bbr

BBR Evaluation

Lawrence Brakmo (brakmo*at*fb.com)

1. Introduction

BBR is a new TCP variant developed by Google that was recently released to Linux's netdev branch. There is very little information at the moment other than the code. There is a QUEUE article that should be available before the end of September. I was specially interested in BBR when I read:

On the arrival of each ACK, BBR derives the current delivery rate of the last round trip, and feeds it through a windowed max-filter to estimate the bottleneck bandwidth. Conversely it uses a windowed min-filter to estimate the round trip propagation delay.

Since that is the exact same mechanism that both Vegas and NV use. Of course, BBR uses that information differently than Vegas and NV and I was curious about its performance. In the patch submission for BBR it is mentioned that it was tested in the context of the Internet and wide area networks including some production environments. The only results provided at the time were for single flows, and in one case, with 1% packet loss not related to congestion (where BBR performed well). This last case made me concerned since it can lead to unfairness when competing with TCP versions like Reno and Cubic that interpret packet losses as a sign of congestion. For example, if BBR fails to detect that the losses are due to congestion, then it will be more aggressive than either Reno or Cubic.

It is unlikely that one congestion control/avoidance will be best under all scenarios. Usually one will perform better under some conditions and worse under others. My goal was to understand under the conditions under which BBR would excel.

I used version 4 of BBR which is the version merged into the netdev kernel tree. I used the fq qdisc for all flows except LAN DCTCP flows. Since there was no explicit warning against its use in a LAN, I also included a LAN scenario. When there are multiple flows in a test, the flows are started staggered. I also did not look at the BBR code, but treated it like a black box.

This evaluation consists of 3 scenarios and 2 set of tests. The tests ran for 60 seconds.

Scenarios:

- *LAN with 20us RTT, 10 Gbps* - servers in same rack.
- *WAN with 10ms RTT, 10 Gbps* - servers in same rack, using netem for latency.
- *WAN with 10-40ms RTT, 10-40 Mbps* - servers in same rack, using server as router, tbf for limiting bandwdith, netem for latency.

Tests:

- *Fairness & Stability* - consists of 2 or 3 stream flow tests (each from a different server) to one receiver. It helps to visualize the dynamics of the congestion control (or avoidance), as well as to examine how each TCP variant competes against itself and against Cubic, and sometimes, Reno.
  - *2-flow tests*: the 1st flow lasts 60 seconds, and the 2nd flow lasts 20 seconds and starts 22 seconds after the 1st one.
  - *3-flow tests*: the 1st flow lasts 60 seconds, the 2nd flow lasts 40 seconds and starts 12 seconds after the 1st one, the 3rd flow lasts 20 seconds and starts 26 seconds after the 1st one.
  - *3, 6, ... flow tests*: To see how the TCP variants handle increasing loads of traffic. Do retransmissions stay stable or are there inflection points after which things degrade rapidly?
- *Latency and Flow Size Fairness* - consists of 3 servers sending to one receiver. Each sender sends N back to back 1MB and 10KB RPCs, where N = 1, 2, 4, 8, 16, 32. That is, for the case N=1, each sender uses 2 connections, 1 connection to send back-to-back 1MB RPCs and 1 connection to send back-to-back 10KB RPCs (i.e. 6 flows total to the receiver). This is also a good test to evaluate how the 10KB RPCs compete against the 1MB RPCs.

2. Results Summary

2.1 10G LAN Scenario

DCTCP performs really well, with no losses and excellent fairness against itself. Cubic does badly against DCTCP, but the cause is not DCTCP but the queuing discipline (RED with ECN markings) in the switches. It can be remedied by directing Cubic traffic to a separate queue with a fifo queue discipline. DCTCP did very well in the 10KB and 1MB RPCs (results not shown here). There were no losses until the very high loads, and the 10KB flows were not starved by the 1MB flows. The 10KB flows did much better under DCTCP than Cubic.

BBR does well in the 2 flow experiments, but not as well in the 3 flow experiments. There is less fairness and larger losses (1.2M packets retransmitted vs. 2K packets for Cubic). In the mixed traffic experiments (Cubic vs. BBR), Cubic took bandwidth away from the 2 flow tests. In the 3 flow tests fairness changed about every 10 seconds (my guess is that BBR changes its view of the network every 10 seconds). Cubic did well during some 10sec periods, and very badly in others. During the 3 flow tests Cubic had 2K retransmissions, while each BBR flow (there were 2) ahd about 225K retransmissions. As expected, Cubic did poorly during the 10 second periods when BBR was retransmitting a lot.

2.2 10G 10ms RTT WAN Scenario

In the 2 flow tests, Cubic did poorly in terms of throughput because, once it reduced its cwnd due to losses, it took a long time to reach full utilization. Fairness between Cubic flows was okay (25% difference in throughput). BBR was very fair most of the time, but had a lot of retransmissions once the 2nd flow started (640K packets, 1.3%, vs. 120 packets for Cubic). However, about every 10% of the time (3/25 tests), the seconds BBR flow did very badly (8.5Gbps vs. 100Mbps).

For the 3 flow tests, Cubic was unfair (but not catastrophically unfair as some of the BBR cases). BBR was quite a bit less fair than the 2 flow tests, but not as bad as Cubic. However, now in about 10% of the tests one of the BBR flows was suppressed (i.e. < 100Mbps throughput), and 2 of the flows were suppressed in another 10% of

the tests. BBR is showing signs of instability, at least at the 10s of seconds scales. BBR retransmits also shot up to 1.2 million packets (2.5%).

The 3 flow mixed tests show that Cubic is suppressed when competing with BBR in this scenario.

The 10KB and 1MB RPC tests indicate unfairness for the 10KB flows when running BBR at higher loads. The average and 99% latencies of the 10KB flows are 5 to 8 times worst at high loads. Probably caused by the sudden increase of BBR retransmissions at these high loads as compared to Cubic and Reno. BBR has 2.7% and 5.2% vs. 0.1% for Cubic and Reno at the high loads. Again, BBR is sensitive to load and its behavior changes suddenly when a threshold is crossed.

2.3 10M 40ms RTT WAN Scenario

BBR does well in this scenario. It prevents losses and it is somewhat fair. Again, the fairness changes about every 10 seconds. What if we increase the load? In this test, 3 hosts send to 1 receiver, 1 or 2 flows per sender, and we reduce the queue size at the bottleneck from 1000 packets (25 x BDP) to 64 packets (1.5 x BDP). With only 1 flow per host, BBR does very well (no retransmissions) until the buffer size is 125 or 64. Under this case the retransmissions jump to 2 or 4 times higher than for Cubic.

When we have 2 flows per host (6 flows total), BBR has large number of retransmissions for all buffer sizes. Ranging from 2.6% to 11% retransmissions (vs. 0.2% to 1% for Cubic). Now, BBR may be able to deal well with these high losses since it doesn't necessarily decrease its cwnd when there are losses. However, non-BBR cross traffic sharing the same bottleneck will perform very poorly because it will interpret losses as a sign of congestion. BTW, in all these experiments losses are a sign of congestion, but in many cases BBR does not interpret them as such.

2.4 Conclusions

Under some scenarios, BBR is able to correctly predict the available bandwidth and adapt to it given enough time (10s of seconds in some cases) when the conditions are somewhat static. However, BBR seems to have a mode where it determines that losses are uncorrelated to congestion and, if triggered incorrectly, will create heavy (2-10%) losses since it will not decrease its cwnd due to the losses. This is of great concern since in this mode, BBR can starve non-BBR flows sharing the bottleneck and, in some cases, even other BBR traffic that did not enter this mode. I saw BBR enter this mode incorrectly in my tests quite often.

