Congestion Control Optimization for Short Video Services: User-End and Edge Server Collaboration in Practice

Jupeng Zhang*, Yan Liu*, Jack Y. B. Lee†, Shengtong Zhu†
*NetLab, Bytedance Inc., Beijing, China
†Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China
Email: {zhangjupeng, liuyan.rockyanliu}@bytedance.com, {yblee, zs021}@ie.cuhk.edu.hk

***Abstract*—Short video applications such as TikTok, Douyin, and Kwai have experienced significant popularity in recent years. However, the quality of experience (QoE) provided by short video streaming services still falls short of expectations. As a leading provider of short video services with proprietary video players and content delivery network (CDN) capabilities, we are in a unique position to optimize the QoE of these services. In this study, we present our pilot investigation into congestion control performance optimization for short video services by leveraging collaboration between user-end video players and edge servers. Based on a comprehensive measurement study of network characteristics from production networks and incorporating feedback from video players, we developed an optimized congestion control algorithm called BBR-E2E. We deployed BBR-E2E in our production network and conducted a large-scale A/B testing across the country, involving trillions of video sessions over a three-month period in China. Overall, we observed a 1.6% reduction in rebuffering duration and a 6.2% decrease in rebuffering count. At the provincial level[1], the improvements were even more substantial, with up to a 7.8% reduction in rebuffering duration and a 13.7% decrease in rebuffering count.**

***Index Terms*—short video, congestion control algorithm, optimization, QoE**

I. Introduction

In recent years, short video applications such as TikTok, Douyin, and Kawi have experienced an unprecedented surge in global popularity. TikTok alone has reported more than 1 billion active users per month since 2021 [1]. Furthermore, market forecasts indicate that the market size for short videos is expected to double between 2023 and 2030 [2], [3].

Short videos typically have duration ranging from 5 to 15 seconds, with the majority falling within 60 seconds [4], [5]. Short video applications enable users to watch a variety of short videos by swiping through their screens. Users can sequentially watch videos from a playlist generated by recommendation systems if they find them interesting. However, if the videos don't catch their attention, users tend to skip them rapidly, a behavior known as fast swiping. This swiping behavior presents a unique challenge as video segments need to be downloaded promptly to avoid long startup delays and playback rebuffering.

[1]A province in China is similar to a state in the USA.

979-8-3503-5171-2/24/$31.00 ©2024 IEEE

Unlike file downloading services, where the primary objective is to optimize the mean throughput across multiple file download events, short video streaming services prioritize reducing scenarios where the throughput falls below the client's video bitrate. When the throughput is below the video playback bitrate, it significantly increases the likelihood of playback rebuffering. Considering that nowadays most short video bitrates are below 2Mbps (as shown in Section III), improving the performance of throughput lower than 2Mbps becomes crucial for optimizing short video streaming.

One way to enhance the performance in low throughput scenarios is to improve congestion control algorithms. However, despite decades of research and numerous proposed algorithms for different application and network scenarios, such as Cubic [6], BBRv1 [7], Copa [8], and various machine learning-based algorithms [9], [10], the challenge of optimizing the performance of low throughput scenarios in short video services is still under-explored.

As a leading provider of short video services with proprietary video players and content delivery network (CDN) capabilities, we are in a unique position to optimize the performance of congestion control algorithms. Specifically, we first conducted a comprehensive measurement study on our production network through instrumenting both video players and edge servers. Our study (as shown in Sec. III-B) revealed that low throughput scenarios have significantly higher round-trip time (RTT) variations compared to non-low throughput scenarios. Moreover, the distribution of RTT variations also varies across different network types, such as cellular (e.g., 3G, 4G or 5G) and WiFi.

Second, motivated by the fact that Internet bandwidth resources are typically contention-based, we found that in production networks, having more competing connections often leads to a lower throughput for a connection when sharing a bottleneck (as shown in Section III-C). As the bitrate of each video is known to the service, we explore a novel bitrate-aware approach to optimize the transport's congestion control algorithm to improve streaming performance under competing-flow environments.

Based on the above observations, we conducted a pilot investigation on low throughput scenarios and proposed a new congestion control algorithm called BBR-E2E, which builds

upon the existing BBRv1 algorithm. We then deployed BBR-E2E in our production network and conducted the largest known experiment in production networks to date. This experiment involved over 600 million end-users and trillions of video sessions over a period of three months. Through A/B testing, we found that BBR-E2E achieved an overall reduction of 1.6% in rebuffering duration (with a 6.2% reduction in rebuffering count). Moreover, at the provincial level, BBR-E2E achieved up to a 7.8% reduction in rebuffering duration (with a 13.7% reduction in rebuffering count).

II. Related Works

Ensuring a high QoE for end-users, including smooth playback, higher video bitrate and minimal initial video startup delay, has become a crucial factor in increasing video watch duration and user engagement [4], [5]. Existing studies on short video streaming have usually focused on application layer optimization, e.g., video preloading [11]–[15], bitrate adaptation [12], [16], and transport protocol layer optimization, e.g., different protocols [17] and new congestion control algorithms [9]. We review the above two as follows.

Moreover, because currently BBRv1 (henceforth BBR for short) is the default congestion control algorithms for our short video streaming services, we also provide a brief introduction of BBR in this section.

