An Evaluation of BBR and its variants*

ABSTRACT

KEYWORDS

congestion control, ns3 simulation, BBR, bandwidth fairness

ACM Reference Format:
. 2019. An Evaluation of BBR and its variants. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTROCTION

Ever since the congestion collapse in internet was observed in 1986, Jocobson [1] proposed to implement the additive increase and multiplicative decrease to regulate the congestion window of TCP flow and the research works on congestion control seem endless. The congestion control for TCP bears such importance that it guarantees the internet can work in normal status and avoids congestion collapse happening again. Several algorithms have been proposed by making minor changes to the basic AIMD control law and to adapt it in some specific network environments [2][3].

As the development of the internet infrastructure, it is realized that these rate control algorithms based on AIMD do not perform well. These AIMD-like algorithms taking packet loss as link congestion indication tend to fill the pipe. Especially in today's internet, large buffers are configured in intermediate routers and the notorious bufferbloat problem [4] is introduced. It is reported in [5], users experience delays of seconds to minutes in cellular network.

In recently, developing congestion control algorithms to achieve high throughout while minimizing transmission delay at the same time becomes a new trend. BBR [5] is one typical example of such kinds. Since its release by google in 2016, BBR has gain much attention. Google first deployed BBR in its B4 wide area network and YouTube. Several works [6–8] were published to analyze its behavior either in simulated links or in real network testbed.

It was concluded that BBR can significantly improve throughput for TCP connection from these experiments with different sources. For such merit, BBR is widely applied to build VPN (virtual private network) service for rate acceleration purpose. BBR suffers from RTT unfairness issue (reported in [6][9]) and tends to overload the bottleneck link when flows coexisting, which causes considerable packets loss in links with shallow buffer.

There are some works Tsunami[1], BBRPlus[2], BBR+[10] to make some modification to these control parameters in BBR in order to gain better performance or apply it to a different network domain. Both Tsunami and BBRPlus have gain some attention. BBRPlus has won 623 stars and 280 forks on github. Whether the two are applied in real program are not clear. There are no public available report to analyze their performance too. In May 2019, google has released BBR v2 in QUIC [3] codebase with a goal to better coexistence with Reno and Cubic [11] flows. And its implementation in Linux net stack can be got at [4].

In this article, a framework is implemented and the performance of BBR algorithm and its variants (Tsunami, BBRPlus, BBR+ and BBR v2) are evaluated on simulated network environment. Since the release of BBR v2 is recent event, it seems no other works doing a full evaluation on its performance.

The transmission protocol implemented in simulation is a simplified version of QUIC protocol and only three frames (STREAM, STOP WAITTING and ACK) are applied to build a reliable transmission protocol on top UDP. The framework is about 10000 lines c++ codes and it can be available at [5].

The rest of this paper is organized as follows. The background and related works on congestion control are briefed in Section 2. The detail on these algorithms are presented on Section 3. Section 4 is the simulation results and evaluation. The conclusion and discussion is made on Section 5.

2 RELATED WORK

In October of 1986, the data rate from LBL to UC Berkeley dropped from 32kbps to 40bps, which was recorded as the first congestion collapse event in internet [1]. A series mechanisms e.g. slow start, round trip time variance estimation and the AIMD control rule were introduced to achieve network stability. These ideas are implemented in TCP Reno and it remained as the default congestion control algorithms in FreeBSD and Linux . The transport connection should obey the packet conservation principle when network system in stability: a new packet can be injected into the network only when an old packet leaves.

In Reno, the packet loss event is interpreted as network congestion signal. On every RTT, Reno sender could inject one more packet into network to probe more available bandwidth and multiplicatively reduces congestion window size by half when packet loss happens to recover the network back to normal status. Its adjustment on congestion window (w) can be summarized as Equation (1). For Reno, $\alpha = 1$ and $\beta = 0.5$.

$$w(t+1) = \begin{cases} w(t) + \frac{\alpha}{w(t)}, & \text{when ack is received} \\ \beta w(t), & \text{when packet loss is detected} \end{cases} \quad (1)$$

*Produces the permission block, and copyright information

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] https://github.com/dlxg/Linux-NetSpeed
[2] https://github.com/cx9208/bbrplus
[3] https://www.chromium.org/quic
[4] https://github.com/google/bbr/blob/v2alpha/net/ipv4/tcp_bbr2.c
[5] https://github.com/SoonyangZhang/DrainQueueCongestion

Based on the fluid model [12], a differential equation on the rate adjustment process of a AIMD flow can be deduced as Equation 2. p is packet loss rate. Let $\dot{x} = 0$, the throughout at equilibrium is $x = \frac{1}{RTT}\sqrt{\frac{1-p}{p}}$, which is usually used as rate control on these flows to be friendly to AIMD flow.

$$\dot{x} = \frac{1-p}{RTT^2} - \frac{x^2 p}{2} \tag{2}$$

The reason to choose AIMD as congestion window control is it can guarantee bandwidth allocation fairness [13] with a decentralized solution. Kelly [14] modeled the congestion control algorithm as optimization problem by introducing utilization function. The goal is to maximum the service satisfactory to each user under the constraint of the capacity (c_l) of bottleneck link. x_s denotes the sending rate of user s, $U(x_s)$ measures the satisfaction or welfare of user s.

$$\begin{aligned} \max \quad & \sum_s U(x_s) \\ \text{s.t.} \quad & \sum_{s \in S(l)} x_s \le c_l \end{aligned} \tag{3}$$

To find the optimal solution in a global view is impractical in consideration of larger scale and heterogeneity of real networks. A showdown price in introduced to decompose the primal problem into its dual form as shown. A decentralized solution is shown in Equation (4). p_l can be interpreted as price per unit bandwidth at link l. Such rate iteration is a mathematical expression to AIMD control law. The rate adjustment in Equation (4) is to converge to the optimal value by iteration. For the first time, the flow rate control problem is backup by mathematical theory. The utility maximization theory is applied for stability analysis in new designed rate control algorithms and is further developed in [15].

$$\dot{x}_s(t) = \lambda(w_s - x_s(t) \sum_{l \in L(s)} p_l) \tag{4}$$

In high speed network, once a packet loss happens, the classic AIMD algorithm will take quite long time to recover back to the congestion window before the multiplicative reduction action and has a low utilization of bandwidth resource. Several algorithms e.g. STCP[16], HSTCP[17], BIC[18], Cubic[11] have been proposed to remedy such problem. The congestion window increase and decrease behavior is modified to adapt for high speed network in these solutions. STCP and HSTCP introduce several RTT unfairness, with which the flow with short RTT obtain more bandwidth. In BIC, the congestion control is viewed as a binary search problem to enable aggressive bandwidth probe. The congestion window grows to the middle point of w_{min} and w_{max}. w_{max} is the congestion window before the last fast recovery and w_{max} is the window value after the fast recovery. When the packet loss is detected, the middle point is assigned to w_{max}, and it is taken as the new value of w_{min} otherwise. Cubic is an update version of BIC. A cubic function is introduced for window adjustment. It can be solve the RTT unfairness issue since its congestion window increase is only depended on the time between two consecutive congestion events. The Cubic algorithm achieves better performance than AIMD, and it has been the default configuration in Linux net stack until now.

There are algorithms taking delay as congestion signal. The delay based algorithms can prevent queue from building up, make highly use of channel resource and maintain throughput stability. Once the delay has exceeded of some threshold, the congestion window will be reduced to alleviate congestion. Vegas [19] and Fast [20] are belonging to such kind. Even Vegas can achieve quite low queue occupation, it get starvation when sharing links with loss based algorithms. For such reason, it is not widely applied in real network. There are other protocols TCP-LP [21] and LEDBAT [22] following the working mechanism of AIMD and taking delay as congestion signal. These two take themselves junior to TCP flows and actively yield bandwidth to when sharing links with high priority TCP flows.