3. Interpretation of Tests

The outputs from all tests include:

- *Goodput (payload throughput) graphs* - These include output for each flow as well as the cumulative goodput.
- *Cwnd graphs* - these include the cwnd for each flow. Red vertical lines indicate the time when one or more packets are retransmitted. the are a visual indicator of when congestion is occurring.
- *RTT graphs* - as seen by each server sending data
- *Table of results* - with rows for each server and the aggregate with columns for:
    - test parameters

- average cwnd
- average RTT
- average ping RTT
- rates (avg, min, max)
- latencies (min, avg, max, 50%, 90%, 99% and 99.9%)
- % packet retransmissions
- packet retransmissions

3.1 2-Flow and 3-Flow Tests

The 2-flow tests are used for:

- examining how quickly existing flow adapt to new flows
- examining how quickly flows adapt to released bandwdith from terminating flows
- fairness between flows
- levels of congestion
- instability - conditions under which TCP variant's performance changes abruptly

3.2 1MB and 10KB RPC Test

These tests are used for:

- Studying the behavior under increasing loads
- Performance (throughput and latency) of 1MB and 10KB flows. How fair is the available bandwidth divided between them?
- instability - conditions when the retransmissions or latency change abruptly

4. 2 and 3 Flow Tests

4.1 LAN Scenario

Consists of all servers within a rack, 10Gbps links and 20us RTTs.

4.1.1 2-Flow LAN, all flows same TCP variant

Figure 1A shows the Goodput of Cubic for 2 flows. It is quite fair at small time scales and very fair at large time scales. More interesting is Figure 1B, which shows the cwnd of each flows. In addition, it shows retransmissions as red vertical lines. To achieve full link utilization, we only need a cwnd of 80. So when cubic starts by itself, the host queue has more than 1000 packets. When the second flow starts, we have losses and the cwnds reduce to about half. When the 2nd flow ends, the 1st flow keep full bandwidth utilization but with a smaller cwnd.

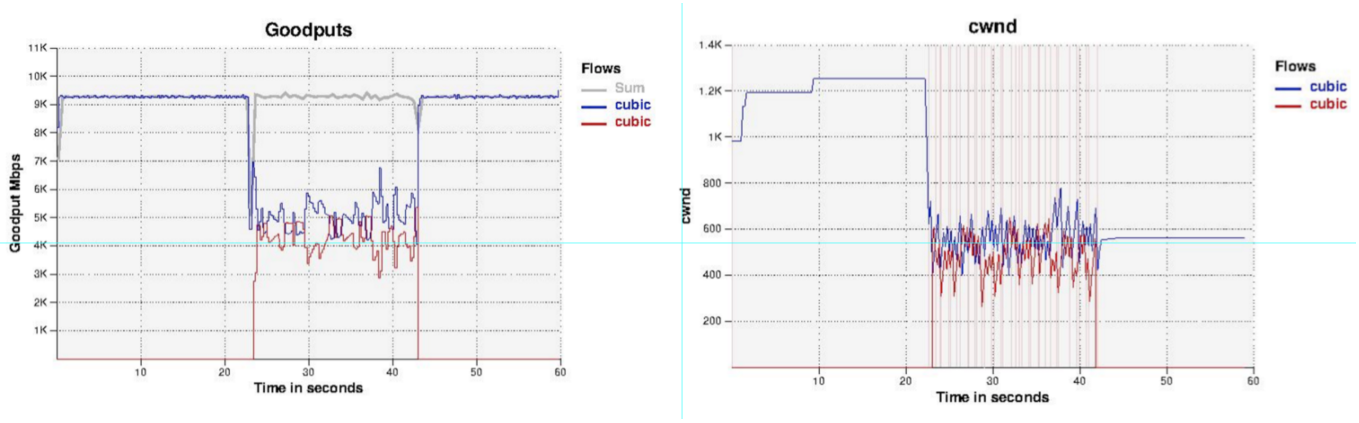

Figure 1: LAN 2 Flow Cubic, A.Goodput B:Cwnd

Figure 2A shows the behavior of DCTCP, which uses ECN markings at the router to achieve congestion avoidance. Note that congestion control creates congestion by increasing the cwnd until losses occur, while congestion avoidance is able to stop cwnd growth to prevent losses and congestion. You can see that DCTCP is more fair than Cubic. Again, Figure 2B is more interesting. It shows a large cwnd when there is only one flow. Since there is no congestion in the switch (all links are 10G), there are not ECN markings at the switch, and DCTCP cannot prevent host queue buildup. This could be prevented by using a queueing discipline that uses ECN marking. When the 2nd flow starts, each flow has a cwnd of around 80 (vs. 500 for Cubic) and there are no retransmissions. After the 2nd flow starts, the local queue grows quickly to 240 and then slowly after that.

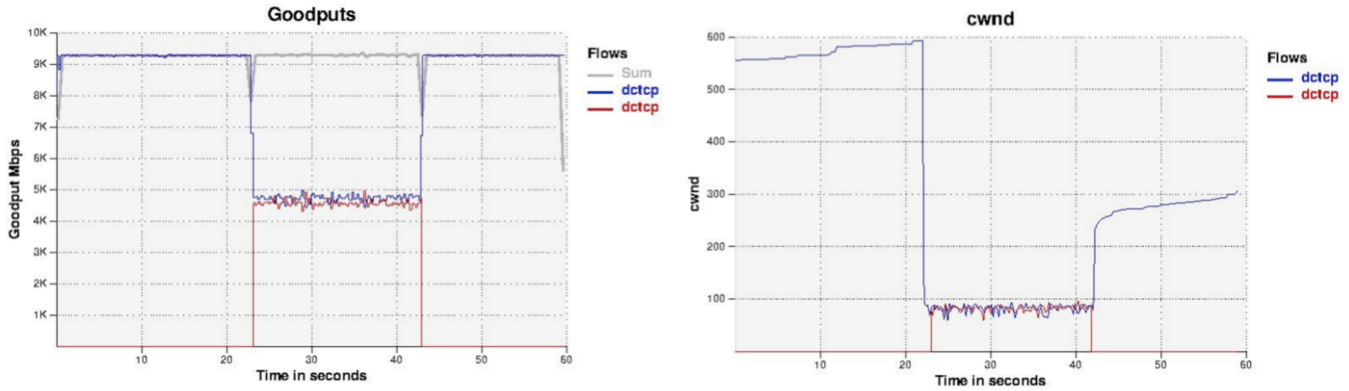

Figure 2: LAN 2 Flow DCTCP, A.Goodput B:Cwnd

Figures 3A and 3B show rate and cwnd for BBR. BBR is quite fair for 2 flows as seen by the rate plot. Figure 6 shows the cwnds for both BBR flows. They are reasonable in size. You can see that they drop every 10 seconds, probably to recalibrate their base RTT.

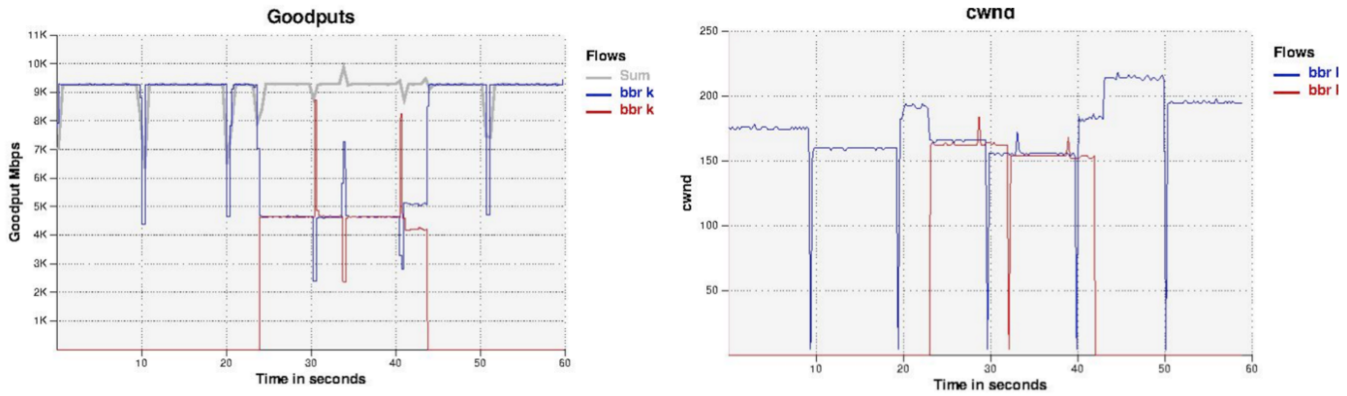

Figure 3: LAN 2 Flow BBR, A.Goodput B:Cwnd

Table 1: 2-Flow Tests same CA for both flows

CA	Rate 1st flow	Rate 2nd flow	RTT	Retransmits
Cubic	7.8 Gbps	4.2 Gbps	890 us	~ 0%
DCTCP	7.7 Gbps	4.6 Gbps	200 us	0 packets
BBR	7.6 Gbps	4.6 Gbps	300 us	0 packets

4.1.2 3-Flow LAN, all flows same TCP variant

So far BBR is behaving quite reasonably. It avoids congestion, just like DCTCP. But now lets look a the 3 Flow tests. I will only show cwnd plots since they all have good throughputs in the LAN environment with small RTTs. Figure 4 shows Cubic, once the 2nd flow starts we start to see retransmissions. These increase with the 3rd flow. Cubic is less fair with 3 flows.