A. Application Layer Optimization

Most research in application layer optimization focuses on video preloading and bitrate adaptation. Video preloading techniques, such as APL (Adaptive Preloading) [14], LiveClip [18], and Dashlet [13], aim to improve the user experience by proactively fetching and caching video content before it is needed and thus reducing buffering and minimizing interruptions. The basic ideas of bitrate adaptation algorithms [12], [16] for short video streaming is to dynamically adjust the bitrate of the video being streamed to match the user's network bandwidth, ensuring smooth playback without excessive buffering or degradation in video quality. Although above approaches can contribute to the overall performance gains of short video services, the QoE of short video service still fall behind expectations.

B. Transport Layer Optimization

Recent advances in the study of Internet congestion control can be categorized into two distinct approaches: the model-based and learning-based [9], [10]. Model-based algorithms are usually handcrafted by domain experts based on various insights into the network, such as CUBIC [6], Copa [8], and BBR [19], among others. However, because model-based algorithms are typically not tailored to specific network environments and Internet applications, they may underperform in certain applications and environments. Consequently, learning-based algorithms have emerged as potential alternatives, such as DeepCC [20], Gemini [21], Sage [22], and Indigo [23], which demonstrate superior performance in a variety of specific network scenarios and applications. Nonetheless, the high overhead associated with their deployment has hindered the large-scale adoption of these algorithms. In contrast, in this study, we propose optimizing the performance of existing model-based algorithms (i.e., BBRv1) specifically for short video services through collaboration with end-user playback, an area that is under-explored.

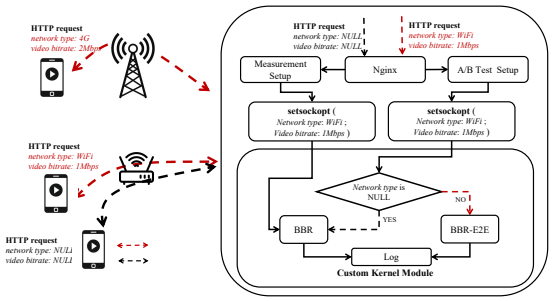

Fig. 1. The architecture of BBR-E2E.

C. BBR Revisited

In this subsection we summarized the major design features of BBR and interested reader may refer to [19] for more details.

The BBR model operates on the bottleneck link and detects it by continuously monitoring the delivery rate and RTT. BBR utilizes a state machine to continuously monitor the bottleneck link. Specifically, BBR operates through four machine states to continuously probe the available network bandwidth.

- STARTUP: Upon entering BBR, it starts in the STARTUP state, similar to traditional slow start, where it exponentially increases the data sent to probe the available bandwidth. If three consecutive bandwidth probes increase by less than 25%, it indicates a bandwidth bottleneck.
- DRAIN: After the aggressive transmission in the STARTUP phase, it enters the DRAIN state to flush the queues in the pipeline using an opposite gain of ln(2)/2. It then transits to ProbeBW.
- ProbeBW: This is the primary state of BBR, similar to congestion avoidance. It uses periodic pacing gains [5/4, 3/4, 1, 1, 1, 1, 1, 1] to control the sending rate. During the 5/4 gain, it probes for available bandwidth, while during the 3/4 gain, it flushes the bandwidth-probing queues. It then maintains a stable execution for six cycles.
- ProbeRTT: ProbeRTT is a specialized state within BBR, aimed at determining the round-trip propagation time (RTprop) of the link. It activates when there's been no update on the minimum RTT for a duration of 10 seconds. Subsequently, it deliberately reduces the congestion window to a maximum of 4 packets and sustains this reduced window size for at least 200 milliseconds. This intentional action facilitates the depletion of any potential queue within the link, thus enabling an accurate estimation of RTprop through the measured minimum RTT.

TABLE I
METRICS OF EACH VIDEO REQUEST

Metrics	Description
Time	The date/hour/minute/second when the request was received.
Net type	The access network type of the video player,i.e., cellular, WiFi.
Bitrate	The video's mean bitrate requested by the video player.
Client IP	The public IP address seen by the edge server.
Request size	The bytes of a video chunk request acknowledged by client.
Request dur.	The duration it takes to deliver a video chunk request.
$srtt$	The latest Smoothed Round-Trip Time (SRTT) value at the end of the request.
rtt_var	The latest Round-Trip Time variation (RTT_VAR) value at the end of the request.

III. MEASUREMENT AND MOTIVATION

In this section, our objective is to gain insights into the behavior of BBR in a production network, especially under low throughput scenarios, and identify opportunities to enhance its performance specifically for short video streaming applications.

To achieve the above goal, we conducted a week-long measurement study in a province in China. In our setup, video players report the video bitrate and network type to the edge servers using HTTP requests, as illustrated in Fig. 1. At the edge server, we implemented a customized kernel module serving two goals: run either BBR algorithm or BBR-E2E algorithm, and log information collected by the kernel module. Upon receiving each HTTP request, the edge server forwards the parsed request to the kernel module by invoking the system call *setsockopt(·)* [24]. In the measurement study setup, the kernel module only runs BBR. In the A/B test setup in Section VI, the kernel module checks whether bitrate and network type information in *setsockopt(·)* call exists. If so then the kernel module runs BBR-E2E; otherwise, it runs BBR. After each request is completed, the edge server records relevant TCP metrics, as listed in Table I, along with the reported network type and video bitrate. These metrics are stored in a centralized database for further analysis.