Some other algorithms take both delay and packet loss to indicate link congestion. Veno [23] takes backlog queue delay to differential random loss and congestion loss to improve TCP throughput performance in wireless network. In Illinois [24], the window adjustment parameters α and β are dynamically changed with delay. When delay signal is small, a large value of α is applied for fast convergence. Compound [25] is the rate control algorithm in Windows operating systems. The congestion widow is increased fast when bandwidth resource is available.

Attempts are made to the coexistence of delay based congestion control with delay base congestion control. When facing packet loss, CHD [26] will back off the congestion window with probability. When the backlog queue delay is above q_{th}, the possibility to reduce congestion window decreases in CHD to gain better competiveness with buffering filling flows. CDG [27] used delay gradient to infer link buffer in full, empty, rising, and falling status. CDG has the ability to tolerate non-congestion related loss and only backs off for congestion related packet loss.

Given the delay problem introduced by loss based algorithms, to design congestion control with high throughput and low delay becomes trend. Serval works e.g. Sprout [28], PCC [29], BBR [5] and Copa [30] were designed with such porpose.

Sprout, Verus [31], ExLL [32] and C2TCP [33] are mainly designed for cellular network. Legacy algorithms have degenerate performance in links with highly variable capacities and non-congested packet loss. DCTCP [34] and Timely [35] are proposed to achieve low message delivery delay and high throughout for datacenter networks. Some works e.g. Remy [36], QTCP [37] and TCP-Drinc [38] apply reinforce learning in rate control. They are evaluated in simulated paltform, whether such methods can be applied in network stack need further investigation.

As the popular of real time communication applications, deploying rate control algorithms on multimedia traffic is a necessary to avoid congestion and to promote fair bandwidth allocation. GCC [39], NADA [40], and SCReAM [41] are optimized for interactive multimedia transmission. GCC is the default rate control algorithm in WebRTC [6] project, which enables video communication among browsers.

3 ALGORITHM DETAIL

Loss based algorithms were designed to avoid congestion, but it inevitably leads the network into overuse. The capacity of a bottleneck link is denoted as $BtlBW$. When only one flow is presence in this link and it packet sending rate is less than $BtlBW$, the packet

[6] https://webrtc.org/

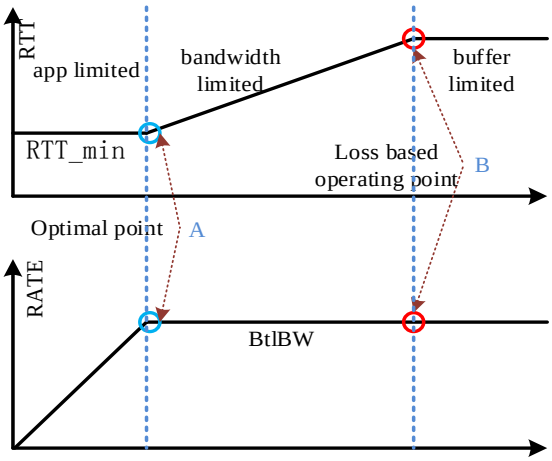


Figure 1: Congestion control operating point, based on [5]

round trip delay is the minimum RTT_{min} as shown in Figure 1. But the flow with loss based rate control will keep increase its rate when no packet loss happens. The rate finally exceeds $BtlBW$ and these extra sent packets will be buffered at routers. The packets can be received with the full rate $BtlBW$ at the cost of increased delay. The points on left edge of the bandwidth limited region achieve the same throughout but lower delay compared with loss based operating point. The optimal point in congestion control is named after Kleinrock [42], maximizing throughput while minimizing delay and loss. However, the Kleinrock's point can not be reached with distributed algorithm, proved in [43]. BBR is proposed to get close to the optimal point.

3.1 BBR

In BBR, the bandwidth can be estimated on each received acknowledge packet. When a new packet is sent out, the packet state information ($total\_byte\_acked$, $last\_acked\_packet\_ack\_time$) is recorded. $total\_byte\_acked$ counts for the bytes that successfully received by its peer. $last\_acked\_packet\_ack\_time$ is the received time of last acknowledgement packet. When a $sent\_packet$ is acknowledged at now, a new bandwidth estimation sample can be calculated as Equation (6). The BW of the channel is the maximum bw_es within 10 RTTs. And the minimal rtt RTT_{min} is monitored during the whole phase.

$$\begin{aligned} &last\_acked\_packet\_ack\_time = now \\ &total\_byte\_acked += sent\_packet.bytes \\ &\Delta t = now - sent\_packet.last\_acked\_packet\_ack\_time \\ &\Delta delivered = total\_byte\_acked \\ &- sent\_packet.total\_byte\_acked \end{aligned} \tag{5}$$

$$bw\_es = \frac{\Delta delivered}{\Delta t} \tag{6}$$

There are four control states StartUp, Drain, ProbeBW, and ProbeRTT in BBR as shown in Figure 2. In each control state, the packet sending rate ($pacing\_rate$) is the product of $pacing\_gain$ and BW.

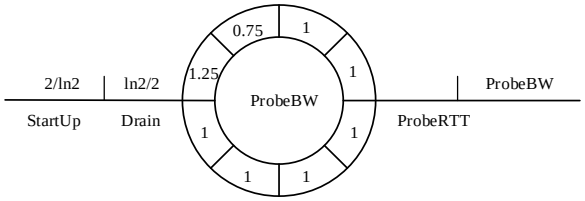


Figure 2: Control states in BBR

The states StartUp and Drain are applied at session initial phase in BBR. The startup state is quite similar to slow start in TCP. In StartUp, $pacing\_gain$ is $\frac{2}{ln2}$ to double the inflight packets at each RTT to let sender probes the maximum available bandwidth. When newly estimated bandwidth is 1.25 times less than the previous value and such circumstance lasts for 3 times, the pipe seems to be get fully filed and the state is changed into Drain state. The $pacing\_gain$ in Drain is $\frac{ln2}{2}$ to decrease the sending rate below bw. Until the inflight packets match BDP ($BW * RTT\_min$), state is changed from Drain to ProbeBW.

During ProbeBW state, The $pacing\_gain$ cycles in 8 RTTs with different values ($kPacingGain[] = [1.25, 0.75, 1, 1, 1, 1, 1, 1]$). The probe up phase with 1.25 gain is to increase the sending rate to probe more available bandwidth, and probe down phase with 0.75 gain is to get rid of the excess queue which may be accumulated in the probe up phase.

The $cwnd$ is set as $2 * BDP$ in ProbeBW to guarantee enough packets can be sent during probe up phase. If $cwnd$ is set exactly equal with BDP, the acknowledgement clocking is affected by disturbance in real network, new packets can not be sent out during probe up once the $cwnd$ is exhausted and a small estimated bandwidth is got. If RTT_{min} is not sampled again within 10 seconds, the link is deemed falling into congestion, and ProbeRTT state id applied. The $cwnd$ is set as 4*MSS. Until the inflight packets size is less than 4*MSS, new packets are injected into the network and ProbeRTT will last at most 200 milliseconds. In ProbeRTT, the inflight packets are nearly totally drained from links and a new RTT_{min} value is sampled.

There are two different versions to update the $pacing\_gain$ from probe down phase to probe cruise phase in ProbeBW. The first is to increase the cycle offset when the prove down phase holds for essentially 1 RTT_{min}. The second is to increase cycle offset only when the inflight packets Less than or equal to the target BDP to achieve lower queue delay than the first form. Most published works only analyse the performance of BBR in the first form. In later part, both versions will be evaluated. For convenience, the second form is named as BBR'.

3.2 BBRPlus

BBRPlus was first introduced in a blog [7]. In the origin version of BBR, the duration of probe down lasting 1 RTT_{min} introduces considerable latency. When the network system changes a lot, the fixed probe cycle length may not adapt well. The probe cycle length is randomized from 2 to 8 (line 4 in Algorithm 1) in BBRPlus. The probe down phase exists only when the inflight packets match

[7] https://blog.csdn.net/dog250/article/details/80629551

the estimated *BDP* (line 11-12 in Algorithm 1). Such change is to improve fairness and reduce packet loss when multiple flows sharing a bottleneck.

