TCP BBR for Ultra-Low Latency Networking: Challenges, Analysis, and Solutions

Rajeev Kumar[†], Athanasios Koutsaftis[†], Fraida Fund[†], Gaurang Naik[‡], Pei Liu[†], Yong Liu[†], and Shivendra Panwar[†]
[†]Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, USA
[‡]Bradley Department of Electrical and Computer Engineering, Virginia Tech, Arlington, VA, USA

***Abstract*—With the new emerging throughput-intensive ultra-low latency applications, there is a need for a transport layer protocol that can achieve high throughput with low latency. One promising candidate is TCP BBR, a protocol developed by Google, with the aim of achieving high throughput and low latency by operating around the Bandwidth Delay Product (BDP) of the bottleneck link. Google reported significant throughput gains and much lower latency relative to TCP Cubic following the deployment of BBR in their high-speed wide area *wired* network. As most of these emerging applications will be supported by Millimeter Wave (mmWave) wireless networks, BBR should achieve both high throughput and ultra-low latency in these settings. However, in our preliminary experiments with BBR over a mmWave *wireless* link operating at 60 GHz, we observed a severe degradation in throughput that we were able to attribute to high delay variation on the link. In this paper, we show that "throughput collapse" occurs when BBR's estimate of minimum RTT is less than half of the average RTT of the uncongested link (as when delay jitter is large). We demonstrate this phenomenon and explain the underlying reasons for it using a series of controlled experiments on the CloudLab testbed. We also present a mathematical analysis of BBR, which matches our experimental results closely. Based on our analysis, we propose and experimentally evaluate potential solutions that can overcome the throughput collapse without adding significant latency.**

***Index Terms*—TCP, BBR, Congestion Control, 5G, Millimeter Wave, RTT**

I. Introduction

Many throughput-intensive ultra-low latency applications are emerging that require high throughput in the range of 100 Mbps to a few Gbps [1] and latency in the range of 1 to 10 ms [2], including augmented reality/virtual reality (AR/VR), tactile Internet, autonomous driving, and eHealth. To meet the stringent latency and capacity requirements of these applications, Internet Service Providers are pushing application servers close to the end-users [3], and are considering Millimeter Wave (mmWave) links in future cellular networks for the large available bandwidth that they offer [4]. However, mmWave links are highly intermittent, and experience frequent handovers due to blockages [5], [6], which can result in large, rapid variations in the Round Trip Time (RTT) and severe performance degradation [7]. To satisfy the Quality of Service requirements of throughput-intensive ultra-low latency applications, transport layer protocols need to consider the unique RTT characteristics of these links [8].

The default TCP protocol adopted by current Linux-based systems is TCP Cubic, a loss-based congestion control algorithm which aims to achieve high throughput [9]. Cubic and any other loss-based congestion control algorithms aim to completely fill the bottleneck buffer, which may lead to a high queueing delay [10] that is detrimental for throughput-intensive ultra-low latency applications. It is well known from the classic paper of Kleinrock [11] that a TCP congestion control algorithm can achieve full throughput with the lowest possible latency if the maximum number of packets allowed in flight is equal to the Bandwidth Delay Product (BDP), which is the product of the bottleneck link capacity and the link's round trip delay excluding queueing delay.

In 2016, Google proposed a new congestion control algorithm, Bottleneck Bandwidth and Round-trip propagation time (BBR), that aims to achieve high throughput with the minimum possible latency by operating around the BDP. BBR estimates the bottleneck bandwidth and the round-trip propagation time of the link in different phases [12], then sets the congestion window (CWND) to a constant multiple of the estimated BDP. BBR also uses pacing to maintain the inter-departure time of consecutive packets based upon the estimated bottleneck bandwidth to avoid queueing delay. A throughput gain of 2 to 133 times and simultaneous reduction in latency relative to TCP Cubic was reported by Google following the implementation of BBR in Google's data centers and high-speed wide area networks (such as B4) [12].

While BBR has been successfully deployed in large-scale wired networks, wireless links introduce new challenges for BBR. Underlying BBR's approach to BDP estimation is the assumption that the minimum RTT observed on the link is a good estimate of the typical RTT when there is no congestion, and that congestion is the main source of added delay above this minimum RTT. On wireless links, however, large RTT variations often occur for reasons other than congestion, including scheduling and error correction at the medium access control layer, buffering within the cellular network, and handover procedures [13], [14]. Therefore, on wireless links the minimum RTT is *not* necessarily a good estimate of the typical RTT when there is no congestion.

Problems with BBR on wireless links were confirmed by Beshay *et al.* [15], who compared several congestion control algorithms on emulated LTE traces and showed that BBR does not perform well in scenarios where variations in the link RTT is high, i.e., BBR fails to simultaneously achieve high throughput and low latency. We observed this problem ourselves when using BBR on a prototype 5G mmWave

ISBN 978-3-903176-16-4 © 2019 IFIP

wireless backhaul link operating at 60 GHz [16], and we identified high delay variation (relative to the mean RTT) as the link characteristic causing the problem. This is a concern because it is expected that many 5G mmWave links and other future wireless and wired links will operate in a low-delay but high-delay variation regime [17]. Recent updates to BBR address some of the issues observed on wireless links, but there are still scenarios where BBR has severe throughput degradation due to RTT variation.

In this paper, we present an analytical and experimental evaluation of BBR in an ultra-low latency network which may experience high delay variation. Our experimental results show that the network performance may degrade when such delay variations are present. Moreover, we justify the observed degradation over an RTT varying link using an analytical model. The key contributions of this paper are as follows:

- Motivated from our preliminary experiments over a mmWave link operating at 60 GHz, we conduct controlled experiments with delay variation on the CloudLab testbed. We show that throughput collapse occurs in BBR when the minimum RTT estimate falls below half of the average uncongested RTT. This causes CWND to drop below the actual uncongested BDP of the link, leading to CWND exhaustion events that cause BBR's bottleneck bandwidth estimate to collapse.
- To explain the throughput and bandwidth estimate collapse, we present a mathematical model that closely models the minimum RTT and the bandwidth estimate for different jitter values, and predicts that both throughput and bandwidth estimates will collapse when the minimum RTT falls below half of the average uncongested RTT.
- Based on our analysis, we argue that correcting the bottleneck link bandwidth or minimum RTT estimates should alleviate CWND exhaustion events, and thereby, throughput and bandwidth estimate collapse. Our preliminary implementation shows that with these corrections, BBR can mitigate throughput and bandwidth estimate collapse without adding significant latency.

The rest of the paper is organized as follows. We substantiate the effect of RTT variation on BBR performance in Section II. A mathematical analysis that explains the throughput and bandwidth estimate collapse in BBR is presented in Section III. Based on our analysis, we present and evaluate possible solutions in Section IV. Finally, Section V concludes with directions for future work.

II. Experimental evaluation of BBR with RTT variation

In this section, we demonstrate the effect of RTT variation on BBR performance using a series of controlled experiments conducted on the CloudLab [18] testbed. We also use these experiments to explain and validate the underlying reason for the performance degradation.