Although congestion control optimization alone cannot address scenarios where the underlying available bandwidth is insufficient for the video bitrate, there are still two potential directions for improving performance in low-throughput scenarios. First, if the available bandwidth is sufficient but the congestion control algorithm does not fully utilize it, there is room for improvement by optimizing the congestion control algorithm. Second, in the presence of competing traffics, the short-video connection may not grab enough throughput to sustain the video bitrate. In the following sub-sections, we will evaluate the performance of BBR in the production network to identify potential areas for improvement. Additionally, we will assess the feasibility of adapting BBR's aggressiveness in the presence of competing traffics.

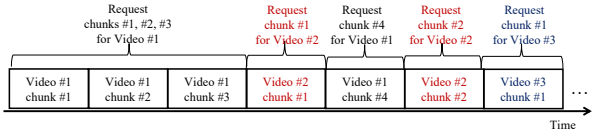

Fig. 2. An example of short video request behavior.

A. Request Behavior of Short Video Applications

The request behavior of short video services presents unique features that differ from long videos and thus presents a unique challenge to congestion control algorithms. As illustrated in Fig. 2, a short video is typically divided into several chunks, and the video player sequentially requests these chunks, with the chunk sequence requests being monotonically increasing for each video. Current video players usually employ persistent HTTP connections for different video chunks, meaning multiple video chunk requests are served by only one TCP connection sequentially. In normal cases, each TCP connection utilizes a single congestion control algorithm. Due to the involvement of video preloading algorithms, the video player may initiate a new video request (e.g., video #2) even before the previous one (e.g., video #1) is fully downloaded.

Additionally, though persistent HTTP connection is used, the video player usually establishes multiple TCP connections to servers (which may not be the same edge server). It is worth noting that only one TCP connection is active in requesting video chunks at any given time, while the other TCP connections remain alive but idle. A typical scenario for activating idle TCP connections occurs when the end-user chooses to watch a new video before the old video has finished downloading. To avoid head-of-line blocking for new video requests, the video player will close the current TCP connection to drop the old video content and start downloading new video chunks by activating a backup TCP connection.

Since the majority of users of short video services are mobile users (e.g., mobile phones or tablets), and due to limitations in screen size and economic concerns regarding high-bitrate video provision, the commonly used video bitrates for short videos are generally lower compared to long videos that are typically viewed on larger screen devices such as laptops or TVs. The distribution of video bitrate requests by users is depicted in Fig. 3, where it is evident that over 90% of video bitrates are less than 2 Mbps. As a result, the chunk sizes are also much smaller than those in conventional on-demand streaming services. Since we generally employ quality-based video encoding instead of fixed bitrate encoding, the video content will have a significant impact on the resulting mean bitrate. Thus, there may not be a clear discrete pattern in the distribution, as shown in Fig. 3. We also show the request size and duration of each chunk in Fig. 4 and Fig. 5, respectively. Since the developer of the video player may empirically configure a fixed chunk request size based on video quality to control the size of video preloading and video

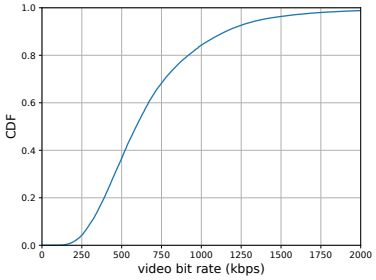

Fig. 3. Distribution of video bitrate.

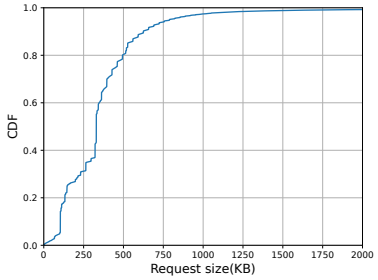

Fig. 4. Distribution of request size.

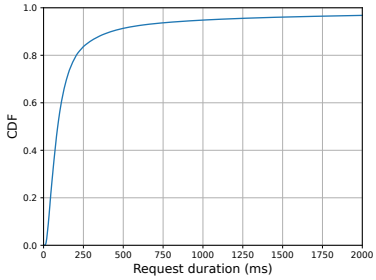

Fig. 5. Distribution of request duration.

playback buffer, we may see the distribution of request sizes concentrate around certain fixed values, e.g., 104 KB, 330 KB, as shown in Fig. 4. However, it is still clear that over 90% of request sizes are smaller than 1000 KB, and over 90% of request durations are less than 1 second. By comparing typical design features in BBR, where in the ProbeRTT state, BBR requires 10 seconds to obtain the round-trip propagation time of the link (as shown in Section II-C) and uses it as a key metric to drive congestion control, it is necessary to investigate the potential impact of these unique features in production short video services, especially when the network experiences significant RTT variations within 10 seconds.

