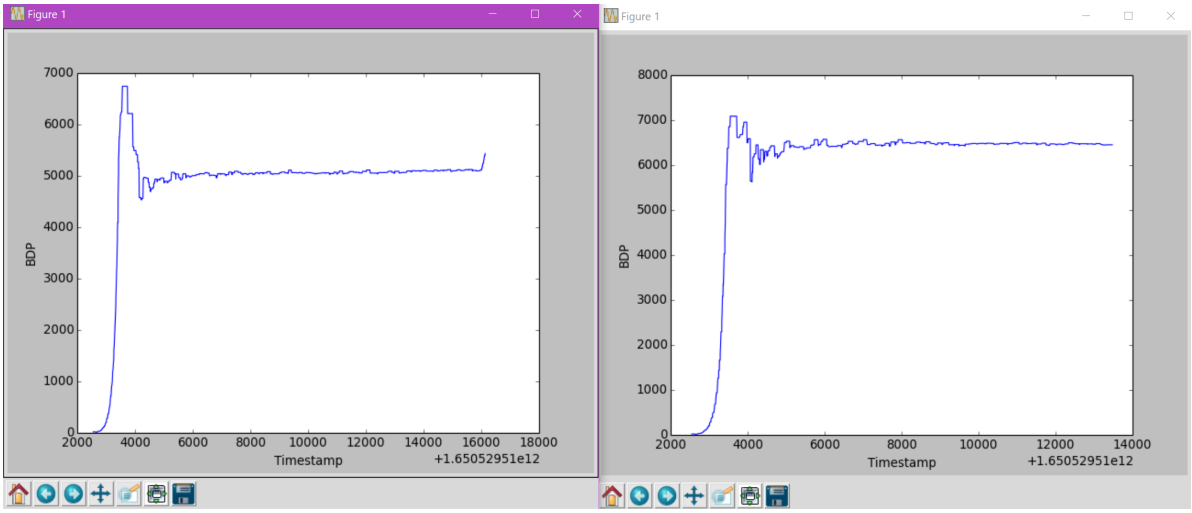
BDP Plot

Based on lab3_topology. Both of them got a fair bandwidth.



Also, when I tested on a topology with delay=50ms, the plot is much smoother.



Structure & Design

I referred to Google's implementation. The BBR module exports 4 API:

- bbr_update(): Every time the endpoint receives a packet, it builds a sample and delivers the sample into bbr_updates.
- bbr_sentNotice(): It tells the bbr how much the endpoint sends out.
- bbr_thisTimeSendPacing(), bbr_thisTimeSendCwnd(): They return the limit given by pacing and cwnd.
- bbr_retransmission_notice(): when the endpoint retransmits data, it tells the bbr, so bbr can get to know to slow down.

Most of the API have simple and direct effect, except *bbr_update* -- It controls bbr's internal status.

bbr_update updates the following parts in order.

update_congestion

This function determines how much data left in congestion mode. If zero data in congestion mode, we can reset to above bandwidth.

update_bw

This function uses the sample to calculate the bandwidth and update its dataset with *minmax_win*, which storages 3 recent sample bandwidth and timestamp in bandwidth decrease order.

update_cycle

It is used for *probe_bw* phase. Every min_rtt cycle, it shifts to another pacing_gain. Most of the time, the gain=1. But also, it sometimes goes higher to probe higher bandwidth, and sometimes it goes lower to share bandwitdth with others or drain the queue from probing bandwidth.

update_check_bw_full

It is used for *start* phase. If the bandwidth grows much slower than before, we can assume it reaches the limit, then we switch to *drain* phase.

update_check_drain

If we are in *drain* phase, and inflight data is less than bdp, we can return to above phase.

update_min_rtt

If we have a lower rtt sample, or, the current min_rtt is outdated, we need to use this sample as the min_rtt.

When we forced to update min_rtt because of the outdating, we will shift to *probe_rtt* phase. In this phase, we try to empty the pipe to get a lower rtt.

Test

1. Test on Dumbbell: It can reach to over 43Kbps, but it fails sometime. I assume the topology with higher delay (>10ms) can have a smoother plot because a small queueing will not lead to heavy retransmission.
2. On i2, it can reach 35Kbps.

Challenge

Test on topology with no delay is a huge challenge. BBR assumes the *min_rtt*=1ms, and any small queue will lead to a double RTT or higher (even tens of it.). When BBR trying to drain the pipe, timeout event occurs. The retransmission inserts too much data into pipe again!

So, I forced the BBR to assume the *min_rtt* must at least 5ms. Also, I try to limit the speed of transmission. From the lecture, I know fast retransmission function can remit the retransmission. So, I implement the fast retransmission function. Also, I used the token to limit the speed of the retransmission.

Remaining Bug

1. I have to shut cksum to keep it works properly.

2. Currently, cwnd just limits the speed when BBR is in *probe_rtt* phase, transmission mode and slow start. I think it should have more uses.
3. I just implemented a simple minmax_win to storage bandwidth sample. I should do more.
4. It has some "weird" bugs. For example, it appends CR to the file.. I have no idea about how it happened. I have to write addition code to delete the tail CR at the receiver. Another example is sometimes the sender will send single "00" in the stream. I have to write additional code to find these "00"s, and restrain these data from sending.