The procedure to update *pacing_gain* in BBRPlus is shown in Algorithm 1. *kGainCycleLen* is 8 and *CYCLE_RAND* equals 7. When there is packet loss event, the probe up phase exists earlier (line 14 in Algorithm 1).

Algorithm 1 UpdateGainCyclePhase

Input:
the timestamp (now), $inflight$, has_loss

1: $elapsed \leftarrow now - cycle\_mstamp\_$
2: **if** $elapsed > cycle\_len\_ * RTT_{min}$ **then**
3: $cycle\_mstamp\_ \leftarrow now$
4: $cycle\_len\_ \leftarrow kGainCycleLen - rand()\%CYCLE\_RAND$
5: $pacing\_gain \leftarrow 1.25$
6: **return**
7: **end if**
8: **if** $pacing\_gain == 1.0$ **then**
9: **return**
10: **end if**
11: **if** $pacing\_gain < 1.0$**and**$inflight \leq BDP$ **then**
12: $pacing\_gain \leftarrow 1.0$
13: **end if**
14: **if** $elapsed > RTT_{min}$**and**$(inflight > 1.25 * BDP$**or**$has\_loss)$ **then**
15: $pacing\_gain \leftarrow 0.75$
16: **end if**

3.3 BBR+ and Tsunami

The performance of Cubic and BBR is tested over LTE on high speed rail (HSR) in [10]. The authors concluded that BBR achieves suboptimal performance in networking environment where both bandwidth and RTT change rapidly. The bandwidth probe strategy and RTT estimation do not adapt well to the network dynamics in HSR situation. The sequence to update *pacing_gain* is set to be more radically as [1.5, 0.5, 1.5, 0.5, 1.5, 0.5, 1.5, 0.5] in BBR+. The constant RTT_{min} is too conservative over the last 10 seconds in BBR, and a compensation is added according to Equation (7) in HSR environment. λ^2 is the shape parameter of gamma distribution. They observed the traced RTT values approximately follow a shifted gamma distribution with a fat tail. In our simulation, the compensation part on the RTT_{min} is not implemented.

$$RTprop = RTT_{min} + \lambda\sqrt{Var(RTT)} \tag{7}$$

From the perspective of egoism, Tsunami applies the sequence [1.5, 0.75, 1.25, 1.25, 1.25, 1.25, 1.25, 1.25] for gain value update during ProbeBW state in order to gain higher throughput. It gains 264 stars and 136 forks on github. As its name claims, when there is extra bandwidth resource available, Tsunami may quick occupy it. If when the capacity of the bottleneck is fully occupied, Tsunami will cause quite high packet loss rate. Such rate adjustment behavior is harmful to other flows without benefiting itself.

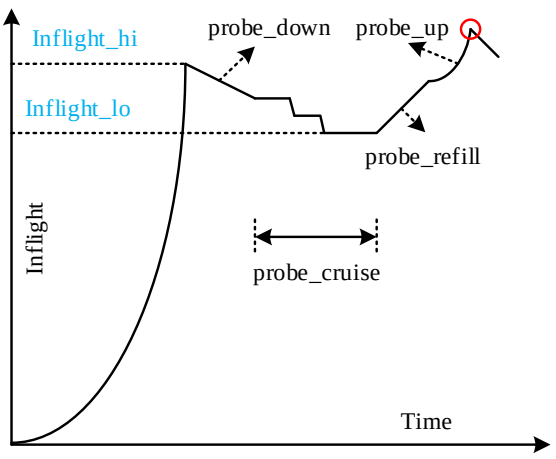


Figure 3: BBR v2 flow life cycle, from [44]

3.4 BBR v2

BBR flow will send packets at the full estimated bandwidth and packet loss is not exploited to indicate link congestion. If the queue length at the bottleneck is smaller than $1.5 * BDP$ [44], multiple BBR flows will cause high packet loss. Reno/CUBIC flows gain low throughput when share bottleneck with BBR flows. BBR v2 [44] is proposed to solve these issues in BBR v1. BBR v2 is claimed that it can make better coexistence with Reno/CUBIC flows and achieves low queue delay.

BBR v2 takes packet loss into its control logic. The life cycle of BBR v2 flow is shown in Figure 3. When the estimated bandwidth dost not exceed the target for 3 times, the sender assumes reaching the full bandwidth in BBR v1. Besides that, condition on packet loss is added to exist from StartUp to Drain: the number of loss packet in a round exceeds 8 and packet loss rate exceeds *loss_threshold* (0.02). Such condition is applied to avoid excessive packet loss. If the condition on packet loss holds true, the calculated *BDP* is assigned to $inflight\_hi$. When a new acknowledge packet arrives, $inflight\_lo$ is updated as Equation (8). Here, $\Delta delivered$ is calculated in the same way in Equation (5) and *kBeta* is 0.3.

$$\begin{aligned} inflight &= \Delta delivered \\ inflight\_lo &= \max(inflight, inflight\_lo * (1 - kBeta)) \end{aligned} \tag{8}$$

In ProbeBW state, the working mechanism of BBR v2 is quite different from BBR v1. The phases (probe_up, probe_down, probe_cruise) switching is no longer depended on the time interval RTT_{min}. *cwnd* is also not set as $2 * BDP$ and is related with $inflight\_lo$ and $inflight\_hi$. $inflight\_hi$ is updated if the inflight packets are too high (inflight_too_high), in which the loss packet rate exceeds *loss_threshold* in last round. In the probe_down phase, the *pacing_gain* is 0.75, the phase will be switched to probe_cruise if the inflight packets are drained to *BDP* or the condition $inflight\_too\_high$ holds true. The *cwnd* in probe_cruise is calculated by Equation (9). *kHeadRoom* is 0.15. $inflight\_hi$ indicates the channel is in dangerous area. To leave headroom to *cwnd* is to alleviate link congestion to some extent. The *interval* for probe_cruise is randomized from 2 seconds to 3 seconds. If duration in probe_cruise phase exceeds the

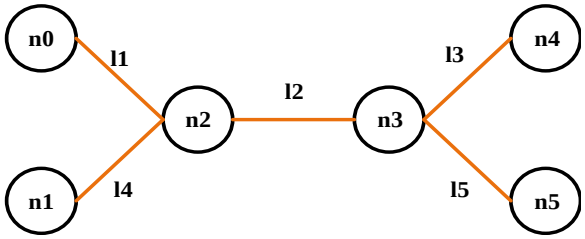


Figure 4: Network topology

interval, a probe_refill phase is applied as shown in Figure 3. *cwnd* is set as $inflight\_hi$ to increase the inflight packets in a round. The probe_refill is to make preparation for probe_up.

$$\begin{aligned} inflight\_headroom &= inflight\_hi * (1 - kHeadRoom) \\ cwnd &= \min(inflight\_lo, inflight\_headroom) \end{aligned} \tag{9}$$

In probe_up phase, *cwnd* is increases exponentially per round:1, 2 ,4, 8.... It makes a fast probe to if extra bandwidth available. Once lost bytes are too much, probe_down phase is applied to get rid of excess queue, as the red ring shows in Figure 3. Algorithm 2 is to exponentially increase $inflight\_hi$ per round. In ProbeRTT, *cwnd* is reduced by half in v2 to remedy the throughput variation.

Algorithm 2 ProbeInflightHighUpward

Input:

bytes_acked, is_round_end

$probe\_up\_acked+ = bytes\_acked$
if $probe\_up\_acked \geq probe\_up\_bytes$ **then**
  $delta = \lfloor \frac{probe\_up\_acked}{probe\_up\_bytes} \rfloor$
  $probe\_up\_acked- = delta * probe\_up\_bytes$
  $inflight\_hi \leftarrow inflight\_hi + delta * MSS$
end if
if is_round_end **then**
  $growth \leftarrow 1 << probe\_up\_rounds$
  $probe\_up\_rounds \leftarrow \min(30, probe\_up\_rounds + 1)$
  $probe\_up\_bytes \leftarrow \lfloor \frac{cwnd}{growth} \rfloor$
  $probe\_up\_bytes \leftarrow \max(MSS, probe\_up\_bytes)$
end if

4 EVALUATION

These algorithms are evaluated on ns3.26 [8] platform. A dumbbell topology as shown in Figure 4 is built.