B. Characteristics of Mobile/Wireless Networks

Existing studies (e.g., [25]–[27]) have reported that BBR can perform sub-optimally in networks with large RTT variations; that is, the higher the RTT variation, the lower the bandwidth utilization. However, this issue has not yet been investigated in any large-scale production services. To fill this gap, we recorded the $srtt$ and rtt_var, computed from Eq. 1 and Eq. 2 according to RFC2988 [28].

$$srtt = 7/8 \times srtt + 1/8 \times rtt. \quad (1)$$

$$rtt\_var = 3/4 \times rtt\_var + 1/4 \times |srtt - rtt|. \quad (2)$$

The measurements were taken at the end of each HTTP request as a measure of RTT and its variations in the network. Note that rtt in Eq. 1 and Eq. 2 measures the time elapsed between when a data packet is sent to the network by the server and when the packet acknowledgment is received by the server. Therefore, the values of $srtt$ and rtt_var are mostly determined by the most recent several TCP acknowledgments according to Eq. 1 and Eq. 2. We use $srtt$ instead of rtt to obtain a more stable RTT measurement of the network and to avoid transient spikes that might be measured with rtt. In Fig. 6 and Fig. 7, we show the distribution of $srtt$ and rtt_var for different network types in our production networks, based on the video player's feedback on its access network type.

Given that both the video bitrate (provided by the player) and the mean throughput (estimated by the server from the chunk size and chunk transfer time) are known, we can classify a chunk request to be either with sufficient throughput when the estimated throughput is higher than the bitrate or insufficient throughput otherwise. For ease of illustration, we say video requests is with *insufficient throughput* if the video bitrate is lower than mean estimated throughput, and *sufficient throughput* otherwise. We then examined the characteristics of RTT in both cellular and WiFi networks, aiming to uncover correlations between network types, RTT dynamics, and throughput sufficiency.

Initially, we scrutinized the smoothed RTT ($srtt$) metric and observed a consistent trend: $srtt$ in cellular networks is consistently higher than that in WiFi networks (see Fig. 6). This discrepancy underscored inherent dissimilarities between the two network infrastructures. However, upon further analysis, contrasting $srtt$ of requests with and without sufficient throughput revealed that the disparity between these two categories was statistically insignificant. This suggests that while $srtt$ serves as a marker for network type, it may not directly correlate with throughput sufficiency.

In contrast, our examination of the RTT variation (rtt_var) metric yielded more distinct results. Fig. 7 illustrates discernible differences between network types and throughput sufficiency levels. Notably, cellular networks exhibited consistently larger rtt_var values compared to WiFi networks, while requests with insufficient throughput consistently manifested higher rtt_var values than those with sufficient throughput for the same network type.

Recognizing the interplay between $srtt$ and rtt_var, we introduced a novel metric: the normalized RTT variation ($rtt\_var/srtt$), aimed at normalizing RTT variations across $srtt$. This metric depicted in Fig. 8 revealed pronounced differences between requests with insufficient and sufficient throughput. Particularly noteworthy was the finding that over 50% of requests with insufficient throughput exhibited a normalized RTT variation exceeding 0.2, compared to only 20% for requests with sufficient throughput.

Next, we extend our investigation to network provider variations within a single province. Surprisingly, while the general trend depicted higher normalized RTT variations in cellular networks compared to WiFi networks as shown in Fig. 8, this pattern was not universally observed across different Internet Service Providers (ISPs) within the same region (Fig. 9). This underscores the complexity of network dynamics and the need for new metrics in network characterization.

Given the variability in RTT observed in real-world net-

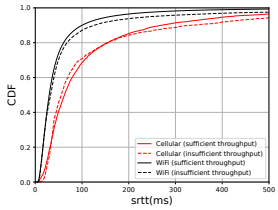

Fig. 6. Distribution of $srtt$ in production networks.

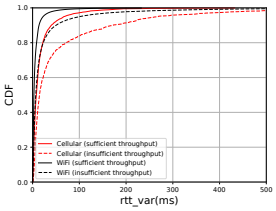

Fig. 7. Distribution of rtt_var in production networks.

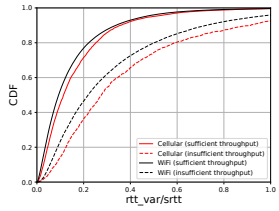

Fig. 8. Distribution of the normalized RTT variation ($rtt\_var/srtt$) in production networks.

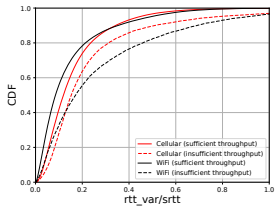

Fig. 9. RTT variation ($rtt\_var/srtt$) in production networks of a different ISP.

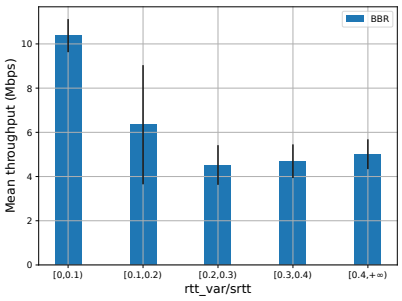

Fig. 10. Empirical study of different $rtt\_var/srtt$ v.s. mean throughput under 10Mbps bandwidth capacity with different normalized RTT variations.

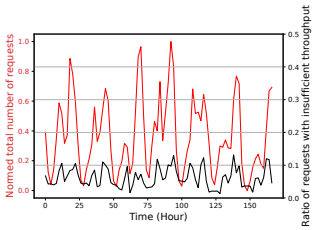

Fig. 11. Temporal variations of the number of requests and ratio of requests with insufficient throughput.