DCTCP does really well (Figure 5), no retransmissions and the cwnd keeps decreasing as we add more flows. It is very fair. BBR (Figure 6) does well with 2 flows, but once the third flow is introduced it does badly (instability). It has even more retransmits than Cubic. Cubic has 2,074 retransmits, BBR has 1,187,682 retransmits! 500 times more! Not only that, there are retransmissions even after the 3rd flow stops. You can also see that the cwnds for each flow are quite different, leading to unfairness at the 10sec granularity. Even after the 3rd flow stops, the cwnds of the 1st and 2nd flow do not return to their previous levels, instead they stay quite high. There is even a period of congestion later on (around 50secs) that has no external trigger (i.e. no other flows are introduced). This does not look good.

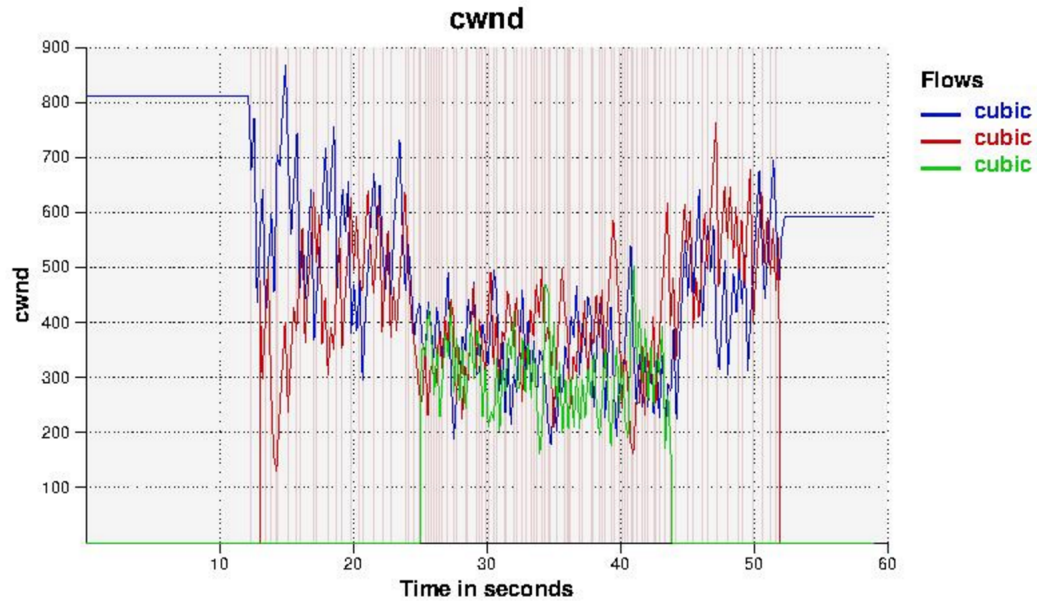

Figure 4: LAN Cubic 3 Flows LAN, cwnd plot. 2100 packets retransmitted

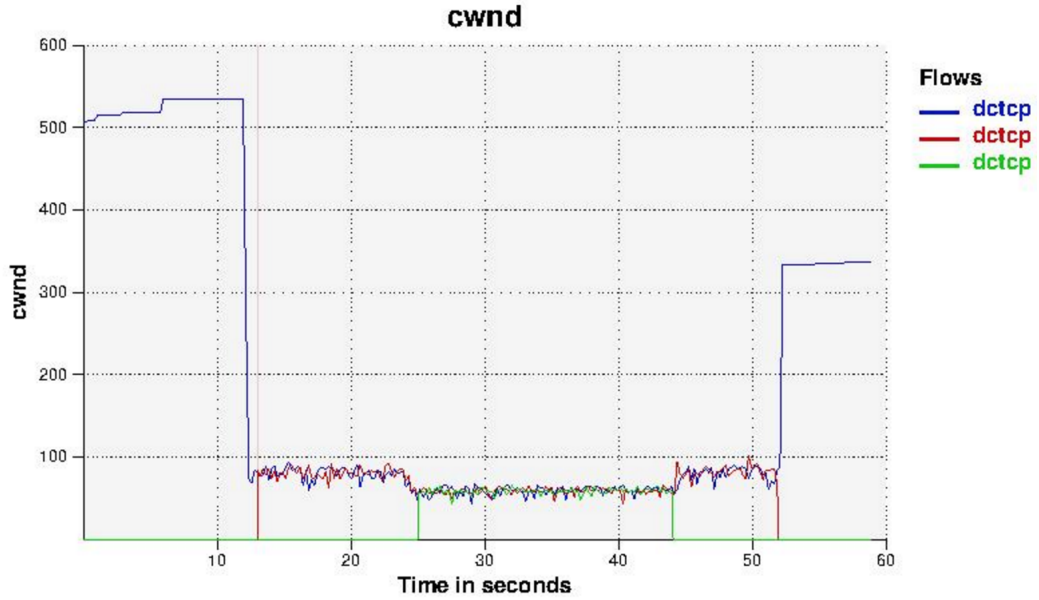

Figure 5: LAN DCTCP 3 Flows LAN, cwnd plot. No retransmissions

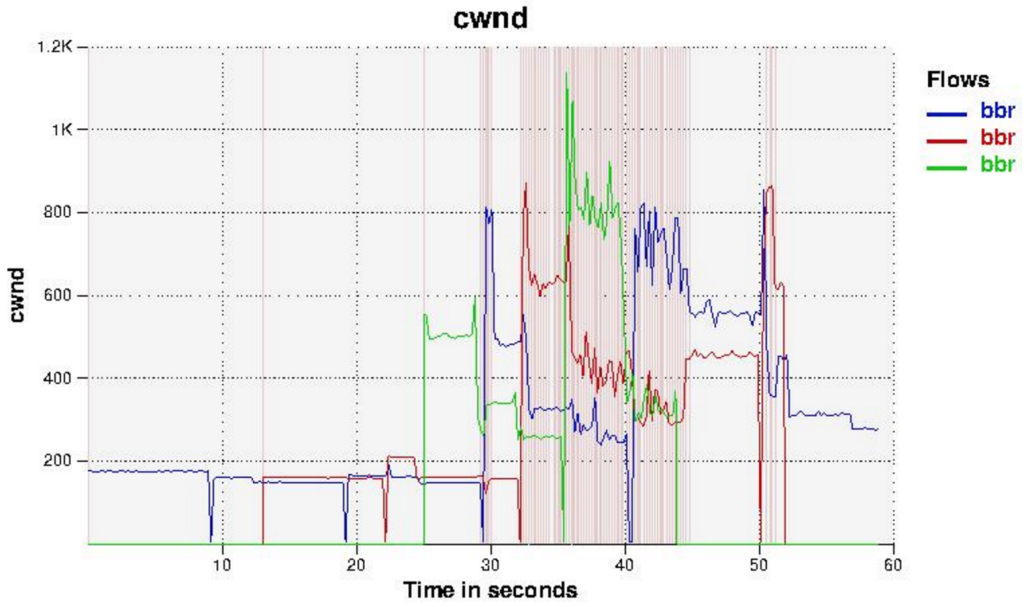

Figure 6: LAN BBR 3 Flows LAN, cwnd plot. 1.2 Million packets retransmitted

Table 2: 3-Flow Tests same CA for all flows

CA	Rate flow 1	Rate flow 2	Rate flow 3	RTT	Retransmits
Cubic	5.7 Gbps	3.9 Gbps	2.7 Gbps	1055 us	2K packets
DCTCP	5.7 Gbps	3.8 Gbps	3.1 Gbps	200 us	0 packets
BBR	5.6 Gbps	3.4 Gbps	3.8 Gbps	1000 us	1.2M packets

4.1.3 LAN, 2-Flow DCTCP vs. Cubic, BBR vs. Cubic

It is not recommended to use Cubic and DCTCP when using only one switch queue that is doing RED ECN marking. When done, as in the tests in this subsection, Cubic gets only a fraction of the available bandwidth. Note that DCTCP is not meant to be used this way. This is shown in Table 3. It also shows that DCTCP gets almost the full share of the bandwidth when it starts running. DCTCP, the 2nd flow, should only get 4.6 Gbps if it was perfectly fair to the 1st flow.