4.1 Intra protocol fairness

To test whether this algorithms can guarantee bandwidth allocation property, four flows are created from source *n*2 to destination *n*3. These parameters in Table 1 to configure link *l*2 are bandwidth (in unit of Mbps), one way propagation delay (in unit of milliseconds) and queue length in nodes. There are total 11 experiments. In each running case, these flows follow a same rate control algorithm. Each simulation process lasts about 400 seconds. The time points to send packets into the network of the four flows are different. The first flow starts at 0 and ends at 400s, the life length of the second flow is 40s to 400s, the third flow is 80s to 200s and the fourth flow is 120s to 300s. At the sender side, when a new packet can be sent out, the rate of the congestion controller is traced. The packet sent time is tagged into ns packet object for receiver to computer one way transmission delay. The one way transmission delay is an indicator to the occupied buffer status in routers. Besides one way delay, at received side, the length of received packet is also recorded.

The average transmission delay of all flows in each experiment is calculated and the results are shown in Figure 5. The results on

[8] https://www.nsnam.org/

Table 1: the configuration of *l*2

Case	Bandwidth	Propagation delay	Queue length
1	5Mbps	50ms	5Mbps*100ms
2	5Mbps	50ms	5Mbps*150ms
3	5Mbps	50ms	5Mbps*200ms
4	6Mbps	50ms	6Mbps*100ms
5	6Mbps	50ms	6Mbps*150ms
6	7Mbps	50ms	7Mbps*150ms
7	7Mbps	100ms	7Mbps*300ms
8	8Mbps	100ms	8Mbps*200ms
9	8Mbps	100ms	8Mbps*300ms
10	10Mbps	50ms	10Mbps*150ms
11	10Mbps	50ms	10Mbps*200ms

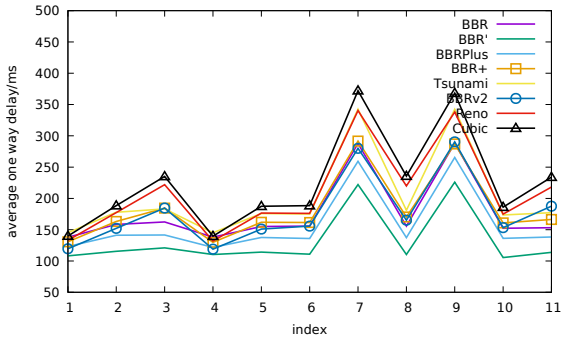


Figure 5: Average one way transmission delay

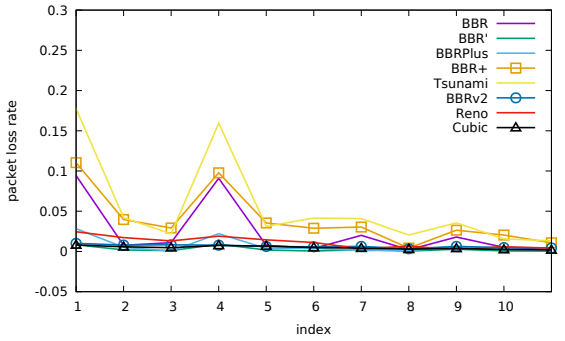
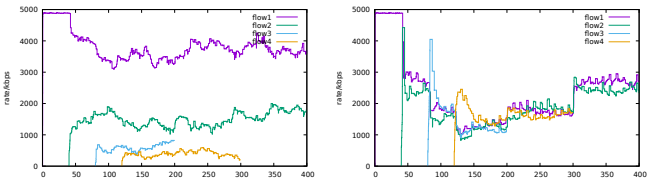


Figure 6: Average packet loss rate

average packet loss rate is shown in Figure 6. Two other buffer filling algorithms Reno and Cubic are also tested.

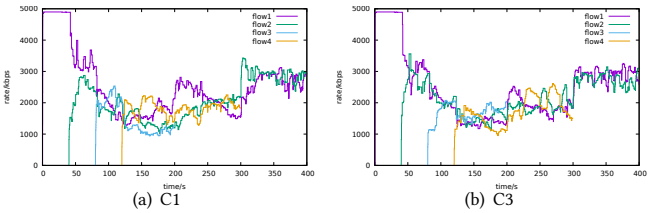
Due to the limitation of space, only two cases work as example for further analysis. The link buffer is configured as $1 * BDP$ in Case 1 and $2 * BDP$ in Case 3. In shallow buffer case, the four BBR flows are not reach the bandwidth fairness line as shown in Figure 7(a). With BBR flows presence, there is considerable packet loss rate (about 9%) in Case 1 as shown in Figure 6. Such high packet loss rate will impact the bandwidth estimation at sender side. In case 3, the bandwidth allocation fairness is achieved. But when a new flow is initialized, it tends to make over estimation on the bandwidth during the StartUp state, as the steep spike shown in Figure 7(b) in flow2 and flow3. Such rate spikes will lead the network into congestion and introduce packet loss. It is the reason for higher packet loss rate (1%) for Case 3 with longer link buffer compared with Case 2, in which the average packet loss rate is 0.7%.



(a) C1 (b) C3

Figure 7: Rate dynamics of BBR flows

BBR' will drain the inflight packets to match the estimated *BDP* in probe down phase. It can achieve the lowest queue delay as shown in Figure 5 in all tested algorithms, and lower packet loss rate compared with BBR. It also achieve better bandwidth allocation fairness than BBR in shallow buffer case as shown in Figure 8(a). But it introduces rate variation. The rate of BBR' flow is not quite stable as BBR flow, as shown in Figure 8(b) and 7(b).

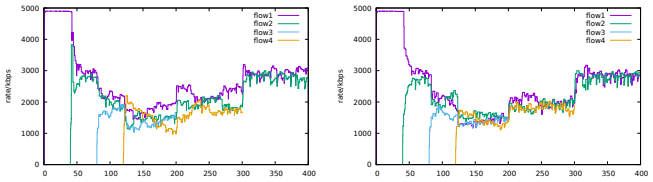


(a) C1 (b) C3

Figure 8: Rate dynamics of BBR' flows

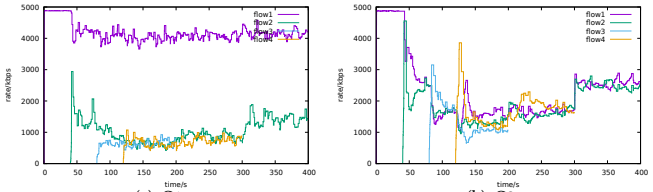
BBRPlus can achieve the second low average transmission delay. The average packet loss rate is also quite low. At each stage, the throughputs of each flow are quite close. There are some small spikes during the rate adjustment process as shown in Figure 9.

Both Tsunami and BBR+ suffer from the bandwidth allocation fairness in links with shallow buffer. The way of Tsunami to adjust rate will get the buffer fully occupied. Tsunami flows have the highest packet loss rate and the average transmission delay is quite high. The rate High rate variation can be observed of BBR+ flows in Figure 11(b).



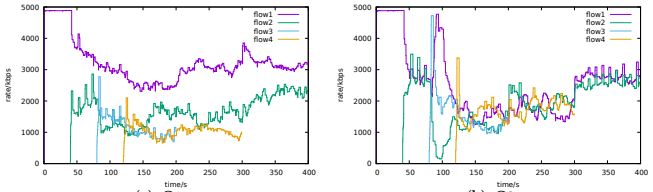
(a) C1 (b) C3

Figure 9: Rate dynamics of BBRPlus flows



(a) C1 (b) C3

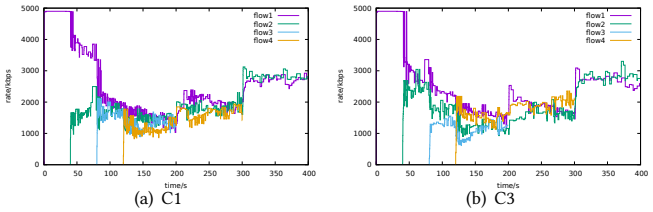
Figure 10: Rate dynamics of Tsunami flows



(a) C1 (b) C3

Figure 11: Rate dynamics of BBR+ flows

BBR v2 flows can maintain well bandwidth allocation fairness. The rate adjustment is quite frequency in BBR v2. That is the result of the balance between probing more bandwidth and avoiding link congestion. The lower queue delay in some test cases is lower than BBR but is still higher than BBR' in all test cases.