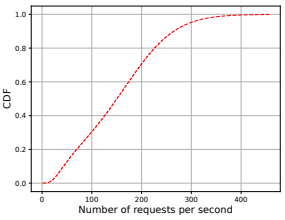

Fig. 12. Distribution of number of requests per second per client IP.

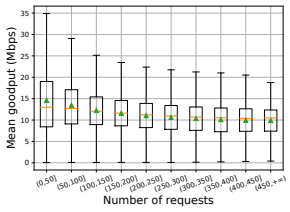

Fig. 13. Boxplot [30] of the distri. of the mean of goodput (MBps), green triangle repsents the mean, orange bar represents median.

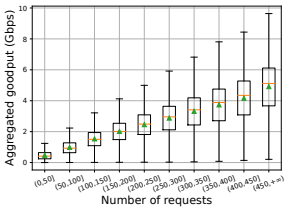

Fig. 14. Boxplot [30] of the distri. of the sum of goodput (GBps), green triangle repsents the mean, orange bar represents median.

works, it is not straightforward to directly link it to performance issues with BBR. To address this gap, we conducted controlled throughput tests using Traffic Control (TC) tools [29] to emulate different levels of RTT variations. The variations in the real world might be caused by various factors, such as network conditions (e.g., mobility issues, channel fading, routing issues) or background traffic. However, we cannot determine which specific factors contribute to the resulting RTT variations in production networks. To quantitatively evaluate how BBR performs under different RTT variations, we varied RTT jitter from 0 ms to 25 ms on a network with a base RTT of 50 ms and a bandwidth of 10 Mbps. After conducting over 200 tests, we recorded the corresponding $rtt\_var/srtt$ values and mean throughput at the end of each test. Our results, shown in Fig. 10, reveal a clear trend: when $rtt\_var/srtt$ exceeds 20%, throughput drops to around 5 Mbps, indicating 50% bandwidth utilization. This demonstrates the negative impact of increased RTT variation on BBR performance. Therefore, it is crucial to consider high RTT variation in production networks when optimizing congestion control algorithms. Note that we use TC to emulate the network limit based on token bucket rate limiting. Due to burst packets at the beginning of each TCP connection, the overall throughput is slightly higher than 10 Mbps.

C. Competing Video Flows

Given the relatively long duration of a persistent video connection, it is not uncommon for it to encounter competing traffics from time to time, e.g., due to background data transfer from within or outside the video app, competing users sharing the same WiFi AP, etc. Consequently, the achieved throughput may fall below the video bitrate, leading to playback rebuffering even if the access network bandwidth is in fact sufficient. This motivates us to investigate the feasibility of dynamically competing for bandwidth based on the required bitrate. This involves adjusting the aggressiveness of BBR's bandwidth probing to grab more bandwidth when the current throughput is insufficient for the video bitrate, or relinquishing some bandwidth when the throughput is already sufficient. It's worth noting that, in the current design of video players, although multiple parallel TCP connections to servers may exist, only one video request is active at any given time, thereby reducing the impact of interplay between parallel requests from the same player.

Intuitively, among the multiple video flows sharing the same bottleneck, some may experience insufficient bandwidth (e.g., due to a lower video bitrate), while others may have more than enough bandwidth. Thus, by increasing the aggressiveness of

the former and reducing it for the latter, we can dynamically reallocate bandwidth among them. To assess the feasibility of this idea, we conducted a measurement to correlate request density with throughput sufficiency and presented the results in Fig. 11. First, we plotted the normalized total number of requests during two-hour intervals over the course of one week with the red curve. Next, we showed the ratio of requests with insufficient throughput (the number of requests with inadequate throughput divided by the total number of requests) with the black curve. The results clearly indicate a strong correlation between the rate of insufficient requests and the normalized total number of requests. In other words, a higher number of total requests leads to a higher rate of requests with insufficient throughput.

Next, to enable throughput control based on video bitrate requirements, it is necessary to have parallel request connections sharing the same bottleneck. While precisely identifying the location of the bottleneck is challenging due to NAT devices and varying resource allocation mechanisms of ISPs, we assume that the bottleneck typically lies in the last mile because our CDN is usually deployed near the last mile. Based on this assumption, we plotted the distribution of video requests from the same client IP address at each second (with one-second granularity). Note that due to the limited number of IPv4 addresses, multiple end-users may share the same client IP address with the help of NAT devices to access edge servers. The results (shown in Fig. 12) demonstrate that we do have requests sharing the same IP address, even when considering only our own services and excluding background traffic from third parties.

It is worth noting that as the number of parallel requests increases with a given client IP address within the same time period, the mean throughput per connection decreases (Fig. 13), indicating the presence of requests sharing the same bottleneck. However, the aggregated goodput (total number of bytes acknowledged for all requests from the same IP address within one-second time intervals, as shown in Fig. 14) continues to increase. This suggests that these parallel connections may not all share a single bottleneck, e.g., multiple WiFi hotspots behind the same NAT.

Overall, the measurements provide evidence supporting the motivation of dynamically grabbing more bandwidth or relinquishing part of it in production networks based on video bitrate requirements.