First, we describe the experiment scenario. The experimental topology includes a TCP source, TCP sink, a link emulator, and a router as shown in Fig. 1. All nodes run Ubuntu Linux

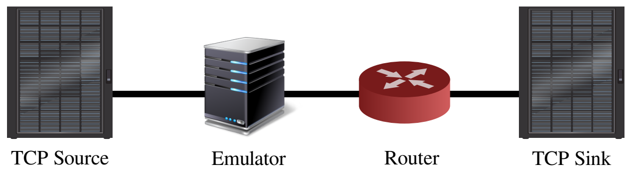
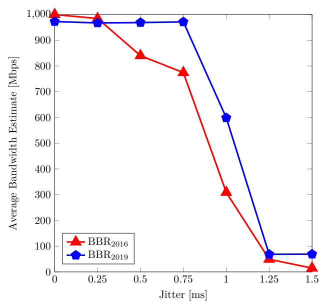


Fig. 1: Experiment setup: the emulator adds delay and delay variation. The link between the router and the TCP sink is the bottleneck link with capacity 1 Gbps.

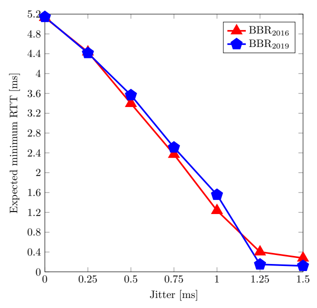
18.04. The bottleneck link is between the router and the TCP sink, and has a capacity of 1 Gbps in both directions; other links in the topology have a capacity several times greater than this. The bottleneck link buffer capacity is 1000 packets, a value that is sufficiently large to avoid packet drops for our experiment settings. The propagation delay between the TCP source and TCP sink is approximately 75 μs. Using the `netem` tool, we add a normally distributed delay with a mean of 5 ms and standard deviation (jitter) in the range of 0 ms to 1.5 ms with a step of 0.25 ms at the emulator. At the TCP source, we evaluate two versions of BBR by loading different Linux kernels: BBR_{2016} is from the Linux kernel version 4.12, which has an implementation of BBR that is essentially identical to the first version introduced in 2016, and BBR_{2019} is from the Linux kernel version 5.1-rc3 and was most recently updated in January 2019. In each experiment, we generate a single 90-second bulk TCP flow from the TCP source to the TCP sink, and record the flow throughput as well as key parameters of BBR such as the bottleneck link bandwidth estimate, minimum RTT estimate, and CWND.

Fig. 2 shows the mean values of the bottleneck link bandwidth estimate, minimum RTT estimate, CWND, and TCP throughput with different jitter values (0 ms to 1 ms) for BBR_{2016} and BBR_{2019}. We first consider BBR_{2016}. From Fig. 2(d), we observe that the throughput of the TCP flow is close to the link capacity (1 Gbps) when the delay jitter is small, but as the jitter increases, it "collapses" to a very small value. Fig. 2(a), Fig. 2(b), and Fig. 2(c) offer several clues regarding the reason for the throughput collapse. We note that the throughput collapse occurs along with a similar collapse in the bottleneck bandwidth estimate (Fig. 2(a)). Furthermore, as per Fig. 2(c), the collapse occurs when the CWND drops below the link BDP, which in this scenario is approximately 450 MSS (maximum segment sizes). Finally, from Fig. 2(b) we observe that the collapse occurs when the jitter is large enough so that the minimum RTT estimate is less than half of the value it is supposed to estimate (the mean RTT of the link when there is no congestion). In summary, when the minimum RTT estimate is less than half of what it should be, the CWND (which in BBR_{2016} is supposed to be double the estimated BDP) drops below the link BDP, and the bottleneck bandwidth estimate and flow throughput collapse.

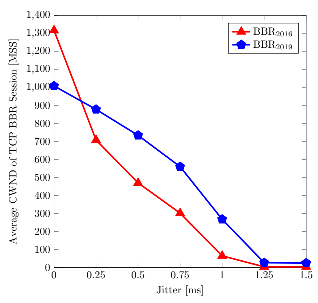
Ordinarily, BBR's sending rate is controlled mostly by pacing, which in turn is set by the bottleneck bandwidth estimate. However, when the CWND is less than the true BDP of the link, BBR is not able to send at the pacing rate due to CWND exhaustion (see Fig. 3). This will eventually



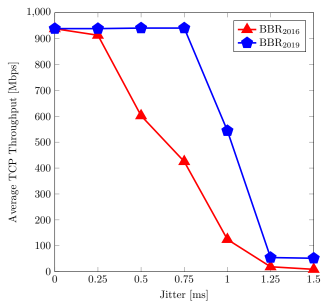
(a) Bandwidth estimate



(b) Minimum RTT



(c) CWND



(d) TCP throughput

Fig. 2: The bottleneck link bandwidth estimate, minimum RTT, CWND, and throughput in BBR$_{2016}$ and BBR$_{2019}$: in an RTT varying link, BBR$_{2019}$ improves the bandwidth estimate as it alleviates CWND exhaustion events by maintaining higher CWND, which reduces throughput loss. In BBR$_{2019}$, a higher CWND is maintained irrespective of lower BDP.

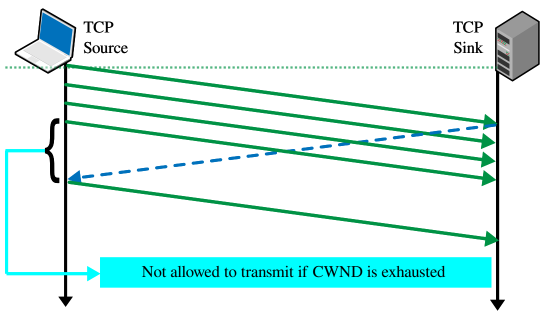


Fig. 3: An example of CWND exhaustion: If ACKs are delayed at the TCP receiver and CWND gets exhausted as it is now below the actual uncongested link BDP, the TCP sender is not allowed to send packets. This results in the incorrect estimation of the bottleneck link bandwidth.

lead to a lower estimate of the bottleneck link bandwidth, and in response to that, the sending rate and CWND will be reduced accordingly. This will in turn increase the likelihood of subsequent instances of CWND exhaustion, which will result in a further drop in the bottleneck link bandwidth estimate and the sending rate, which explains the sharp throughput and bandwidth estimate collapse observed in Fig. 2(a) and Fig. 2(d) in BBR$_{2016}$.

The explanation of TCP BBR behavior around the cliff where throughput starts dropping (see Fig. 5(c) for jitter values between 0.5 ms to 1.25 ms) will be provided in Section III.