(a) C1 (b) C3

Figure 12: Rate dynamics of BBR v2 flows

4.2 RTT unfairness

As indicated in several reports, BBR flows traversing the bottleneck link with same RTT converge to a fair share bandwidth line within their group, but these flows with longer RTT can achieve higher

throughput, contrary to the classic Reno algorithm. Reno algorithm favors towards shorter RTT flows. The Reno flows with shorter RTT can get acknowledge packet more quickly and the congestion window can be increased faster in these flows. Such RTT unfairness property in BBR can be easily manipulated by malicious receiver for data transmission acceleration by delaying the acknowledge packets.

The dumbbell topology is applied to reproduce the RTT unfairness phenomenon of BBR algorithm. The bottleneck link is $l2$. The capacities of $l1, l3, l4, l5$ are 100 Mbps. The propagation delay values for $l1, l3$ are 10 milliseconds. 20 milliseconds is configured for $l4, l5$. Nine experiments cases are tested. The parameters to configure $l2$ are in each case given in Table 2. Two flows are applied. $flow1$starts from $n0$ to destination $n4$ (path1) and $flow2$ starts from $n1$ to destination $n5$ (path2). The max round trip propagation delay of path1 and path2 is denoted as RPT_{max}. $Qdelay$ is $1.5 * RPT_{max}$. In other word, the buffer length of bottleneck link is $1.5 * BDP$. The buffer length of Other links is $BW * Qdelay$.

The running time of each simulation lasts 200 seconds. Both flows are running in the whole simulation time. The average throughout is calculated as Equation (10). $bytes$ is the length of all received packets. The jain's fairness index [45] is exploited to indicate how fair the bandwidth is shared when flows competing for bandwidth resource. The way to compute the jain's fairness index is shown in (11). The closer Jain's fairness index is to 1, the better in terms of bandwidth allocation fairness.

In this part, only two flows are involved in each case. As we found in some case, Jain's fairness index may not work well to reflect the throughput variance and the throughput ratio of the two flow is computed as Equation (12). The final results are shown in Table 3. $\overline{x}_1$ and $\overline{x}_2$ are the average throughput (in unit of kbps) of $flow1$ and $flow2$ in respectively.

$$\overline{x} = \frac{bytes}{duration} \tag{10}$$

$$J = \frac{(\sum_i^n x_i)^2}{n * (\sum_i^n x_i^2)} \tag{11}$$

$$r = \frac{\overline{x}_{max}}{\overline{x}_{min}} \tag{12}$$

Some conclusions can be made based on Table 3. BBR $flow2$ with longer RTT acquires more than four times the throughput of $flow1$ in Case7. As the RTT ratio gets smaller in different cases, the rates of two flows are get closer and the jain's fairness is also increase. The reason for RTT unfairness is related to buffer occupation in intermediate routers. The flow with large RTT will send more packets out when the bottleneck is already in congestion. It will get larger bandwidth estimation than the flow with shorter RTT flow.

In BBR' and BBRPlus cases, the throughput ratios are not so large as in BBR cases. The RTT unfairness issue has been significantly improved. Both algorithms will drain the inflight packet to match the estimated BDP in the probe down phase and lower queue delay can be achieved as shown in Figure 5. The action to drain to the target will give other flows an opportunity to probe more available bandwidth. That's the reason behind such improvement.

Table 2: the configuration of $l2$ to test RTT unfairness

Case	Bandwidth	Propagation delay	Q(bw*Qdelay)
1	4Mbps	10ms	4Mbps*150ms
2	4Mbps	20ms	4Mbps*180ms
3	4Mbps	30ms	4Mbps*210ms
4	6Mbps	10ms	6Mbps*150ms
5	6Mbps	20ms	6Mbps*180ms
6	6Mbps	30ms	6Mbps*210ms
7	8Mbps	10ms	8Mbps*150ms
8	8Mbps	20ms	8Mbps*180ms
9	8Mbps	30ms	8Mbps*210ms

The RTT unfairness is not so severe in BBR v2 test cases as BBR. The results on Reno verify the conclusion that the flow with shorter RTT can gain higher throughout.

4.3 Channel utilization

The channel utilization of these congestion control algorithms is tested in links with random loss. The configuration of the point to point channel ($n2$ to $n3$) remains unchanged as in Table 1. Only five cases (C2, C5, C7, C9, C10) are involved and the configured buffer length of the bottleneck is $1.5 * BDP$. In each case, the random packet loss rates are 1%, 3% and 5%. The four flows are running in the whole simulation process.

The channel utilization of all flows is calculates as Equation (13). $bytes_i$ is the length of all received packets at application layer of flow i. cap is the bandwidth of the bottleneck link and $duration$ is the simulation running time. The final results are given in Table 4.

When no random packet loss is existence, these algorithms achieve channel bandwidth utilization above 90%. BBR v2 can achieve channel utilization about 97%, similar to the two buffer filling algorithms Reno and Cubic. The link utilization of the three algorithms (BBR, BBR' and BBRPlus) is less affected by random packet loss. With 5% random packet loss rate, the channel utilization of BBR v2 flows is quite low in C7 and C9.

$$util = \frac{\sum_i bytes_i}{cap * duration} \tag{13}$$

4.4 Responsiveness

In cellular access network or wireless network, channel throughput can present drastic change in a short time span due to noise interference and fading. The performance of BBR is tested in cellular network in [46]. Here, the point to point link $n2$ to $n3$ is used to test whether these algorithm can make fast response to link bandwidth change. The link capacity is changed every 50 seconds from 1Mbps to 4Mbps to simulate link throughout change. The propagation delay is 50 milliseconds. Two flows are involved and the simulation process lasts 400 second.

The rate adjustment process of each flow is shown in Figure 13. All these algorithms can adapt well the throughput of the flow as the link rate changes. The average packet loss rate, average packet transmission delay and channel utilization are calculated in Table 5. Since the buffer length is configured as $1.5 * 4Mbps * 100ms$, the