In contrast, BBR uses less than its fair share when Cubic starts running (BBR only uses 2 Gbps).

Table 3: 2-Flow Tests Mixed CA

CAs	Rate Cubic	Rate 2nd flow	RTT	Retrans Cubic	Retrans 2nd flow
Cubic vs DCTCP	6.2 Gbps	9 Gbps	188 us	31K pkts	0 pkts
Cubic vs BBR	8.6 Gbps	2.0 Gbps	940 us	796 pkts	965 pkts

4.1.4 Lan, 3-Flows DCTCP vs. Cubic, BBR vs. Cubic

In these experiments, a Cubic flow starts first followed by two DCTCP or BBR flows. Figure 7A shows the Rates for a Cubic flow versus 2 DCTCP flows. Once one DCTCP flow starts the Cubic rate collapses. When the 2nd DCTCP flow starts, both DCTCP flows share the bandwidth evenly, the Cubic flow keeps performing poorly. Again, this situation is known and should be avoided by using 2 queues at the switches, one with ECN enabled for DCTCP, the other without ECN for Cubic flows. Cubic retransmissions: 50K packets, DCTCP: 0 packets.

Figure 7B shows the case for Cubic vs. two BBR flows. When the 1st BBR flow starts, the Cubic flow is using most of the bandwidth for a while, then its rate collapses as the BBR flow explosively increases its cwnd. When the 3rd flow starts, the rate of the Cubic flow suddenly increases and then drops as both BBR flows quickly increase their cwnds. Packet retransmissions occur during the periods when the BBR cwnds increase and Cubic goodput collapses. Note that the increases in the rate of the BBR flows happens at the 10 second intervals when it temporarily resets its cwnd value. The behavior looks very unstable. Cubic retransmissions: 2K packets, BBR: 500K packets.

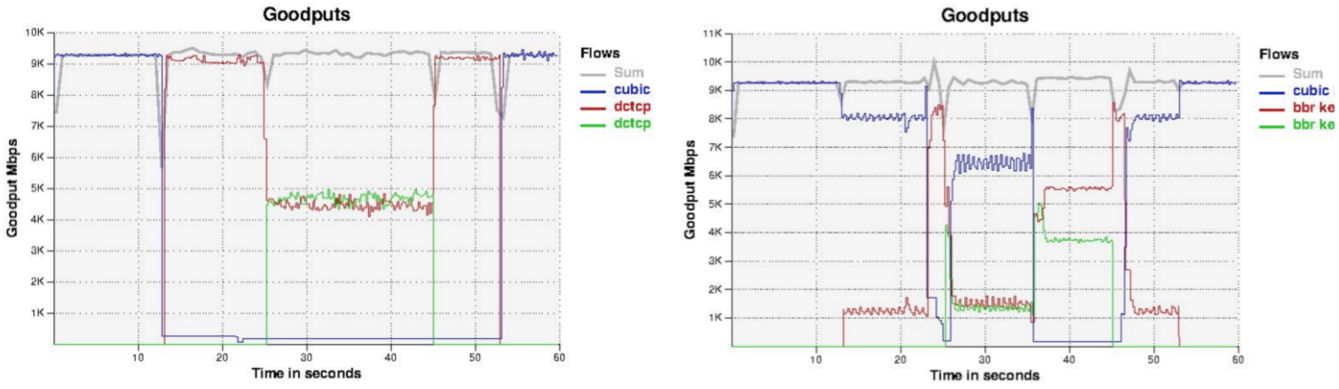

Figure 7: LAN 3 Flow Mixed, A:DCTCP vs Cubic, DCTCP B:BBR vs Cubic

Table 4 shows rates and retransmissions. Note that Cubic has a lot more retransmitted packets when competing with DCTC than it had when competing against itself (57K vs. 1K). In contrast, both BBR flows have a lot of retransmissions, more than 200K packets for each flow. It is of great concern how aggressive BBR is and how it doesn't react to a large number of retransmissions.

Table 4: 3-Flow Tests Mixed CAs

CAs	Rate Cubic	Rate flow 2	Rate flow 3	RTT	Retrans Cubic	Retrans 2	Retrans 3
Cubic vs. 2-DCTCP	3.2 Gbps	6.8 Gbps	4.7 Gbps	200 us	57K pkts	0 pkts	0 pkts
Cubic vs 2-BBR	6.4 Gbps	3.0 Gbps	2.6 Gbps	1100 us	2K pkts	277K pkts	215K pkts

4.2 10Gbps, 10ms RTT WAN Scenario

For this scenario links are 10G and uncongested RTT is 10ms (through netem). The switch queue has space for approximately 1,000 packets (about 1/8 BDP).

4.2.1 10Gbps, 10ms WAN, 2-Flows, all flows same TCP variant

We are only going to look at Cubic and BBR since DCTCP is only recommended for within a DC. Figure 8A shows the rates of 2 Cubic flows. It is somewhat fair, but the link is underutilized at times. Also, Cubic takes a long time to grow its cwnd after the 2nd flow ends.

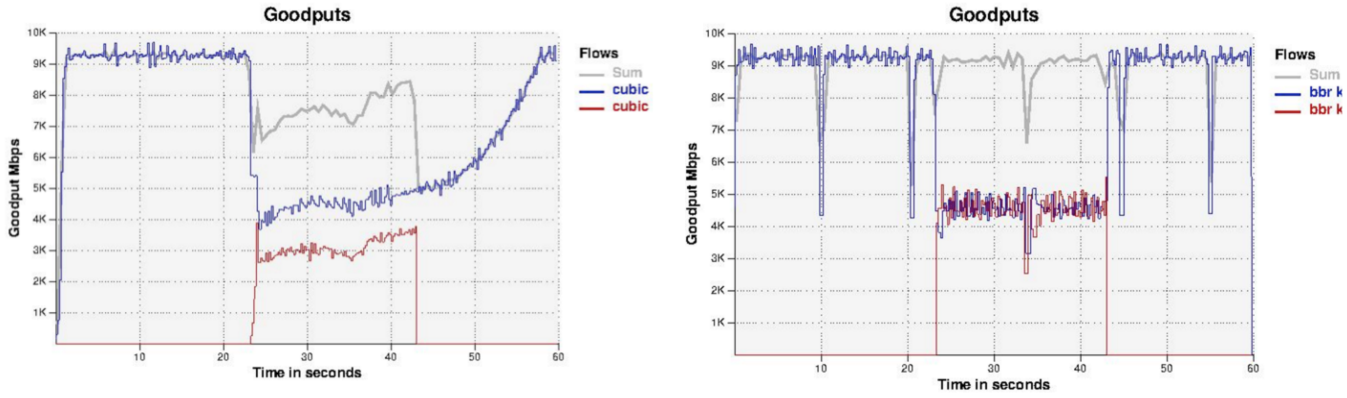

Figure 8: 10G WAN 2 Flow, A: Cubic 120 packet retrans B: BBR 640K packet retrans (1.3%)

Figure 8B Shows the rates for 2 BBR flows. It is quite fair, and good link utilization. Also, the throughput of the 1st flow increases quite quickly after the 2nd flow ends. However, every so often (around 10% of the time or 3/25 instances), the plots look like those of Figure 9. The 2nd flow achieves very little throughput. What about losses? The Cubic flows retransmit 500 packets, the fair BBR flows retransmit 640,000 packets (1.3% retransmission rate), the unfair ones only about 25K packets.