BBR$_{2019}$ tries to decrease the impact of delay variation by maintaining a higher CWND as compared to BBR$_{2016}$. By conducting experiments with the same settings as previously used, we found that the CWND value in BBR$_{2019}$ is slightly higher (200 MSS for most of the jitter values) than BBR$_{2016}$ (see Fig. 2(c)). However, at high jitter values, CWND in both BBR$_{2016}$ and BBR$_{2019}$ collapses. Note that for zero jitter in BBR$_{2016}$, BBR$_{2016}$ maintains a significantly higher CWND value and never exits the startup phase, i.e. a CWND and pacing gain of $2/\ln 2$ are always maintained. We suspect this happens as a result of a bug in the BBR$_{2016}$ source code, but we do not claim this with full certainty.

The higher value of CWND in BBR$_{2019}$ alleviates the observed issue in BBR$_{2016}$ by reducing the likelihood of CWND exhaustion events, which results in the higher bottleneck link bandwidth estimate and throughput observed in Fig. 2(a) and Fig. 2(d). However, even with the changes implemented in BBR$_{2019}$, BBR loses around 95% of its throughput in the high delay variation (jitter) regime (see Fig. 2(d)). We should mention that the increase in CWND in BBR$_{2019}$ was not aimed at obtaining correct estimates of the bottleneck link bandwidth and minimum RTT (see Fig. 2(a) and Fig. 2(b)), but due to the many changes that have been implemented in BBR since 2016, to address ACK aggregation, etc., which we will not document in this work.

In the next section, to explain the throughput and the bottleneck bandwidth estimate collapse in BBR, we develop a simple mathematical model that closely matches the observations from experiments conducted over our testbed on CloudLab. In general, our mathematical model predicts when throughput and bottleneck bandwidth estimate collapse will occur, given the average RTT of the TCP session and the jitter distribution. Our mathematical model also estimates the expected value of minimum RTT in a delay varying scenario. Note that our mathematical model is developed for BBR$_{2016}$ but can be extended for the latest BBR version by considering all the changes made.

III. Mathematical Model

In this Section, we present a mathematical analysis to explain the throughput loss observed with link RTT variation. BBR is designed to operate at the BDP in order to achieve the maximum throughput with the minimum possible latency. In order to correctly estimate BDP, BBR uses four different phases to estimate the bottleneck link bandwidth and the minimum RTT. In the first phase, called the *Bandwidth Estimation Phase*, BBR uses slow start to quickly estimate the bottleneck link bandwidth. In the second phase, called the *Drain Phase*, BBR quickly empties the queue that was built up during bandwidth estimation phase to find the minimum RTT of the TCP connection. Upon the completion of these two phases, BBR obtains the initial BDP estimate of the bottleneck link and enters the steady state. After entering the

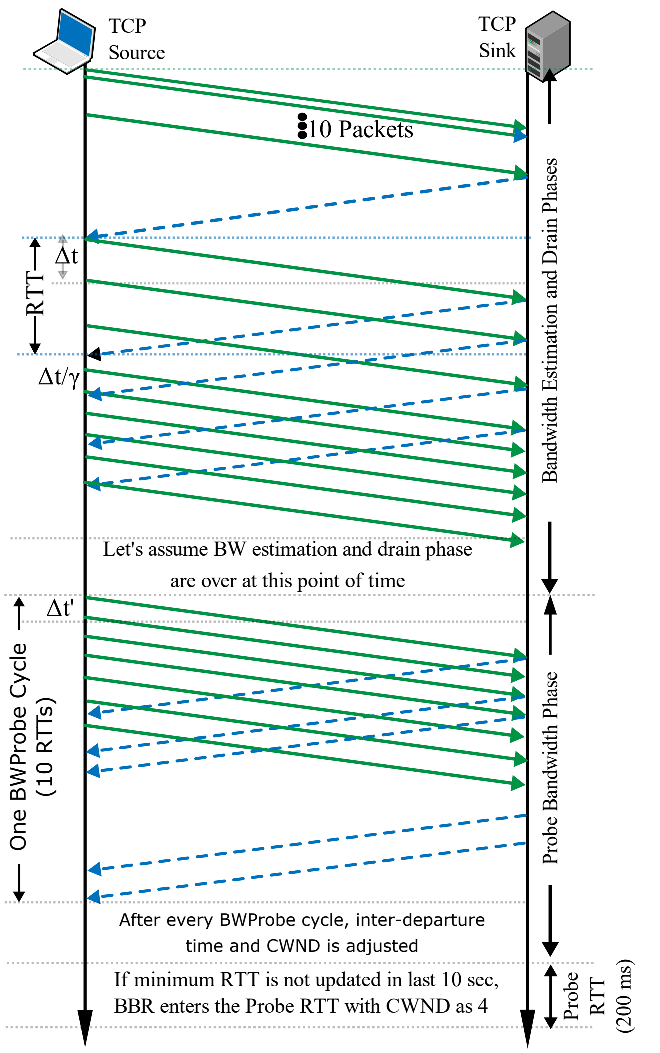

TCP Source
TCP Sink
10 Packets
Bandwidth Estimation and Drain Phases
RTT
Δt
Δt/γ
Let's assume BW estimation and drain phase are over at this point of time
Δt'
One BWProbe Cycle (10 RTTs)
Probe Bandwidth Phase
After every BWProbe cycle, inter-departure time and CWND is adjusted
If minimum RTT is not updated in last 10 sec, BBR enters the Probe RTT with CWND as 4
Probe RTT (200 ms)

Fig. 4: Four Phases of TCP BBR: first two phases are used to estimate the BDP of the bottleneck link and the later two phases continuously update the available capacity at the bottleneck link and its minimum RTT to keep adjusting the sending rate and the CWND by calculating BDP. The inter-departure time of the packets and CWND can be adjusted before the completion of the bandwidth probe cycle if a higher delivery rate sample is observed during this period. In this paper with a single flow, we assumed that such events may be rare and assume that the inter-departure time of packets and CWND are adjusted only after a bandwidth probe cycle.

steady state, BBR constantly obtains new bandwidth estimates in the *Bandwidth Probe Phase* (where it spends approximately 98% of the time) and new minimum RTT values in the *Probe RTT Phase* (with a duration of 200 ms), if the minimum RTT was not updated in the last 10 seconds. A simple diagram to illustrate all of the four phases of BBR is presented in Fig. 4.

A. Bandwidth Estimation Phase

The objective of the first phase (Bandwidth Estimation phase) of BBR is to estimate the bandwidth of the bottleneck link. At the beginning of this phase, the BBR transmitter sends 10 packets and waits for the acknowledgement. Once the sender receives an ACK, it computes an *initial* estimate of the link bandwidth, say BW_{ini} as the ratio of the number of bytes sent during the RTT. The initial packet inter-departure time is then computed as $\Delta t_{\text{ini}} = 1/BW_{\text{ini}}$ (packets/sec)[1]. After obtaining the initial bandwidth estimate, BBR uses binary search to correctly estimate the bandwidth of the bottleneck link. BBR does this by constantly reducing the inter-departure time of packets by a factor of $\gamma = 2$, after every RTT duration. This is known as the *transmission gain* or *gain in the sending rate*. At the same time, CWND is increased by the same factor, i.e. γ, such that the packet transmission rate is not limited by the CWND.