IV. The Design of BBR-E2E

Based on the above two insights, BBR-E2E introduces two optimizations accordingly. The first optimization aims to enhance BBR's performance on requests with insufficient throughput in mobile/wireless networks by adapting to different RTT variations of mobile/wireless networks. In addition, we implement a bandwidth probing aggressiveness control mechanism based on video bitrate requirement to mitigate throughput insufficiency caused by contention.

Algorithm 1 BDP estimation in BBR-E2E

```
Input:
RTTVAR_THRESH ← RTT variation threshold
TCP variables: rtt_var, srtt, min_rtt
BBR variables: max_bw
Output:: Congestion Window (CWnd)
if rtt_var > srtt * RTTVAR_THRESH then
    CWnd = 2 * max_bw * srtt
else
    CWnd = 2 * max_bw * min_rtt
end if
```

A. Adapting to RTT Variations

In existing congestion control algorithm design, the control of the congestion window plays a key role in its overall performance. If the congestion window is underestimated, it can lead to bandwidth under-utilization. Conversely, overestimating the congestion window can result in buffer bloat [31], flooding the network with excessive packets.

BBR relies on the bandwidth-delay product (BDP), calculated as the maximum delivery rate multiplied by the min_rtt, as a crucial reference for managing congestion windows. However, its slow updates of the minimum round-trip time (min_rtt), which can take up to 10 seconds, pose a challenge for the ever-changing network conditions due to factors such as user mobility or background traffic.

To adapt to the changing network conditions, one straightforward approach is to use $srtt$ as a replacement for min_rtt to compute the congestion window size. However, this can introduce buffer bloat [31] and potentially degrade performance further. To tackle this challenge, motivated by the measurement results in Section III, we use $rtt\_var/srtt$ as a measure of the current degree of normalized RTT variation. We then utilize a threshold, RTTVAR_THRESH (e.g., set to 20%), to control when to use $srtt$ or min_rtt in the process of determining the congestion window, as outlined in Algorithm 1. The underlying intuition is as follows: if the current RTT variation falls below the threshold, we assume it to be an occasional fluctuation, whereas if it exceeds the threshold, we consider it a persistently high RTT variation that degrades BBR's efficiency as demonstrated earlier. Fine-tuning the RTTVAR_THRESH or further optimization based on different network types can be explored as part of our future work. Note that although the video player provides us with different network types, we did not observe significant differences in $rtt\_var/srtt$ between the two (as shown in Section III), and thus, both network types share the same threshold in our design.

B. Adapting to Throughput Sufficiency

To meet the throughput requirements of short video bitrates and minimize potential playback buffering, we propose a control mechanism to adjust the aggressiveness of the BBR algorithm in accordance with the request's throughput sufficiency, as depicted in Fig. 15.

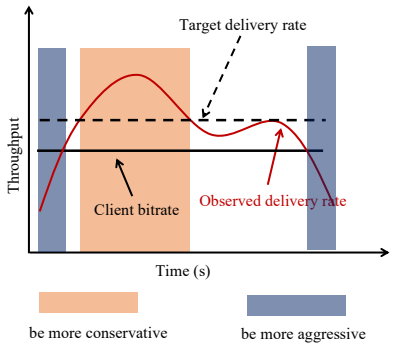

Fig. 15. Pacing rate control behavior with a target delivery rate.

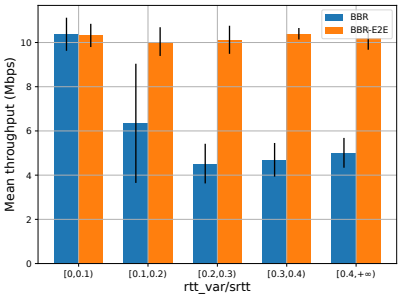

Fig. 16. BBR-E2E vs BBR in the same emulated networks (with 10Mbps bandwidth capacity, 50ms base RTT and different jitters).

Algorithm 2 Aggressiveness control given video bitrate

1: **Variables:**
2: $v \leftarrow$ video bitrate from video player
3: $r \leftarrow (1+\alpha) \times v$ # target delivery rate
4: $B \leftarrow$ measured delivery rate
5: $probeBW\_para \leftarrow$ [5/4,3/4,1,1,1,1,1,1]
6: **BBR_MODE in BBR_STARTUP**
7: **If** $B \geq r$
8: Exit BBR_STARTUP mode
9: **BBR_MODE in BBR_PROBE_BW**
10: **If** $B \geq r$
11: $probeBW\_para \leftarrow$ [3/4,3/4,1,1,1,1,1,1]
12: **Else if** $B < v$
13: $probeBW\_para \leftarrow$ [5/4,5/4,1,1,1,1,1,1]
14: **Else**
15: $probeBW\_para \leftarrow$ [5/4,3/4,1,1,1,1,1,1]

The core idea of our approach is to compare whether the measured delivery rate (computed by the total number of bytes acknowledged by the client in the recent round trip time divided by the value of the measured round trip time, denoted by B) can match the rate of video playback consumption, i.e., video bitrate (denoted by v). Specifically, if B is higher than the video playback consumption rate, it becomes conservative in bandwidth probing in the congestion control algorithm. Conversely, if B is lower than the playback consumption rate, it becomes more aggressive in probing bandwidth to achieve higher throughput. Since no buffer occupancy of the video player is provided at the time of request, to avoid potential playback rebuffering due to erroneous over-conservative bandwidth probing, we introduce a safety margin for the target delivery rate based on measured throughput, denoted as α (e.g., 30% over the video bitrate), to account for network fluctuations.