It looks like BBR has determined that the losses are not due to congestion so it ignores them.

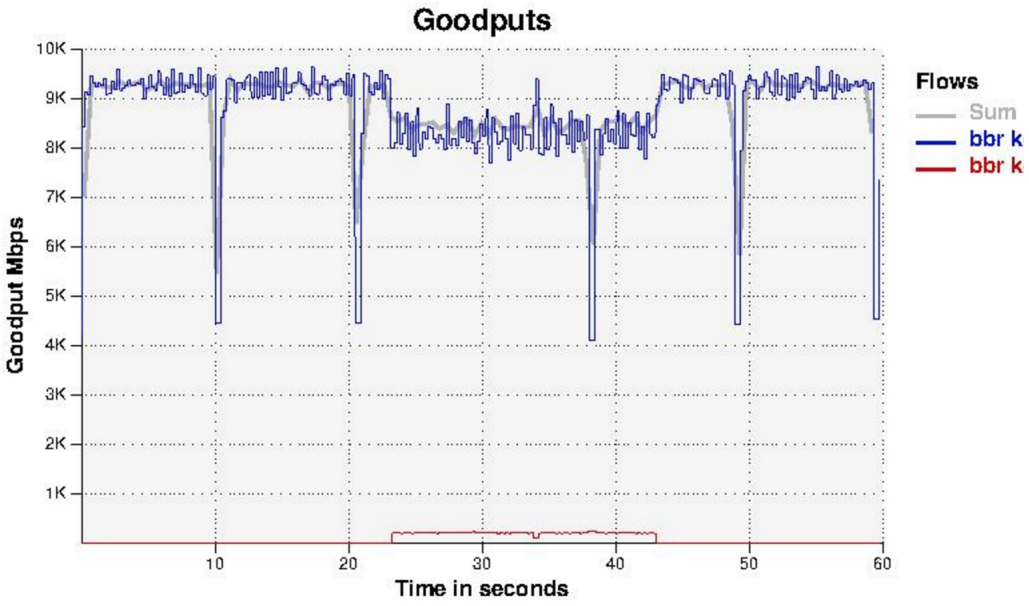

Figure 9: 10G WAN 2 Flow BBR Unfair. 25K retransmissions (0.06%)

4.2.2 10Gbps, 10ms WAN 3-Flows, all flows same TCP variant

Next let's look at 3 Flows. Figure 10A shows 3 Cubic flows. The link is underutilized because Cubic cannot grow fast enough. This is unexpected, will need to check if something bad has happened to Cubic behavior in the latest kernels. It is not very fair, primarily based on where slow start ends (due to losses) and its slow growth.

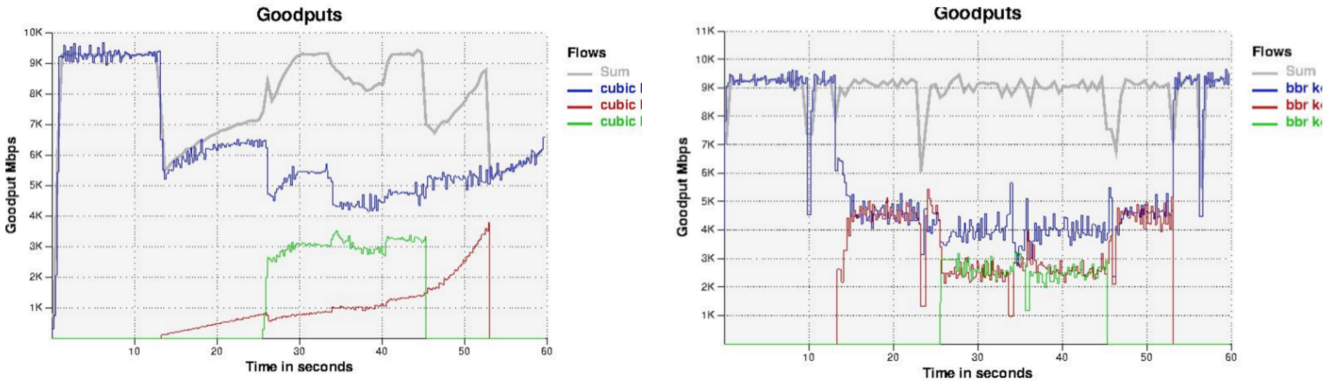

Figure 10: 10G WAN 3 Flow, A: Cubic 300 packet retrans B: BBR 1.2 million packet retrans (2.5%)

Figure 10B shows 3 BBR flows. Utilization and fairness reasonable. So BBR is much better than Cubic, right? Not so fast. Every so often I saw the behaviors shown in Figures 17 (4/35 test instances) and 18 (3/35 test instances) where one or two of the flows had very low rates. There is even one test instance where the 1st flow collapsed soon after starting, before any other flow started. Not sure of the causes, but BBR doesn't look very deterministic in its behavior under this scenario.

On average, 3-Flow BBR retransmitted 1.2 million packets per test instance (2.5%)

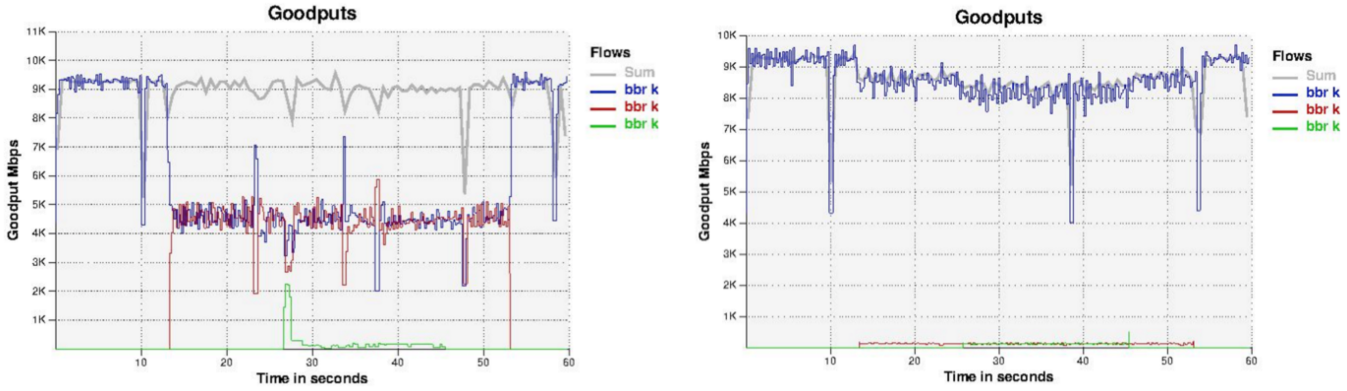

Figure 11: 10G WAN 3 Flow, A: BBR unfair B: BBR doubly unfair

4.2.3 10Gbps, 10ms WAN, BBR vs. Cubic

We have seen that BBR creates a lot of losses and it itself is (most times) unaffected by these losses. As a result, we can predict that BBR will damage Cubic flows that share the same bottleneck. Figure 12 shows that a Cubic flow gets very little bandwidth when competing with one or two BBR flows that started first. Figure 13 shows that the story is the same when Cubic starts first. Even after the BBR flows stop, Cubic is not making headway for 7 seconds, probably due to RTO exponential backoff.