Every time BBR receives the acknowledgment (ACK) of a packet P, it obtains the delivery rate as:

$$d_P = \frac{\mathcal{B}[t_P] - \mathcal{B}[t_C]}{t_P - t_C}, \tag{1}$$

where $\mathcal{B}[t_C]$ is the number of bytes acknowledged just before the transmission of packet P, $\mathcal{B}[t_P]$ is the number of bytes acknowledged until the reception of the ACK corresponding to packet P, t_P is the packet P ACK reception time, and t_C was the time the last ACK is received, just before the transmission of packet P. Until the link capacity is reached, the delivery rate increases approximately by a factor of two after every RTT duration, since the sending rate is doubled. When the observed gain in the delivery rate is less than 1.25 for 3 consecutive RTT durations, the bandwidth estimation phase ends, and the observed delivery rate is set to be the initial estimate of the bottleneck link bandwidth.

As the Bandwidth Estimation Phase is only needed to estimate the bottleneck link capacity BW_{BEP} when the TCP BBR session starts, and it does not have a significant impact on the performance of the rest of the TCP BBR session, we will not focus on this phase.

B. Drain Phase

Once the initial estimate of the bottleneck link bandwidth (BW_{BEP}) is obtained, BBR enters the Drain Phase. In this phase, after every RTT duration, the packet sending rate is reduced by a factor of $\gamma = 2$, to deplete the queue that has been built up during the Bandwidth Estimation Phase. When there are no packets left in the queue, and a reduction in the packet transmission rate does not result in further reduction of the RTT, the RTT at this stage is considered to be the estimate of minimum RTT, $\widehat{RTT}$[2].

Using the minimum RTT estimate, the initial BDP is estimated as:

$$BDP_{\text{ini}} = BW_{\text{BEP}} \times \widehat{RTT} \tag{2}$$

Again, as the duration of Drain Phase is quite small and may not have much significance over the duration of the TCP session, we will not focus on this phase either.

[1]Whenever the inter-departure time of packets is computed, we convert the bandwidth estimate into packets per second.

[2]We refer to the true minimum RTT as RTT_{min}, while $\widehat{RTT}$ is an estimate of RTT_{min}.

C. Bandwidth Probe Phase

Recall from our previous discussion in the Bandwidth Estimation Phase that at the reception of each ACK, BBR computes the delivery rate using (1). Thereafter, it estimates the bottleneck link bandwidth as the maximum of delivery rate samples over ten RTT periods. BBR continuously updates the bottleneck link bandwidth estimate using this sliding window of 10. At the same time, as BBR also keeps updating the minimum RTT at the reception of ACKs, it uses the bandwidth and the minimum RTT estimate to update the BDP and the CWND. Note that in this phase, BBR maintains approximately constant sending rate, i.e., a pacing gain of one is maintained. However, BBR occasionally increases its pacing gain, γ, to 1.25 to probe for more bandwidth and then decreases it to 0.75 to achieve a lower RTT by emptying any queue built up during the increased sending rate cycle. Note that when the bottleneck link is shared by multiple TCP flows, the start of a new TCP session or the end of an existing TCP session may impact the available bandwidth and the minimum RTT. BBR maintains these pacing gains to probe for such events. We neglect these occasional pacing gain cycles as we have considered a single TCP flow in this work. Furthermore, for simplicity, we assume the best case scenario where BBR correctly estimates the bottleneck bandwidth in the bandwidth estimation phase.

Now, to develop a sound understanding, let us start with the simpler case of a stable link with no RTT variation when it is uncongested. Assume that the bottleneck bandwidth estimate in the first phase is BW_{BEP}, then the inter-departure time of packets is calculated as $\Delta t = 1/BW_{\text{BEP}}$, where BW_{BEP} is converted from bytes/sec to packets/sec. Consider a packet P, and assume that the TCP source sends P at time t_0 and receives its acknowledgment at time t_1. Then the number of bytes getting ACKed during the flight of packet P can be computed as $(t_1 - t_0)/\Delta t = RTT/\Delta t$[3]. Note that the time difference $t_1 - t_0$ is the RTT of the packet P. Furthermore, in an uncongested stable link, the RTTs of all of the packets will be same as $RTT = t_1 - t_0$ and this value will be estimated as the minimum RTT by TCP BBR. We will refer to this value as the uncongested average RTT and call it RTT_{UCAV}. The delivery rate estimate during the flight of packet P is computed as:

$$d_p = \frac{1}{RTT}\frac{RTT}{\Delta t} = \frac{1}{\Delta t}. \tag{3}$$

Thus, in a stable link case, during the flight of each packet, the delivery rate will be always computed as the initial bandwidth and the minimum RTT as the uncongested average RTT. This will result in bandwidth estimation during the j^{th} bandwidth probe cycle as $BW^j = 1/\Delta t,\ \forall j$.

Next, consider a wireless link with uncongested average RTT RTT_{UCAV}, where there may be an offset between the RTT of each packet and the average RTT due to RTT variation. Recall that BBR computes the delivery rate by measuring the number of bytes acknowledged over an RTT. Although these windows may vary around RTT_{UCAV} due to the link RTT variation, for simplicity, we assume RTT_{UCAV} as the observation window and compute delivery rate on the expiration of such window. Then the delivery rate estimate during each observation window in the j^{th} bandwidth probe cycle is computed as:

$$d^j = \frac{1}{RTT_{\text{UCAV}}}\min\left(W_j, \frac{RTT_{\text{UCAV}}}{\Delta t_{j-1}}\right), \tag{4}$$

where Δt_{j-1} is the inter-departure time of the packets during the j^{th} bandwidth probe cycle based upon the bandwidth estimate in the $(j-1)^{\text{th}}$ bandwidth probe cycle. Note that the initial inter-departure time in the first bandwidth cycle is computed using the bandwidth estimate in the bandwidth estimation phase, i.e. $\Delta t_0 = 1/BW_{\text{BEP}}$. W_j is the CWND during the j^{th} bandwidth probe cycle based upon the bandwidth estimation in the $(j-1)^{\text{th}}$ bandwidth probe cycle and minimum RTT until the $(j-1)^{\text{th}}$ bandwidth probe cycle. W_0 is computed using the bandwidth estimate in the bandwidth estimation phase and minimum RTT estimation in the drain phase, i.e $W_0 = 2BW_{\text{BEP}}\mathbb{E}\left[\widehat{RTT}\right]$. If W_j is lower than the number of packets that can be transmitted by the TCP source during an RTT, then in the j^{th} bandwidth probe cycle CWND exhaustion may occur. Thus, the bandwidth estimation in the j^{th} bandwidth probe cycle can be computed as:

$$\begin{aligned} BW^j &= \frac{1}{RTT_{\text{UCAV}}}\min\left(W_j, \frac{RTT_{\text{UCAV}}}{\Delta t_{j-1}}\right) \\ &= \frac{1}{RTT_{\text{UCAV}}}\min\left(\frac{2RTT_{\min}^{j-1}}{\Delta t_{j-1}}, \frac{RTT_{\text{UCAV}}}{\Delta t_{j-1}}\right) \\ &= \frac{1}{RTT_{\text{UCAV}}\Delta t_{j-1}}\min\left(2RTT_{\min}^{j-1}, RTT_{\text{UCAV}}\right) \end{aligned} \tag{5}$$

where $RTT_{\min}^{j-1}$ is the minimum of all RTT values observed until the $(j-1)^{\text{th}}$ bandwidth probe cycle.