Table 3: The calculated results in RTT unfairness simulation

algo	1	2	3	4	5	6	7	8	9
	($\overline{x_1}$, $\overline{x_2}$, jain's fairness index, r)								
BBR	840, 2970, 0.76, 3.53	1029, 2767, 0.83, 2.69	1177, 2606, 0.88, 2.21	1154, 4543, 0.74, 3.94	1511, 4166, 0.82, 2.76	1703, 3947, 0.86, 2.32	1421, 6164, 0.72, 4.34	1944, 5611, 0.81, 2.89	2259, 5269, 0.86, 2.33
BBR'	1668, 2159, 0.98, 1.29	1712, 2098, 0.99, 1.23	1800, 2001, 1.0, 1.11	2498, 3221, 0.98, 1.29	2508, 3168, 0.99, 1.26	2729, 2964, 1.0, 1.09	3329, 4273, 0.98, 1.28	3344, 4216, 0.99, 1.26	3517, 4055, 0.99, 1.15
BBRPlus	1713, 2105, 0.99, 1.23	1785, 2017, 1.0, 1.13	1774, 2021, 1.0, 1.14	2594, 3126, 0.99, 1.21	2580, 3098, 0.99, 1.2	2656, 3024, 1.0, 1.14	3381, 4223, 0.99, 1.25	3394, 4164, 0.99, 1.23	3389, 4182, 0.99, 1.23
Tsunami	902, 2914, 0.78, 3.23	1103, 2706, 0.85, 2.45	1336, 2462, 0.92, 1.84	1226, 4486, 0.75, 3.66	1621, 4080, 0.84, 2.52	1792, 3873, 0.88, 2.16	1418, 6182, 0.72, 4.36	1977, 5601, 0.81, 2.83	2360, 5181, 0.88, 2.19
BBR+	1345, 2481, 0.92, 1.84	1500, 2305, 0.96, 1.54	1682, 2116, 0.99, 1.26	2073, 3653, 0.93, 1.76	2235, 3436, 0.96, 1.54	2495, 3181, 0.99, 1.27	2543, 5054, 0.9, 1.99	2999, 4556, 0.96, 1.52	3334, 4225, 0.99, 1.27
BBRv2	1695, 2182, 0.98, 1.29	1663, 2212, 0.98, 1.33	2025, 1850, 1.0, 1.09	2474, 3343, 0.98, 1.35	2337, 3469, 0.96, 1.48	3338, 2473, 0.98, 1.35	3649, 4102, 1.0, 1.12	4134, 3602, 1.0, 1.15	4465, 3279, 0.98, 1.36
Cubic	1716, 2166, 0.99, 1.26	1625, 2257, 0.97, 1.39	2429, 1452, 0.94, 1.67	2845, 2978, 1.0, 1.05	3239, 2583, 0.99, 1.25	3501, 2320, 0.96, 1.51	3501, 4263, 0.99, 1.22	3524, 4239, 0.99, 1.2	4824, 2936, 0.94, 1.64
Reno	2237, 1645, 0.98, 1.36	2062, 1820, 1.0, 1.13	2239, 1642, 0.98, 1.36	3406, 2417, 0.97, 1.41	3393, 2429, 0.97, 1.4	3370, 2451, 0.98, 1.37	4219, 3546, 0.99, 1.19	4154, 3608, 1.0, 1.15	4331, 3429, 0.99, 1.26

Table 4: channel utilization with random loss

algo	rand loss(%)	C2	C5	C7	C9	C10
BBR	0	0.95	0.95	0.94	0.94	0.95
	1	0.95	0.94	0.93	0.92	0.94
	3	0.92	0.92	0.91	0.90	0.92
	5	0.91	0.90	0.89	0.89	0.90
BBR'	0	0.95	0.95	0.94	0.94	0.95
	1	0.94	0.94	0.93	0.93	0.94
	3	0.92	0.92	0.91	0.91	0.92
	5	0.90	0.90	0.89	0.89	0.90
BBRPlus	0	0.95	0.95	0.95	0.94	0.95
	1	0.94	0.94	0.93	0.92	0.94
	3	0.92	0.92	0.91	0.90	0.92
	5	0.90	0.90	0.89	0.88	0.90
BBR v2	0	0.97	0.97	0.97	0.97	0.97
	1	0.96	0.96	0.95	0.96	0.96
	3	0.94	0.93	0.87	0.84	0.92
	5	0.91	0.91	0.62	0.51	0.81
Cubic	0	0.97	0.97	0.96	0.96	0.97
	1	0.96	0.96	0.70	0.62	0.92
	3	0.93	0.86	0.38	0.33	0.53
	5	0.76	0.64	0.28	0.25	0.40
Reno	0	0.97	0.97	0.96	0.96	0.97
	1	0.96	0.96	0.85	0.75	0.96
	3	0.94	0.92	0.46	0.40	0.64
	5	0.87	0.78	0.34	0.30	0.48

Table 5: The statistical results of the responsive experiments

	loss	$\overline{owd}$	utility
BBR	0.001	170.21	0.95
BBR'	0.001	117.35	0.95
BBRPlus	0.001	139.15	0.95
Tsunami	0.003	182.37	0.95
BBR+	0.005	186.19	0.95
BBrv2	0.006	224.58	0.97
Cubic	0.009	272.23	0.97
Reno	0.020	257.26	0.97

loss rate of flows with BBR like algorithms is quite small (below 1%). The packet transmission delay values are higher in Tsunami, BBR+ and BBRv2. BBR' can achieve the lowest packet transmission delay and the second is BBRPlus.

Table 6: the configuration of $l2$ to test inter protocol fairness

Case	Bandwidth	Propagation delay	Queue length
1	4Mbps	50ms	4Mbps*100ms
2	4Mbps	50ms	4Mbps*150ms
3	4Mbps	50ms	4Mbps*200ms
4	6Mbps	50ms	6Mbps*100ms
5	6Mbps	50ms	6Mbps*150ms
6	6Mbps	50ms	6Mbps*200ms
7	8Mbps	50ms	8Mbps*150ms
8	10Mbps	50ms	10Mbps*150ms
9	12Mbps	50ms	12Mbps*150ms

Table 7: Results of Jain's fairness index and ratio

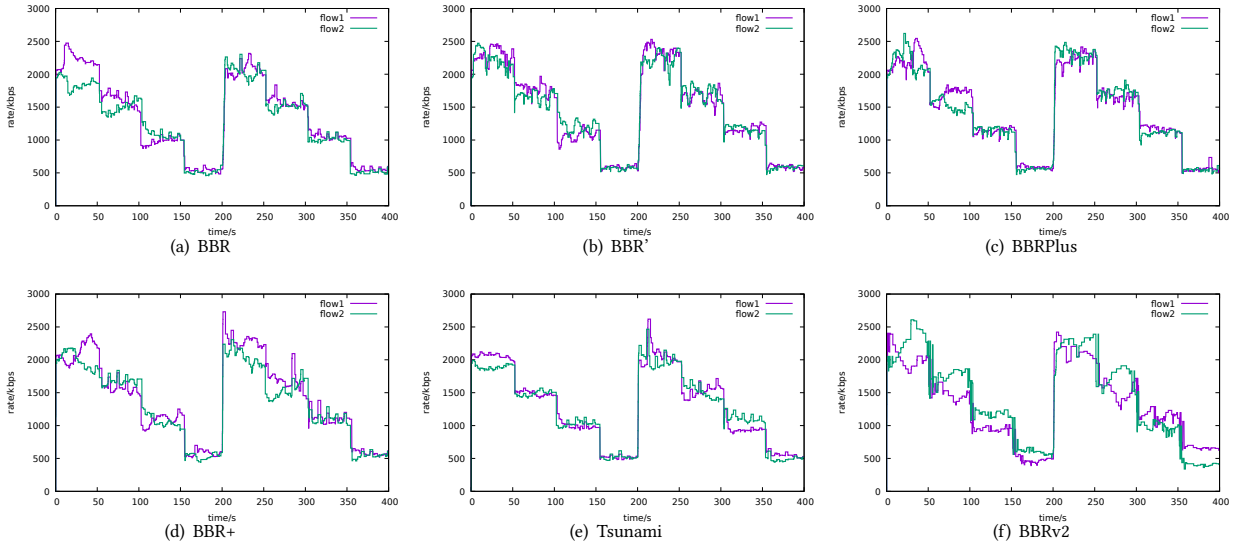
	1	2	3	4	5	6	7	8	9
	(Jain's fairness index, ratio)								
BBR	0.55, 13.61	0.95, 1.7	1.0, 1.09	0.5, 17.08	0.97, 1.52	0.98, 1.42	0.96, 1.63	0.97, 1.53	0.97, 1.59
BBR'	0.93, 1.94	0.97, 1.68	0.99, 1.31	0.99, 1.38	0.98, 1.48	0.97, 1.57	0.96, 1.73	0.95, 1.86	0.95, 1.81
BBRPlus	0.98, 1.48	0.98, 1.4	1.0, 1.13	0.99, 1.22	1.0, 1.17	0.99, 1.28	0.99, 1.28	0.98, 1.51	0.98, 1.35
BBR+	0.65, 5.87	0.81, 3.0	0.95, 1.69	0.54, 7.93	0.92, 1.97	0.97, 1.58	0.86, 2.46	0.96, 1.6	0.92, 2.0
Tsunami	0.4, 26.08	0.91, 2.04	0.74, 4.03	0.38, 34.44	0.91, 2.13	0.74, 3.67	0.86, 2.62	0.86, 2.56	0.95, 1.76
BBRv2	0.96, 1.65	0.98, 1.46	0.99, 1.35	0.99, 1.31	0.97, 1.44	1.0, 1.13	1.0, 1.13	0.97, 1.61	0.95, 1.73

4.5 Inter Protocol Competition

In real networks, a bottleneck link is multiplexed by multiple flows following different congestion control algorithm. And one of the goals of BBR v2 is to make better coexistence with Cubic and Reno flows. The performance of these algorithms are evaluated when the Cubic flows are presence. Total nine experiments are designed and the configuration of $l2$ is shown in Table 6. The simulation process lasts 200 seconds. In each case, the congestion control algorithm used by flow3 and flow4 is Cubic.