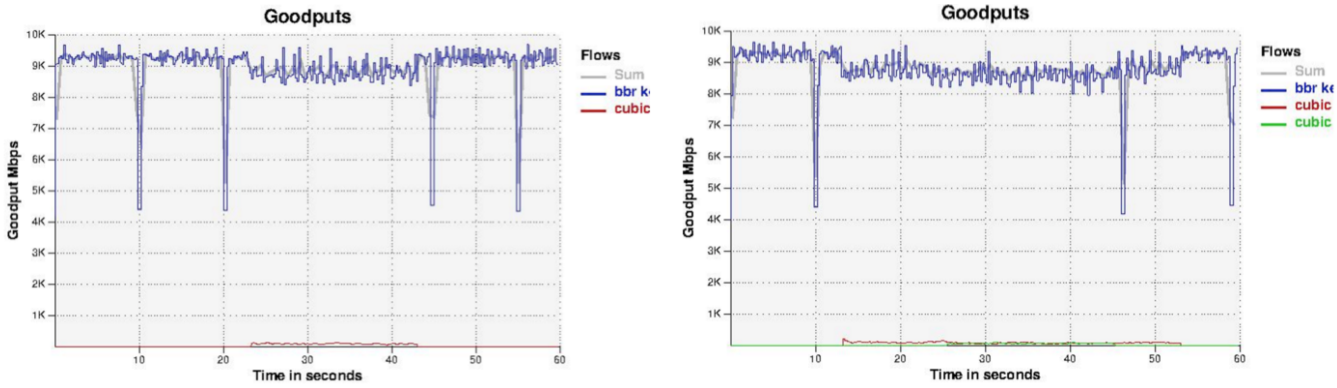

Figure 12: 10G WAN mixed Flows, A: BBR vs Cubic B: BBR vs 2 Cubics

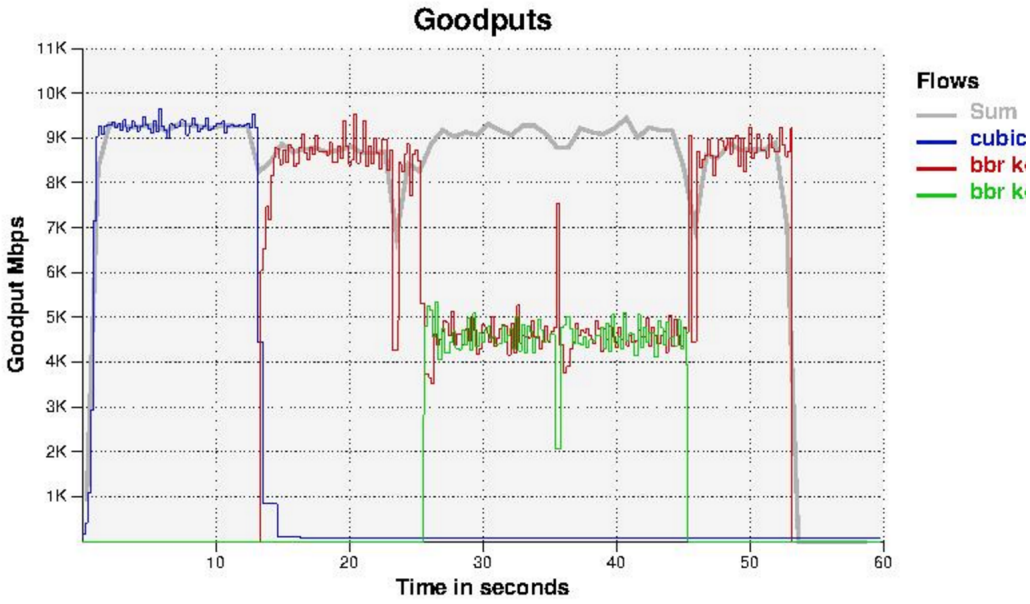

Figure 13: WAN 1 Cubic Flow vs. 2 BBR Flows

What about losses? When there a one long BBR and one short Cubic, BBR retransmits about 5600 packets, Cubic 44. When there are 3 Flows, if 2 flows are Cubic, then BBR retransmits 18600 while Cubic retransmits 180. Things get ugly when there are 2 BBR flows and one Cubic. BBR retransmits 340,000 packets/flow, Cubic only 6,800

4.3 10-40Mbps, 40-10ms WAN

Under these conditions, BBR performs well with 2 or 3 flows. It controls its cwnd to prevent losses and is somewhat fair. Figure 14 shows the cwnd for 3 instances of BBR at 10Mpbs, 40ms RTT. We see that it is fair with two flows, less fair when the 3rd flow starts.

Instead, we are going to explore stability by increasing the number of flows and decreasing the amount of buffering at the bottleneck.

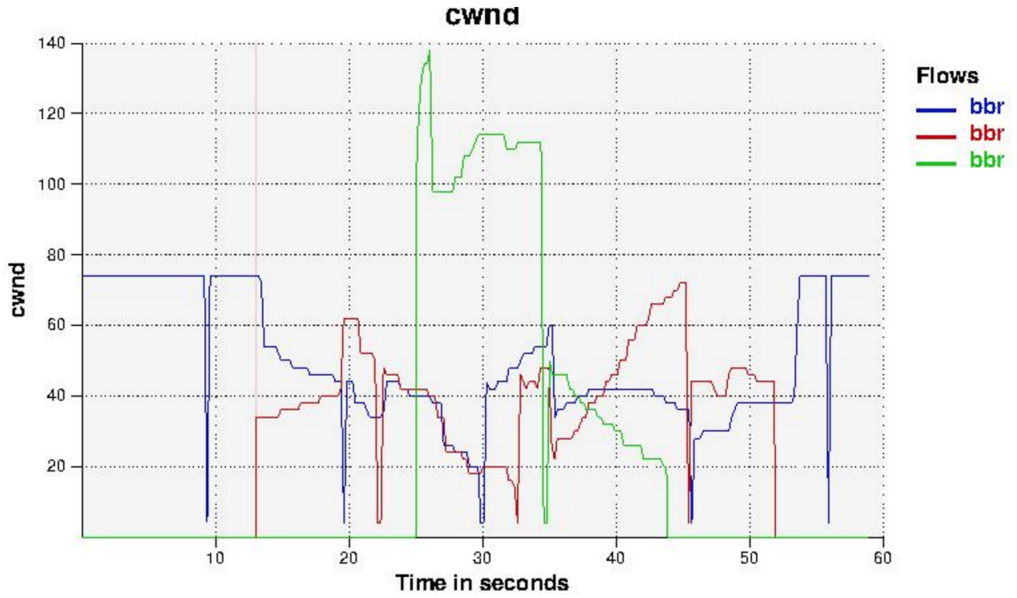

Figure 14: 10M,40ms WAN, 3 BBR Flows

4.3.1 10Mbps, 40ms RTT, 3 senders, 1 and 2 flows per sender to 1 receiver

For this scenario, we will maintain 10Mbps and 40ms RTTs, but will reduce the queue size at the bottleneck from 1000 packets down to 64 packets. Note that at 10Mbps and 40ms RTT, the BDP (bandwidth delay product is less than 40 packets.

But first, Figures 15 and 16 show cwnd graphs for 2 and 3 flows for both Cubic and BBR. 1 flow/host Cubic has 1% packet retransmissions. The cwnd are large (and so is the RTT, around 1 sec). 1 flow/host BBR has no packet retransmissions. The cwnd converges in about 10 seconds and becomes very fair. 2 flows/host Cubic has 1.2% retransmissions. The cwnd is very slow to converge. 2 flows/host BBR has 4.8% packet retransmissions. The cwnds take more than 20 secs to converge, at which point BBR behaves very well. However, for this tests the flows start within the first second or so. What would happen if the flows are more staggered? Well, if the flows start every 3 seconds (plus some randomness) and only last 18 seconds (i.e. average number of active flows is 6) the cwnd stay large and the average RTT is about 1 second.