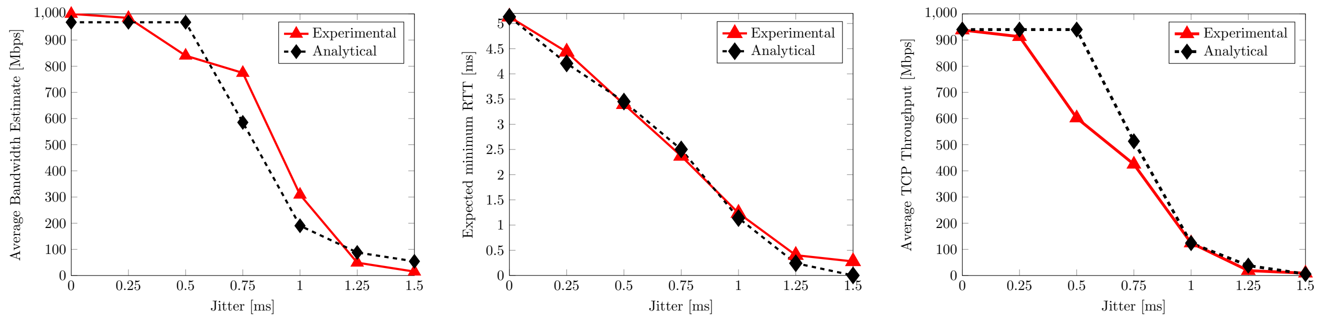
We further simplify (5) to obtain the bottleneck bandwidth in the two boundary cases: the low delay variation regime and the high delay variation regime. In the low delay variation regime, the effect of variation will not be prominent, i.e. $\min\left(2RTT_{\min}^{j-1}, RTT_{\text{UCAV}}\right) = RTT_{\text{UCAV}}$. Then, the bandwidth estimate in the j^{th} bandwidth probe cycle can be simply written as:

$$BW^j = \frac{1}{\Delta t_{j-1}}. \tag{6}$$

From (6), we can observe that in the low delay variation regime, where CWND exhaustion is not likely and the estimated minimum RTT is greater than half of the RTT_{UCAV}, BBR estimates the bottleneck link bandwidth close to the link capacity and will not suffer the bandwidth estimate collapse.

Next, consider a high delay variation regime, where the minimum RTT falls below half of RTT_{UCAV}. Recall from the previous discussion that the CWND in the j^{th} bandwidth probe cycle is a function of the minimum RTT $RTT_{\min}^{j-1}$ and the bandwidth estimate $(1/\Delta t_{j-1})$ after the $(j-1)^{\text{th}}$ bandwidth probe cycle. Thus, a significant drop in the estimate

[3]For a stable link, we can assume that the number of bytes sent is equal to the number of bytes ACKed during the flight of packet P.



(a) Bandwidth estimate (b) Minimum RTT estimate (c) BBR throughput

Fig. 5: BBR$_{2016}$ experimental and analytical results: the bottleneck link bandwidth estimate, minimum RTT estimate, and throughput observed during our experiment and predicted by our mathematical model. Both the experimental system and the mathematical model consider an RTT normally distributed with mean 5 ms and jitter varied in the range of 0 ms to 1.5ms.

RTT_{min}^{j-1} due to high RTT variation will result in significantly smaller CWND (less than average uncongested BDP), which will result in CWND exhaustion. Furthermore, in this regime, $\min\left(2RTT_{\text{min}}^{j-1}, RTT_{\text{UCAV}}\right) = 2RTT_{\text{min}}^{j-1}$. Then, the bandwidth estimate in the j^{th} bandwidth probe cycle can be written as:

$$BW^j = \frac{1}{\Delta t_{j-1}} \frac{2RTT_{\text{min}}^{j-1}}{RTT_{\text{UCAV}}}. \qquad (7)$$

From (7), we can observe that in a high delay variation regime, when the minimum RTT estimate falls below half of the RTT_{UCAV}, the bottleneck link bandwidth estimate will keep reducing with the depletion of the minimum RTT estimate. This happens because a lower estimate of the minimum RTT results in a lower estimate of CWND, which in turn results in CWND exhaustion events. In a high RTT variation regime, it is highly likely that the minimum RTT falls below the RTT_{UCAV} which we will discuss in the next section.

The expressions in (5), (6), and (7) explain the bandwidth estimates obtained in different delay variation regimes.

D. *RTT Probe Phase*

BBR spends around 98% of the time in the Bandwidth Probe phase [12]. At every ACK arrival, BBR updates the minimum RTT value if a new minimum RTT value is obtained. If in an entire duration of 10 seconds BBR does not update the minimum RTT value, it enters the *RTT Probe Phase* which has a duration of 200 ms, where it maintains a low CWND of 4 packets. From the bandwidth probe phase, we observe that when the minimum RTT falls below half of the uncongested average RTT, i.e. RTT_{UCAV}, throughput and bandwidth estimate collapse will occur. Thus, we are interested in obtaining the expected value of the minimum RTT during a TCP BBR session and the RTT variation that will cause throughput and bandwidth estimation collapse.

1) Expected value of minimum RTT: Based upon our previous discussion, the minimum RTT RTT_{min} is either obtained in the RTT probe phase or in the bandwidth probe phase. Let us assume that a and b are the minimum RTT values obtained in the Probe RTT phase and probe bandwidth phase, respectively. Then, minimum RTT can be written as:

$$RTT_{\text{min}} = \min(a, b). \qquad (8)$$

Although RTT values may be correlated, for simplicity, we model RTT samples obtained during *Probe RTT Phase* as independent and identically distributed (i.i.d) random variables. Let us assume that RTT sample in the probe RTT phase is denoted by X. We further assume that the RTT samples in the *Probe Bandwidth Phase* are i.i.d. and denoted as Y. We model the RTT samples in these two phases by different random variables as BBR may have queueing delay in the probe bandwidth phases since it always maintains a CWND twice of BDP.