The jain's fairness and the throughout ratio between the flow with the maximum rate and the flow with the minimum rate are calculated in Table 7. As example, the rate dynamic over time of the four flows in Case 5 is plotted in Figure 14.

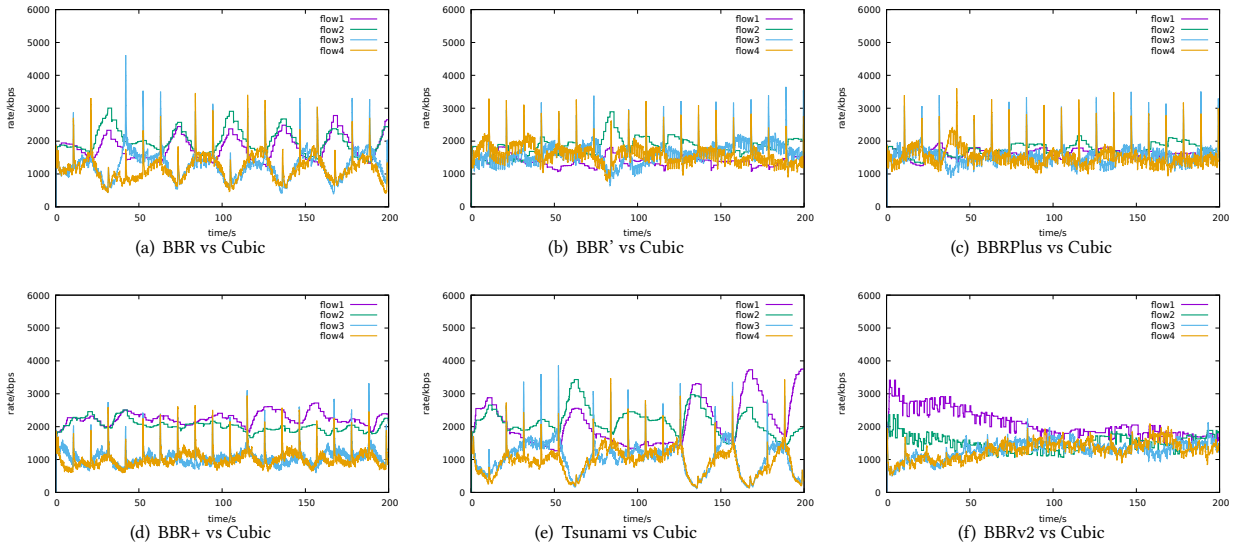
In Case1 and Case4, the buffer length is equal to BDP. There are high packet loss rate when BBR, BBR+ and Tsunami are applied for flow1 and flow2 as congestion control algorithm. The two Cubic flows suffer great loss in throughout due to these packet loss event. Most of bandwidth is occupied by flow1 and flow2. The jain's fairness index is quite low and the throughput ratio is high in this two test cases. As the link buffer increase (Case2, Case3, Case5 and Case6), Cubic flows can achieve higher throughput, but the throughputs of flow1 and flow2 still dominate. This conclusion gets support in Figure 14(a), Figure 14(d), Figure 14(e). The rate of BBR



(a) BBR (b) BBR' (c) BBRPlus

(d) BBR+ (e) Tsunami (f) BBRv2

Figure 13: Rate dynamic of flows in link with variable capacity



(a) BBR vs Cubic (b) BBR' vs Cubic (c) BBRPlus vs Cubic

(d) BBR+ vs Cubic (e) Tsunami vs Cubic (f) BBRv2 vs Cubic

Figure 14: Rate dynamic of flows with different congestion control algorithms

flows (Figure 14(a)), Tsunami flows (Figure 14(e)) shows oscillation when competing bandwidth with Cubic flows.

When BBRPlus flows sharing bottleneck link with Cubic flows, the value of Jain's fairness is quite close to 1, and the rates of BBRPlus flows and Cubic flows are quite close in Case5 as shown in Figure 14(c). In Figure 14(f), the rates of BBRv2 flows and Cubic flows converge close to the fairness line. Both BBRPlus and BBRv2 can be friendly to Cubic.

5 DISCUSSION AND CONCLUSION

We evaluate the performance of BBR, BBR v2 and other algorithms (BBR', BBRPlus, BBR+, Tsunami) modified from BBR on ns3 platform. Bandwidth allocation fairness, packet loss rate, link queue buffer occupation, RTT unfairness, rate responsiveness in variable links and inter protocol fairness are evaluated or measured in this article.

Rethinking on the development of congestion control. The congestion control for transmission protocol can be seen as an optimization problem, different algorithms has been proposed to find the optimal point. A lot of works have been published based on

the original AIMD law and these control parameters are tuned in different network situation to gain better performance.

REFERENCES

[1] V. Jacobson, Congestion avoidance and control, in: ACM SIGCOMM computer communication review, Vol. 18, ACM, 1988, pp. 314–329.

[2] J. Lin, L. Cui, Y. Zhang, F. P. Tso, Q. Guan, Extensive evaluation on the performance and behaviour of tcp congestion control protocols under varied network scenarios, Computer Networks 163 (2019) 106872. doi:https://doi.org/10.1016/j.comnet.2019.106872.
URL http://www.sciencedirect.com/science/article/pii/S1389128618311265

[3] B. Turkovic, F. A. Kuipers, S. Uhlig, Fifty shades of congestion control: A performance and interactions evaluation, arXiv preprint arXiv:1903.03852.

[4] J. Gettys, K. Nichols, Bufferbloat: Dark buffers in the internet, Queue 9 (11) (2011) 1–15.

[5] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson, Bbr: Congestion-based congestion control, Queue 14 (5) (2016) 50:20–50:53. doi:10.1145/3012426.3022184.

[6] M. Hock, R. Bless, M. Zitterbart, Experimental evaluation of bbr congestion control, in: 2017 IEEE 25th International Conference on Network Protocols (ICNP), 2017, pp. 1–10. doi:10.1109/ICNP.2017.8117540.

[7] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer, G. Carle, Towards a deeper understanding of tcp bbr congestion control, in: 2018 IFIP Networking Conference (IFIP Networking) and Workshops, 2018, pp. 1–9. doi:10.23919/IFIPNetworking.2018.8696830.

[8] V. Jain, V. Mittal, M. P. Tahiliani, Design and implementation of tcp bbr in ns-3, in: Proceedings of the 10th Workshop on Ns-3, WNS3 '18, ACM, New York, NY, USA, 2018, pp. 16–22. doi:10.1145/3199902.3199911.

[9] S. Ma, J. Jiang, W. Wang, B. Li, Fairness of congestion-based congestion control: Experimental evaluation and analysis, arXiv preprint arXiv:1706.09115.

[10] J. Wang, Y. Zheng, Y. Ni, C. Xu, F. Qian, W. Li, W. Jiang, Y. Cheng, Z. Cheng, Y. Li, X. Xie, Y. Sun, Z. Wang, An active-passive measurement study of tcp performance over lte on high-speed rails, in: The 25th Annual International Conference on Mobile Computing and Networking, MobiCom '19, ACM, New York, NY, USA, 2019, pp. 18:1–18:16. doi:10.1145/3300061.3300123.
URL http://doi.acm.org/10.1145/3300061.3300123

[11] S. Ha, I. Rhee, L. Xu, Cubic: a new tcp-friendly high-speed tcp variant, ACM SIGOPS operating systems review 42 (5) (2008) 64–74.