We show a detailed design of our algorithm in Algorithm 2. First, to optimize the BBR algorithm, we modify the conditions for state transitions. For instance, in the BBR STARTUP phase, instead of relying solely on the condition of not detecting a 25% increase in bandwidth for three consecutive probes as the exit criteria, we compare B with the target delivered rate r. We exit the STARTUP phase when B exceeds the target delivery bytes ($B \geq r$). This adjustment allows for an earlier exit from the STARTUP phase, relinquishing bandwidth for other flows, e.g., video flows with insufficient throughput.

Furthermore, we control the aggressiveness by modifying the gain of the *probe_bw* parameter based on the target delivery rate in the BBR ProbeBW phase. Our approach aims to be more aggressive in probing bandwidth when the throughput falls short of meeting the consumption rate for video playback (as shown in *Line* 13). Conversely, we adopt a more conservative approach to probing bandwidth when the available throughput is sufficient for playback (as shown in *Line* 11).

V. Performance Evaluation

In this section, we will begin by evaluating our two optimizations individually, i.e., the optimization considering RTT variations *only* and the optimization considering video bitrate *only*, using both emulated networks and small-scale A/B tests in production networks. Afterward, we will shift our focus to the fairness issue introduced by BBR-E2E. We will present and discuss the challenges related to achieving fairness in the context of BBR-E2E.

A. Performance with Optimization for RTT Variations

We evaluate the performance gain of the optimization described in Section IV-A in BBR-E2E over BBR in emulated networks. Note that we conducted emulations with different bandwidth capacities (e.g., 5 Mbps, 10 Mbps and 20 Mbps), base RTTs (e.g., 100 ms and 200 ms), and jitter values (e.g., 0 ms to 200 ms), and achieved similar results. In this paper, we report the results with a bandwidth capacity of 10 Mbps, a base round-trip time (RTT) of 50 ms, and varying jitter levels. Experiments were conducted using the iperf [32] tool, with each test repeated 30 times and lasted for 60 seconds. Mean and standard deviation of throughput across different jitter values are presented in Fig. 16. The results demonstrate that as jitter values increase, the mean throughput of BBR decreases from 10 Mbps to approximately 4 Mbps. This reduction corresponds to a decrease in bandwidth utilization from 100% to around 50%. By contrast, BBR-E2E achieves consistently higher throughput across different normalized RTT variations.

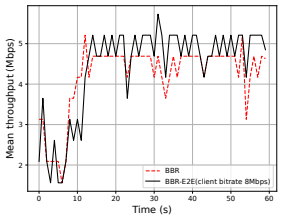

Fig. 17. A bandwidth grabbing case of BBR-E2E (with 10Mbps bandwidth capacity, 50ms base RTT).

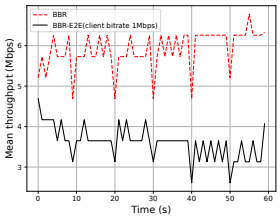

Fig. 18. A bandwidth relinquishing case of BBR-E2E (with 10Mbps bandwidth capacity, 50ms base RTT).

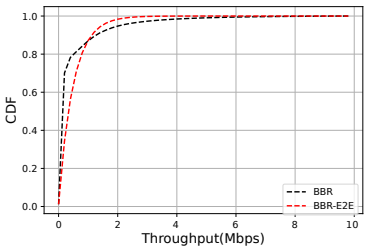

Fig. 19. Throughput comparison between BBR and BBR-E2E.

However, the optimization carries the risk of increasing packet retransmissions or congestion in the production network. This risk is managed through threshold RTTVAR_THRESH, as detailed in Section IV-A. In our evaluation, we conducted an A/B test using two servers with same workloads. The results revealed that BBR achieved an average retransmission rate of around 2% (calculated by dividing the total number of retransmitted bytes by the total number of bytes cumulatively acknowledged by the clients), while BBR-E2E exhibited a higher average retransmission rate of around 2.5%. We plan to conduct further fine-tuning of the threshold through additional experiments in our production network.

B. Performance with Optimization for Video Bitrate

In this section, we first provided an illustrative validation of the aggressiveness control in BBR-E2E. Specifically, we show two flows sharing the same bottleneck with a link capacity of 10 Mbps and base RTT of 50ms: one flow with BBR and another one with BBR-E2E, given different client video bitrates. We can see that BBR-E2E with a higher client bitrate (e.g., 8 Mbps) than equal share (e.g., 5 Mbps) tends to contend for more bandwidth (as shown in Fig. 17), and by contrast, BBR-E2E with a lower client bitrate (e.g., 1 Mbps) tends to offer more bandwidth to the background (as shown in Fig. 18).