In (8), $a := \min\{x_1, x_2, \cdots, x_N\}$ is the minimum RTT value obtained from N i.i.d. samples following the distribution A observed in the Probe RTT Phase. Similarly, $b := \min\{y_1, y_2, \cdots, y_M\}$ is the minimum RTT value obtained from M i.i.d. samples following the distribution B observed in the Probe Bandwidth Phase. Then, the expected value of RTT_{min} is computed as:

$$\begin{aligned}\mathbb{E}\left[RTT_{\text{min}}\right] &= \int_0^\infty \mathbb{P}\left(RTT_{\text{min}} > s\right) ds \\ &= \int_0^\infty \mathbb{P}\left[x_i > s\right]\mathbb{P}\left[y_j > s\right] ds, \forall i \in \{1, N\}, \forall j \in \{1, M\} \\ &= \int_0^\infty \mathbb{P}\left[x > s\right]^N \mathbb{P}\left[y > s\right]^M ds \\ &= \int_0^\infty \left(1 - F_X(s)\right)^N \left(1 - F_Y(s)\right)^M ds\end{aligned} \qquad (9)$$

where $F_X(\cdot)$ and $F_Y(\cdot)$ are the cumulative distribution functions (CDF) of X and Y, respectively. In (9), the second step is obtained using the fact that RTT samples are independent.

2) Prediction of throughput and estimated bandwidth collapse: To predict the throughput and the bandwidth estimate collapse with the link RTT variation, we assume following:

- We assume that the bandwidth was correctly estimated in all the previous phases before bandwidth and throughput collapse starts.

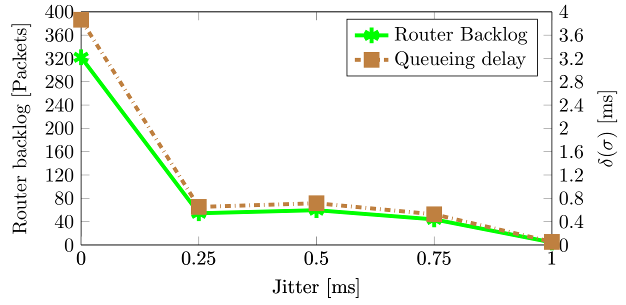


Fig. 6: Router backlog and queueing delay in BBR_{2016}.

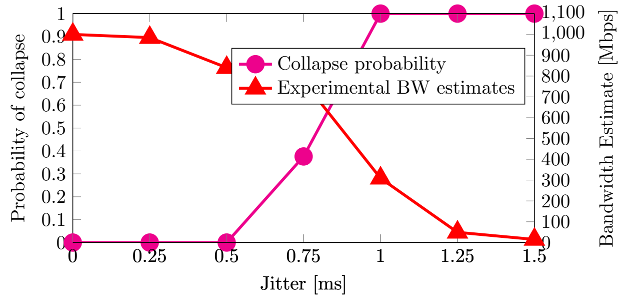


Fig. 7: Probability of bottleneck link bandwidth estimate collapse and experiment values in a varying RTT link for TCP BBR_{2016}: our model shows that with higher jitter values, the probability of throughput and bottleneck estimated bandwidth collapse increases.

- Based upon our observations, we assume that throughput collapse happens only in the Probe Bandwidth Phase.

Let us assume that L is the number of RTT samples obtained during Probe Bandwidth Phase before the minimum RTT is estimated below half of the true uncongested average RTT (RTT_{UCAV}). Then, the minimum RTT value is computed as:

$$c = \min\{b_1, b_2, \cdots, b_L\} \tag{10}$$

Recall that bottleneck bandwidth estimation and throughput collapse happens when the minimum RTT is estimated below half of RTT_{UCAV}. Thus, to evaluate the jitter at which this collapse happens, we need to solve the equation below:

$$\begin{aligned} &\mathbb{P}\left[c < \frac{RTT_{\text{UCAV}}}{2}\right] > 1-\epsilon \\ \Leftrightarrow\ & F_C\left(\frac{RTT_{\text{UCAV}}}{2}\right) > 1-\epsilon \\ \Leftrightarrow\ & 1-\left(1-F_Y\left(\frac{RTT_{\text{UCAV}}}{2}\right)\right)^L > 1-\epsilon \\ \Leftrightarrow\ & F_Y\left(\frac{RTT_{\text{UCAV}}}{2}\right) > 1-\sqrt[L]{\epsilon}, \end{aligned} \tag{11}$$

where ϵ is small number chosen arbitrarily. Note that the third step in (11) is obtained using independence of RTT samples.

E. Experimental and Analytical Result Comparison

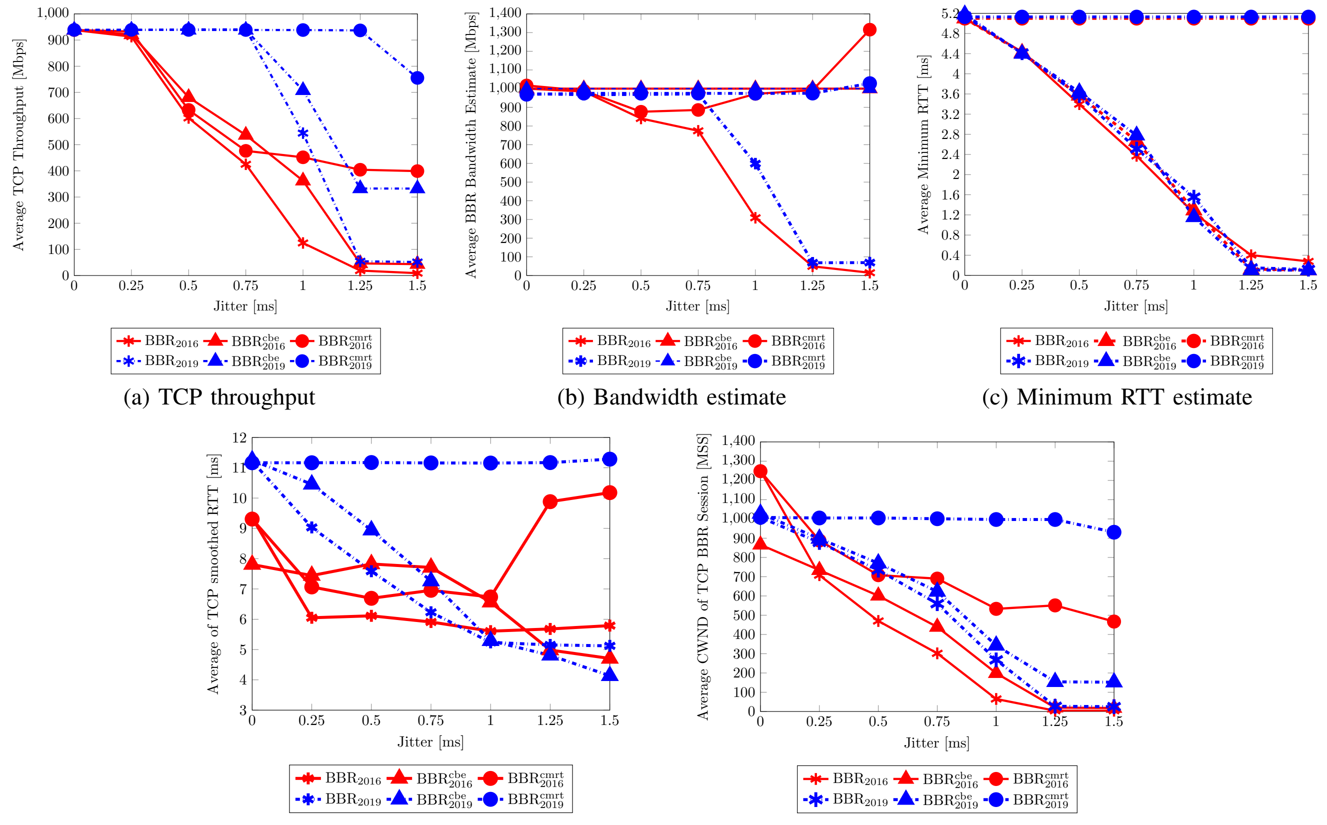
In our analytical evaluations, we assume that the RTT samples in the probe RTT phase are i.i.d. and follow a truncated normal distribution with the lower bound as the propagation and transmission delay η, i.e., $X \sim \mathcal{N}(\mu, \sigma; \eta, \infty)$. As the CWND is maintained at twice the BDP in the probe bandwidth phase, BBR introduces queueing delay at the bottleneck link. Let us assume that the queueing delay $\delta(\sigma)$ at the bottleneck link is a function of jitter only. Note that we obtain $\delta(\sigma)$ from the testbed experiments using the router average backlog (see Fig. 6). Then, we compute $\delta(\sigma)$ as the ratio of router average backlog to the bottleneck link capacity in packets per second. Thus, the RTT samples in the probe bandwidth phase are i.i.d. and follow a truncated normal distribution with the lower bound as the propagation and link transmission delay η, i.e. $Y \sim \mathcal{N}(\mu+\delta(\sigma), \sigma; \eta, \infty)$. In (9), the value of N is computed as the ratio of CWND in probe RTT phase (4 packets) times probe RTT duration (200 ms) to average RTT, i.e. $N = 160$ packets. We consider $M = 10000$ packets based upon the throughput in the high delay variation regime.