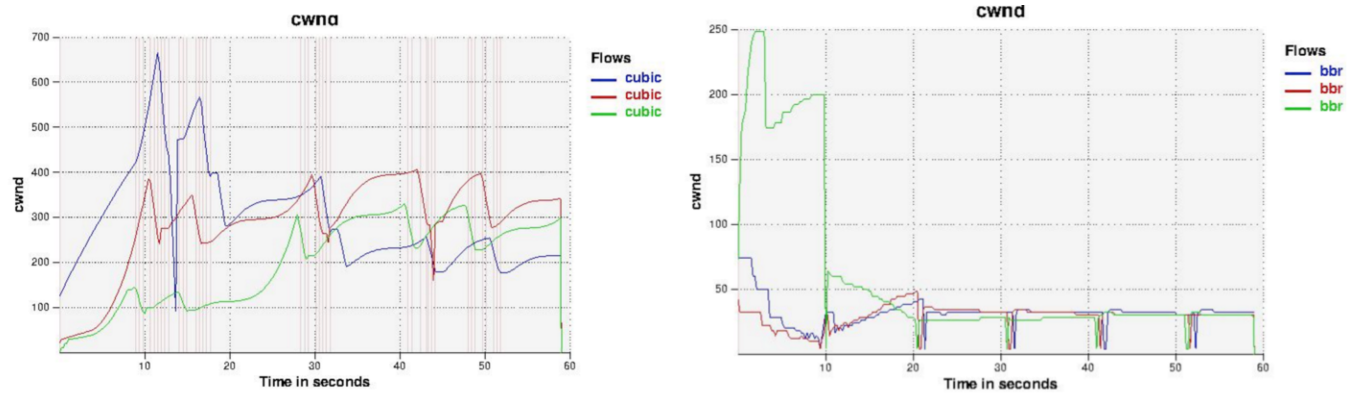

Figure 15: 10M,40ms WAN A: 2 Flows Cubic B: 2 Flows BBR

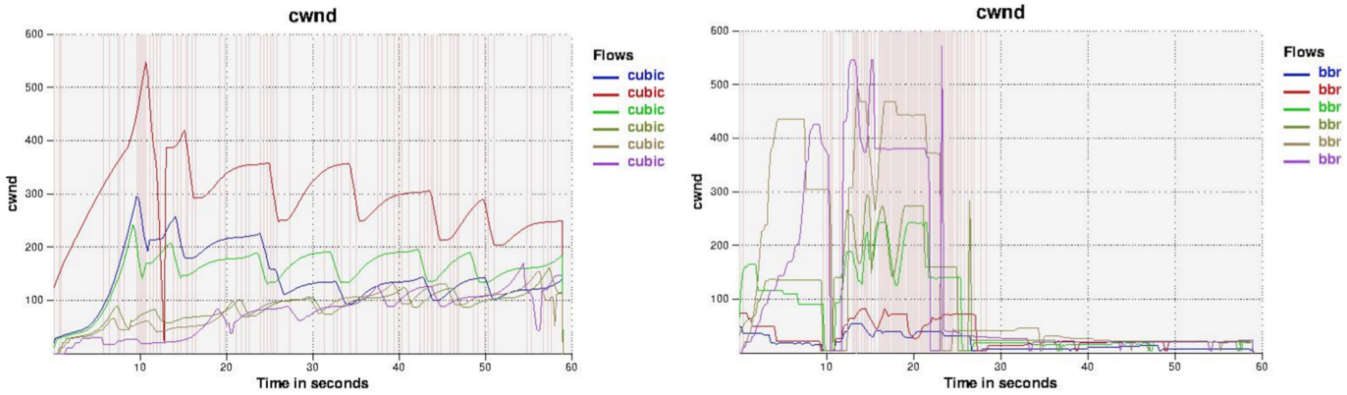

Figure 16: 10M40ms WAN A: 3 Flows Cubic B: 3 Flows BBR

Table 5 shows the effect on retransmissions for 3 sending to 1, 1 and 2 flows per host (all bulk) as we decrease the amount of buffering. It starts with 1000 buffers (25 x BDP) and decreases to 64. For the case of 2 flows per host, 6 flows total, the retransmissions range from 2.6% to 11%. Even if BBR can handle it, any other flow doing standard congestion control will suffer (get killed).

Table 5: Percent Retransmissions for 3 sending to 1, 1 and 2 flows/host, 10mbit/s, 40ms

# Buffers	3x1 Cubic	3x2 Cubic	3x1 BBR	3x2 BBR
1000 (25 x BDP)	1.0%	1.1%	0%	4.3%
500 (12 x BDP)	0.4%	0.5%	0%	3.8%
250 (6 x BDP)	0.2%	0.4%	0%	6.7%
125 (3 x BDP)	0.2%	0.4%	0.7%	2.6%
64 (1.5 x BDP)	0.3%	0.8%	3.2%	11%

Table 6 shows the efffects of reducing RTT but keeping the number of buffers at 500. The percent of retransmissions doesn't change. It seems there is enough buffering for BBR.

Table 6: Percent Retransmissions for 3 sending to 1, 1 and 2 flows/host, 40-10ms, 500 buffers

RTT	3x1 Cubic	3x2 Cubic	3x1 BBR	3x2 BBR
40ms (12 x BDP)	0.4%	0.5%	0%	3.4%
20ms (24 x BDP)	0.4%	0.6%	0%	3.1%
10ms (48 x BDP)	0.4%	0.6%	0%	3.9%

5. 10KB and 1MB RPCs

5.1 10KB and 1MB RPCs 10G 10ms WAN

Figure 17 shows the average and 99% latencies for the 1MB and 10KB RPCs for Reno, Cubic and BBR for the 10Gbps 10ms RTT WAN scenario. The 1MB average latencies look good for BBR, however the 99% latencies look a little worst. Does this mean that BBR is doing well? If we look at the 10KB latencies, we see that when the average latencies for the 1MB RPCs start be less that for Cubic, the 10KB average latencies are shooting up. That is, the 1MB RPC are taking bandwidth away from the 10KB RPCs.

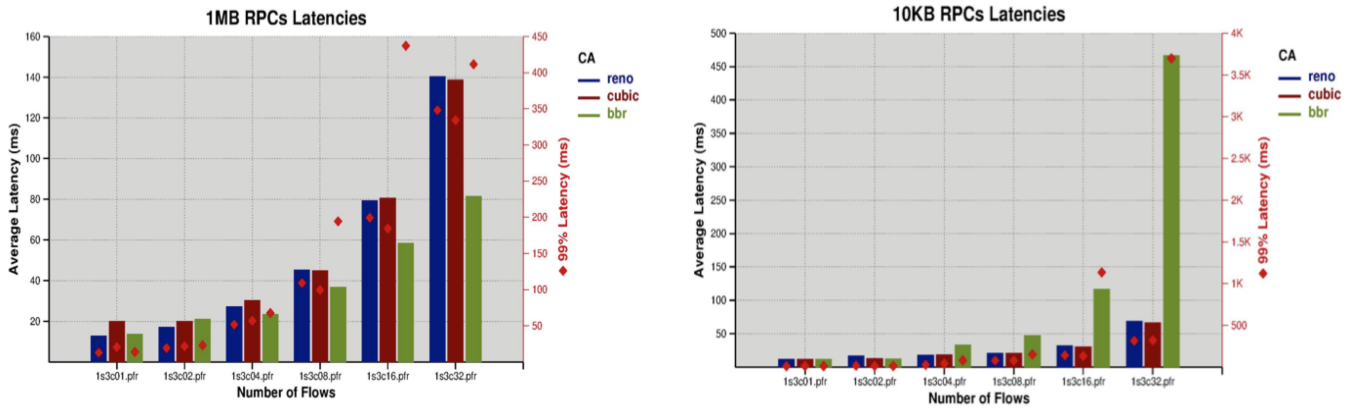

Figure 17: 10G 10ms RTT RPC Latencies

The 10KB 99% latencies look particularly bad for higher loads. The reason can be seen in Figure 18 that shows that the percent retransmissions shoot up from less than 0.1% for lower loads up to 2.7% and 5.2% for BBR. These loses are resulting in RTO back-offs for BBR resulting in 99% latencies of 1 to 3.5 seconds (vs. 200 to 400ms for Reno and Cubic).