[12] V. Misra, W.-B. Gong, D. Towsley, Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red, SIGCOMM Comput. Commun. Rev. 30 (4) (2000) 151–160. doi:10.1145/347057.347421.

[13] D.-M. Chiu, R. Jain, Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, Computer Networks and ISDN Systems 17 (1) (1989) 1 – 14. doi:https://doi.org/10.1016/0169-7552(89)90019-6.

[14] F. P. Kelly, A. K. Maulloo, D. K. H. Tan, Rate control for communication networks: shadow prices, proportional fairness and stability, Journal of the Operational Research Society 49 (3) (1998) 237–252. doi:10.1057/palgrave.jors.2600523.

[15] S. H. Low, D. E. Lapsley, Optimization flow control. i. basic algorithm and convergence, IEEE/ACM Transactions on Networking 7 (6) (1999) 861–874. doi:10.1109/90.811451.

[16] T. Kelly, Scalable tcp: Improving performance in highspeed wide area networks, SIGCOMM Comput. Commun. Rev. 33 (2) (2003) 83–91. doi:10.1145/956981.956989.

[17] D. Leith, R. Shorten, H-tcp: Tcp for high-speed and long-distance networks, in: Proceedings of PFLDnet, Vol. 2004, 2004.

[18] Lisong Xu, K. Harfoush, Injong Rhee, Binary increase congestion control (bic) for fast long-distance networks, in: IEEE INFOCOM 2004, Vol. 4, 2004, pp. 2514–2524 vol.4. doi:10.1109/INFCOM.2004.1354672.

[19] L. S. Brakmo, L. L. Peterson, Tcp vegas: End to end congestion avoidance on a global internet, IEEE Journal on selected Areas in communications 13 (8) (1995) 1465–1480.

[20] D. X. Wei, C. Jin, S. H. Low, S. Hegde, Fast tcp: motivation, architecture, algorithms, performance, IEEE/ACM transactions on Networking 14 (6) (2006) 1246–1259.

[21] A. Kuzmanovic, E. W. Knightly, Tcp-lp: a distributed algorithm for low priority data transfer, in: IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), Vol. 3, 2003, pp. 1691–1701 vol.3. doi:10.1109/INFCOM.2003.1209192.

[22] D. Rossi, C. Testa, S. Valenti, L. Muscariello, Ledbat: The new bittorrent congestion control protocol, in: 2010 Proceedings of 19th International Conference on Computer Communications and Networks, 2010, pp. 1–6. doi:10.1109/ICCCN.2010.5560080.

[23] Cheng Peng Fu, S. C. Liew, Tcp veno: Tcp enhancement for transmission over wireless access networks, IEEE Journal on Selected Areas in Communications 21 (2) (2003) 216–228. doi:10.1109/JSAC.2002.807336.

[24] S. Liu, T. Başar, R. Srikant, Tcp-illinois: A loss- and delay-based congestion control algorithm for high-speed networks, Performance Evaluation 65 (6) (2008) 417 – 440, innovative Performance Evaluation Methodologies and Tools: Selected Papers from ValueTools 2006. doi:https://doi.org/10.1016/j.peva.2007.12.007.

[25] K. Tan, J. Song, Q. Zhang, M. Sridharan, A compound tcp approach for high-speed and long distance networks, in: Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, 2006, pp. 1–12. doi:10.1109/INFOCOM.2006.188.

[26] A. H. David, G. Armitage, Improved coexistence and loss tolerance for delay based tcp congestion control, in: IEEE Local Computer Network Conference, 2010, pp. 24–31. doi:10.1109/LCN.2010.5735714.

[27] D. A. Hayes, G. Armitage, Revisiting tcp congestion control using delay gradients, in: J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, C. Scoglio (Eds.), NETWORKING 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 328–341.

[28] K. Winstein, A. Sivaraman, H. Balakrishnan, Stochastic forecasts achieve high throughput and low delay over cellular networks, in: Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), 2013, pp. 459–471.

[29] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, M. Schapira, Pcc: Re-architecting congestion control for consistent high performance, in: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), 2015, pp. 395–408.

[30] V. Arun, H. Balakrishnan, Copa: Practical delay-based congestion control for the internet, in: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), 2018, pp. 329–342.

[31] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, C. Görg, Adaptive congestion control for unpredictable cellular networks, SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 509–522. doi:10.1145/2829988.2787498.

[32] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, K. Lee, Exll: An extremely low-latency congestion control for mobile cellular networks, in: Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '18, ACM, New York, NY, USA, 2018, pp. 307–319. doi:10.1145/3281411.3281430.

[33] S. Abbasloo, Y. Xu, H. J. Chao, C2tcp: A flexible cellular tcp to meet stringent delay requirements, IEEE Journal on Selected Areas in Communications 37 (4) (2019) 918–932. doi:10.1109/JSAC.2019.2898758.

[34] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data center tcp (dctcp), SIGCOMM Comput. Commun. Rev. 41 (4) (2010) –.
URL http://dl.acm.org/citation.cfm?id=2043164.1851192

[35] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats, Timely: Rtt-based congestion control for the datacenter, SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 537–550. doi:10.1145/2829988.2787510.

[36] K. Winstein, H. Balakrishnan, Tcp ex machina: Computer-generated congestion control, SIGCOMM Comput. Commun. Rev. 43 (4) (2013) 123–134. doi:10.1145/2534169.2486020.

[37] W. Li, F. Zhou, K. R. Chowdhury, W. M. Meleis, Qtcp: Adaptive congestion control with reinforcement learning, IEEE Transactions on Network Science and Engineering (2018) 1–1doi:10.1109/TNSE.2018.2835758.

[38] K. Xiao, S. Mao, J. K. Tugnait, Tcp-drinc: Smart congestion control based on deep reinforcement learning, IEEE Access 7 (2019) 11892–11904. doi:10.1109/ACCESS.2019.2892046.

[39] G. Carlucci, L. De Cicco, S. Holmer, S. Mascolo, G. Carlucci, L. De Cicco, S. Holmer, S. Mascolo, Congestion control for web real-time communication, IEEE/ACM Trans. Netw. 25 (5) (2017) 2629–2642. doi:10.1109/TNET.2017.2703615.

[40] X. Zhu, R. Pan, M. Ramalho, S. de la Cruz, C. Ganzhorn, P. Jones, S. D'Aronco, Nada: A unified congestion control scheme for real-time media, Internet-draft, Internet Engineering Task Force, work in Progress (2018).
URL https://tools.ietf.org/html/draft-ietf-rmcat-nada-07

[41] I. Johansson, Z. Sarker, Self-clocked rate adaptation for multimedia, RFC 8298, RFC Editor (Dec 2017).
URL https://www.rfc-editor.org/rfc/rfc8298.txt

[42] L. Kleinrock, Power and deterministic rules of thumb for probabilistic problems in computer communications, in: Proceedings of the International Conference on Communications, Vol. 43, 1979, pp. 1–10.

[43] J. Jaffe, Flow control power is nondecentralizable, IEEE Transactions on Communications 29 (9) (1981) 1301–1306. doi:10.1109/TCOM.1981.1095152.

[44] N. Cardwell, Y. Cheng, S. H. Yeganeh, I. Swett, V. Jacobson, Bbr v2 a model-based congestion control, 2019.
URL https://datatracker.ietf.org/meeting/104/materials/slides-104-iccrg-an-update-on-bbr-00

[45] R. K. Jain, D.-M. W. Chiu, W. R. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA.

[46] E. Atxutegi, F. Liberal, H. K. Haile, K. Grinnemo, A. Brunstrom, A. Arvidsson, On the use of tcp bbr in cellular networks, IEEE Communications Magazine 56 (3) (2018) 172–179. doi:10.1109/MCOM.2018.1700725.