As BBR estimates the bottleneck link bandwidth and the minimum RTT using a feedback mechanism, which involves information obtained on previous phases, we simulated the TCP session in Fig. 5(a) to evaluate the average bandwidth estimate. To calculate how the bandwidth estimate is updated throughout the bandwidth probe phase, we use (5). In Fig. 5(b) we use the average queueing delay information from our experiment to compute the minimum RTT estimate in our analytical results. From Fig. 5, we observe that our analytical results closely follow the experimental ones with a small deviation for a few jitter values.

Fig. 7 demonstrates the probability of the throughput and the bottleneck link bandwidth estimate collapse in the different delay variation regimes. In Fig. 7, we selected $L = 10000$ packets based upon the throughput observed in the testbed experiments in the high jitter regime. From Fig. 7, we observe that in the low delay regime the probability of collapse tends to zero, while in the high delay variation regime this probability tends to one. In the cliff, we observe that the probability of collapse lies between zero and one during a single bandwidth probe cycle, i.e. collapse may or may not happen in the bandwidth probe cycles. Thus, the results in Fig. 7 strongly agree with our experimental observations described in Section II.

IV. Possible Solutions

Based upon the expression in (5), we propose two possible solutions to alleviate the throughput and bandwidth estimate collapse. Note from the expression in (5) that the bandwidth estimate in the j^{th} bandwidth probe cycle can be affected by either the incorrect bandwidth estimate in the $(j-1)^{\text{th}}$ bandwidth probe cycle or by the incorrect estimate of the minimum RTT values observed till the $(j-1)^{\text{th}}$ bandwidth probe cycle. A solution would be to obtain the correct estimate of the bottleneck link bandwidth and the minimum RTT of the TCP session. To check the efficacy of such a solution, we implement an ad hoc solution, where we assume that BBR has the information of the actual bottleneck link capacity or the minimum RTT of the TCP session. We implement these two possible solutions in both BBR_{2016} and BBR_{2019}. $\text{BBR}^{\text{cbe}}_{2016}$



(a) TCP throughput (b) Bandwidth estimate (c) Minimum RTT estimate

(d) Smoothed TCP RTT (e) CWND

Fig. 8: Implementation of two possible solutions in the BBR source code: either the bottleneck link capacity or the uncongested average RTT value is provided. Note that BBR_{2016} is the original version as of Linux kernel 4.12 without any modifications, BBR^{cbe}_{2016} is the version with the correct bottleneck link bandwidth provided to BBR_{2016} source code, BBR^{cmrt}_{2016} is the version with the correct minimum RTT provided to BBR_{2016} source code. Similar notation applies to BBR_{2019}, BBR^{cbe}_{2019}, and BBR^{cmrt}_{2019}.

and BBR^{cbe}_{2019} are the BBR versions, where we provided the correct information of the bottleneck link capacity to BBR_{2017} and BBR_{2019}, respectively. Similarly, BBR^{cmrt}_{2016} and BBR^{cmrt}_{2019} are BBR versions where we provided RTT_{UCAV} to BBR_{2016} and BBR_{2019}, respectively.

Recall from Section II that BBR_{2016} obtains incorrect estimates of the bottleneck link bandwidth and minimum RTT, due to CWND exhaustion. In particular, we observed from both our experiments and analytical model that bandwidth estimate collapse is evident when the minimum RTT estimate is less than half of the RTT_{UCAV}. By providing BBR_{2016} the correct bottleneck link bandwidth, BBR^{cbe}_{2016} maintains a constant sending rate (see Fig. 8(b)) and a higher value of CWND as compared to BBR_{2016} (see Fig. 8(e)), which results in alleviating CWND exhaustion events. However, in a high delay variation regime, CWND still collapses as the minimum RTT estimate drops to a significantly low value, which results in CWND exhaustion events. As a result, throughput collapses as well. Note that for most jitter values, BBR^{cbe}_{2016} achieves higher TCP throughput without adding significant delay (around 2 ms) (see Fig. 8(d))).