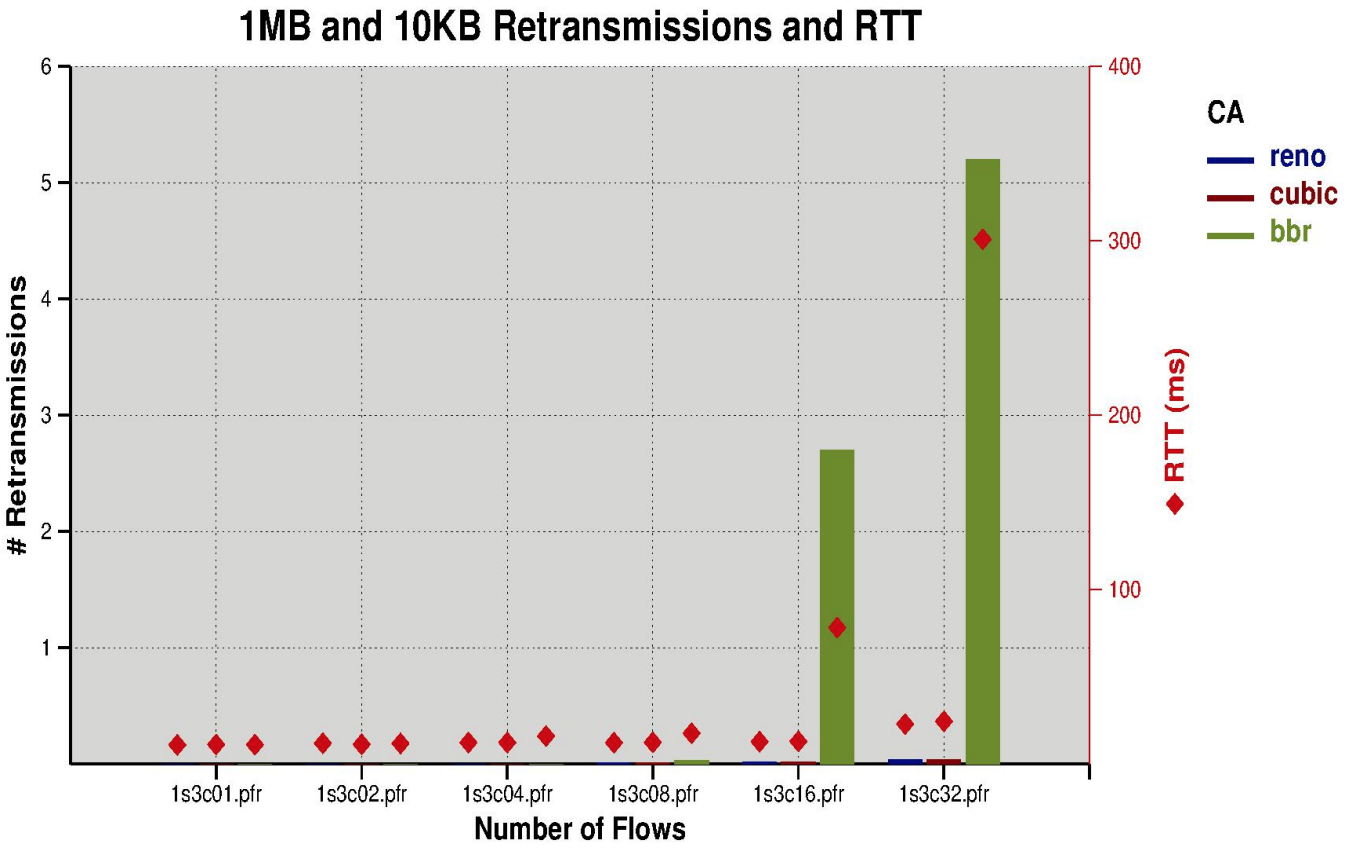

Figure 18: 10G 10ms RTT RPC Retrans and RTTs

5.2 10KB and 1MB RPCs 10Mbps 40ms WAN

Next, lets consider the 10Mbps 40ms RTT WAN scenario. Figure 19 shows the latencies for the 1MB RPCs. Since the available bandwidth is much lower, we keep the load lower (1, 2, or 4 RPC pairs per host). The average and 99% latencies for the 1MB RPCs are a little higher for BBR. However, the average 10KB latencies are much smaller for BBR, while the 99% latencies are a little higher.

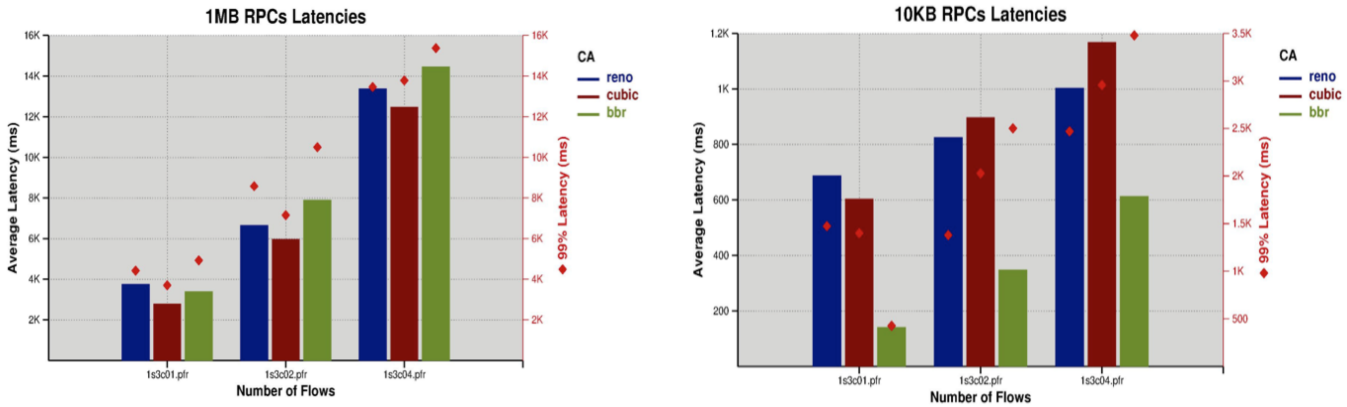

Figure 19: 10M 40ms RTT RPC Latencies

Figure 20 graphs the RTTs as seen by all the RPC flows. It shows that they stay higher for Cubic, at 600 to 700ms. The RTT graph for BBR shows that the RTTs are about as high at the beginning, at 600ms, but then decrease all the way to 200ms later on. In this case BBR converges well to good cwnd sizes, but it takes around 20 seconds. This seems to indicate that, in a static environment where flows are started at the beginning and no more flows are created later, BBR will perform well. The problem is that if flows are terminated and new flows are started, then BBR may not have enough stable time to converge.

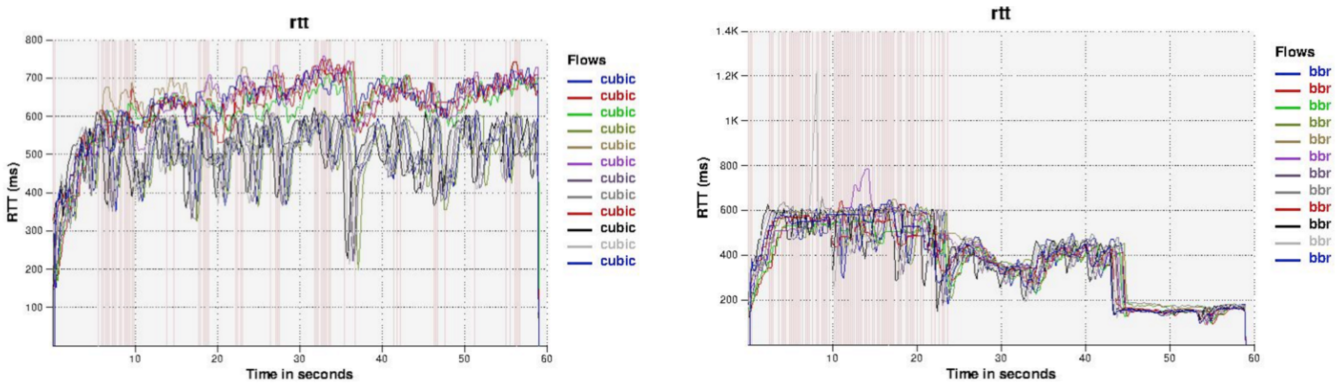

Figure 20: 10M 40ms RTT RPC RTTs for 2 flows/host

6. Conclusions

Under some scenarios, BBR is able to correctly predict the available bandwidth and adapt to it given enough time (10s of seconds in some cases) when the conditions are somewhat static. However, BBR seems to have a mode where it determines that losses are uncorrelated to congestion and, if triggered incorrectly, will create heavy (2-10%) losses since it will not decrease its cwnd due to the losses. This is of great concern since in this mode, BBR can starve non-BBR flows sharing the bottleneck and, in some cases, even other BBR traffic that did not enter this mode. I saw BBR enter this mode incorrectly in my tests quite often.