By modifying BBR_{2016} with the RTT_{UCAV} for this experiment (see Fig. 8(c)), we can observe that BBR^{cmrt}_{2016} avoids any bandwidth estimate collapse and always maintains a higher CWND (see Fig. 8(b) and Fig. 8(e)). Recall from our previous discussion in Section II about BBR_{2016} for a link with no RTT variation that BBR_{2016} never exits startup phase possibly due to a bug in the BBR code. We encounter the same problem in BBR^{cmrt}_{2016} for the zero jitter value, where it does not leave the startup phase. As a high CWND is maintained in BBR^{cmrt}_{2016}, BBR^{cmrt}_{2016} can obtain high delivery rate estimates even with the link RTT variation, which results in a high bandwidth estimate. Furthermore, the CWND in BBR^{cmrt}_{2016} decreases somewhat in the high delay variation regime as BBR maintains a safe margin to avoid unnecessary queueing at the bottleneck link by setting the CWND as the minimum of the target CWND and the sum of previous CWND and packets ACKed during last RTT (see Fig.8(e)). Here, the target CWND is twice the estimated BDP in the last bandwidth probe phase. The CWND in BBR^{cmrt}_{2016} is maintained significantly higher than CWND in BBR_{2016}, which results in lower inter-departure time of the packets. This in turn results in a higher queueing delay (see Fig. 8(d)). As BBR^{cmrt}_{2016} maintains high CWND, CWND exhaustion events are avoided which results in higher throughput performance(see Fig. 8(a)). Note that in both potential solutions BBR^{cbe}_{2016} and BBR^{cmrt}_{2016}, the modified

BBR versions achieve higher throughput compared to the default implementation.

Recall from Section II that BBR_{2019} obtains incorrect estimates of the bottleneck link bandwidth and minimum RTT, due to the same CWND exhaustion problems that BBR_{2016} has. As $\text{BBR}^{\text{cbe}}_{2019}$ maintains a higher CWND (by approximately 120 MSS) (see Fig. 8(e)) and constant sending rate (see Fig. 8b), $\text{BBR}^{\text{cbe}}_{2019}$ alleviates CWND exhaustion events that results in higher throughput (see Fig. 8(a)). A higher CWND also results in slightly higher delay as a result of queueing at the bottleneck link(see Fig. 8(d) and negligibly higher minimum RTT (see Fig 8(b)). Furthermore, by obtaining the correct minimum RTT of the TCP session as RTT_{UCAV}, $\text{BBR}^{\text{cmrt}}_{2019}$ estimates the correct BDP of the bottleneck link and CWND. We can observe from Fig 8(a) and Fig. 8(d) that $\text{BBR}^{\text{cmrt}}_{2019}$ attains a very high throughput (close to the bottleneck link capacity) and an almost constant smoothed RTT for different jitter values.

Based on our findings, we conclude that if BBR_{2019} estimates RTT_{UCAV} as the minimum RTT, it can achieve maximum throughput and minimum delay. However, BBR_{2019} estimates the minimum RTT as the minimum of all of the RTT samples, which results in performance degradation in a RTT varying link as we have argued in this work. In our future work, we are will use both short term and long-term statistics to optimize BBR performance. A possible solution is to correctly estimate RTT_{UCAV} involves using the statistics of RTT samples in the probe RTT phases, instead of *a priori* providing the information to the source code. In this phase, the CWND is set to a minimum of four packets, which depletes the bottleneck link queue and the bottleneck link will be eventually uncongested. By gathering statistics of RTT samples, such as mean, variance, and the 95% percentile value, we plan to build an RTT_{UCAV} estimator.

V. Conclusion

In this work, we presented a mathematical and experimental evaluation of TCP BBR over an RTT varying link. In our preliminary experiments over a mmWave point-to-point link operating at 60 GHz, we observed significant throughput loss with BBR_{2016}. We argued that the performance degradation in BBR is caused by the link RTT variation, which results in incorrect estimates of the bottleneck link bandwidth and minimum RTT. To justify this, we conducted several experiments in a controlled environment over CloudLab. We observed that in an RTT varying link, when the minimum RTT estimates fall below half of the true uncongested average RTT, BBR suffers a throughput collapse due to CWND exhaustion. Our mathematical model provides an estimate of the bottleneck link bandwidth and minimum RTT, while simultaneously predicting the point at which the throughput and bandwidth estimate collapse occurs in a link with delay variance. We argue that any TCP congestion control algorithm that aims to operate around the BDP by estimating the bottleneck link bandwidth and RTT may suffer similar challenges as BBR in an RTT varying link. Based on our mathematical analysis, we argued that obtaining the correct estimates of the bottleneck link bandwidth and minimum RTT will mitigate such events. Our preliminary implementation shows that by obtaining correct estimates, BBR can avoid throughput collapses without adding significant latency. Our future work includes designing a method for obtaining estimates of the bottleneck link bandwidth and minimum RTT in an RTT varying link by exploiting the statistics of RTT samples in the probe RTT phase.

References

[1] A. Seam, A. Poll, R. Wright *et al.*, "Enabling mobile augmented and virtual reality with 5G networks," *tech. rep., AT&T Foundry*, Jan. 2017. [Online]. Available: https://soc.att.com/2XwZbSu

[2] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and low-latency wireless communication: Tail, risk, and scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, Oct 2018.

[3] S. Singh, Y.-C. Chiu, Y.-H. Tsai, and J.-S. Yang, "Mobile edge fog computing in 5G era: architecture and implementation," in *Proc. of IEEE ICS*, Dec. 2016.

[4] T. S. Rappaport, S. Sun, R. Mayzus *et al.*, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE Access*, vol. 1, pp. 335–349, May 2013.

[5] I. K. Jain, R. Kumar, and S. Panwar, "Driven by capacity or blockage? a millimeter wave blockage analysis," in *Proc. of International Teletraffic Congress (ITC 30)*, vol. 01, Sept 2018, pp. 153–159.

[6] I. K. Jain, R. Kumar, and S. S. Panwar, "The impact of mobile blockers on millimeter wave cellular systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 854–868, Apr. 2019.

[7] M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Mobility management for TCP in mmwave networks," in *Proc. of ACM mmNets*, 2017.

[8] K. Inoue *et al.*, "Low-latency and high bandwidth TCP/IP protocol processing through an integrated HW/SW approach," in *Proc. of IEEE INFOCOM*, Apr. 2013.

[9] S. Ha *et al.*, "Cubic: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, 2008.

[10] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," *Queue*, vol. 9, no. 11, p. 40, 2011.

[11] 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. 1–43.

[12] N. Cardwell *et al.*, "BBR: congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016.

[13] J. Korhonen and Y. Wang, "Effect of packet size on loss rate and delay in wireless links," in *IEEE Wireless Communications and Networking Conference, 2005*, vol. 3. IEEE, 2005, pp. 1608–1613.

[14] 3GPP TS 36.331, "LTE; evolved universal terrestrial radio access (E-UTRA); radio resource control (RRC); protocol specification," 2010.

[15] J. D. Beshay *et al.*, "Link-coupled TCP for 5G networks," in *Proc of IEEE/ACM IWQoS*, June 2017.

[16] InterDigital, Inc., "Edgelink solution," Tech. Rep., Jun. 2017. [Online]. Available: https://bit.ly/2JlXMpp

[17] D. Chitimalla, K. Kondepu, L. Valcarenghi, M. Tornatore, and B. Mukherjee, "5G fronthaul-latency and jitter studies of CPRI over Ethernet," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 2, pp. 172–182, 2017.

[18] R. Ricci, E. Eide, and C. Team, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications," *The magazine of USENIX & SAGE*, vol. 39, no. 6, pp. 36–38, 2014.