Next, to evaluate the effectiveness of aggressiveness control in production networks, we did an A/B test using two servers with same workloads. We have one server running BBR-E2E with target delivery rate based on video bitrate as described in Algorithm 2 and another one running default BBR. A total of over 2 million requests are collected during the tests. We show the throughput of two servers in Fig. 19. It is clear to us that BBR-E2E with target delivery rate achieves higher throughput

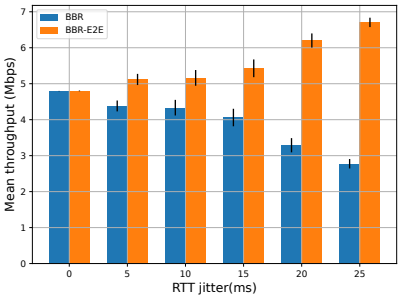

Fig. 20. BBR-E2E for RTT variations v.s. BBR co-exist in the same emulated bottleneck (with 10Mbps bandwidth capacity, 50ms base RTT and different jitters).

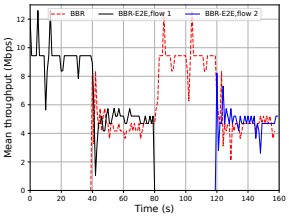

Fig. 21. Fairness with 10Mbps bandwidth capacity, 50ms base RTT and no extra jitter.

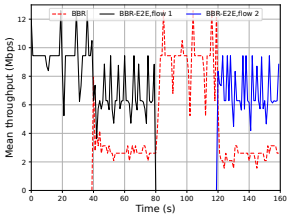

Fig. 22. Fairness with 10Mbps bandwidth capacity, 50ms base RTT and extra 25ms jitter.

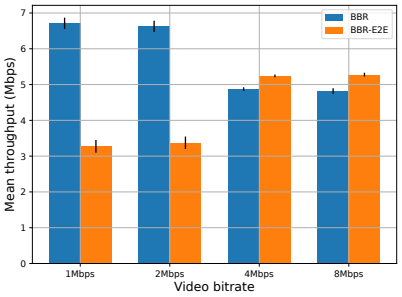

Fig. 23. BBR-E2E with different client bitrate v.s. BBR co-exist in the same emulated bottleneck (with 10Mbps bandwidth capacity, 50ms base RTT).

distribution when the throughput is below 1 Mbps (i.e., the cross point of two curves), but BBR has a higher proportion of throughput larger than 1 Mbps. Though BBR may achieve higher mean throughput, it in fact may not be needed given the video bitrate. In contrast, BBR-E2E's higher throughput under challenging network conditions can contribute to lower playback rebuffering.

C. Fairness Comparison with BBR

Since BBR-E2E is based on BBR and the two optimizations in BBR-E2E involve different degrees of aggressiveness control, it is necessary to evaluate how it interacts with BBR when sharing the same bottleneck. First, we evaluate the fairness of BBR-E2E with the optimization considering RTT variations under different jitters with a 10 Mbps bandwidth capacity, as shown in Fig. 20. It is expected that replacing min_rtt

with $srtt$ will introduce more aggressiveness to BBR-E2E, and the value of RTTVAR_THRESH controls this aggressiveness. This expectation is confirmed, as we observe that BBR-E2E progressively grabs more bandwidth from BBR for jitter values greater than 0. Additionally, we plot the microscopic behavior of BBR-E2E when competing with BBR in Fig. 21 and Fig. 22. In both figures, the first BBR-E2E flow starts at 0 seconds and stops at 80 seconds. The BBR flow starts at 40 seconds and continues until 160 seconds. The second BBR-E2E flow starts at 120 seconds and stops at 160 seconds. This setup is used not only to show the stable bandwidth share of competing flows but also to evaluate whether the second flow can share the bandwidth when the first flow is already present. It is evident that BBR-E2E exhibits a fair share with BBR when no extra jitter is added to the network emulator, but it does grab bandwidth from BBR in networks with more jitter. It is a limitation of BBR-E2E and warrants further study in our future work. In addition to tuning the value of RTTVAR_THRESH as discussed in the previous section, we will further investigate triggering such optimization when the actual throughput is lower than the required bitrate according to the video players' feedback in our future work.

Next, we evaluate the fairness for BBR-E2E with target delivery rate larger than fair share (e.g., 8 Mbps) and smaller than fair share (e.g., 1 Mbps, 2 Mbps, and 4 Mbps), as shown in Fig. 23. As expected, BBR-E2E with a higher video bitrate grabbed more bandwidth for client bitrate higher than the equal-share bandwidth when competing with BBR, and relinquished bandwidth when the client bitrate requirement is less than the available bandwidth.

Overall, BBR-E2E is more aggressive under higher RTT variations but becomes more conservative when the achieved throughput is larger than the required video bitrate.

VI. Large-scale deployment of BBR-E2E

We conducted two large-scale experiments for the performance evaluation of BBR-E2E. In each experiment, we employed controlled user groups. Each user was randomly tagged as group A or B, and the tags were stored in the HTTP request header for video requests. The server parses the HTTP header to retrieve the group tag and then uses Linux *setsockopt(·)* to select either the BBR or BBR-E2E algorithm for each group (as shown in Fig. 1). Note that the group tags are assigned randomly to each user, rather than to each video request. Because users are tagged without prior knowledge of their habits and preferences for watching short videos (e.g., heavy users vs. light users, user preferences on video content), this may introduce some random system bias into the experiment. To mitigate such an impact, we first conducted a 7-week-long experiment, and then swapped user tags and ran a second 5-week-long experiment to validate the results from the first experiment. The entire experiment involved over 600 million users and trillions of video playback sessions. Due to the massive volume of data, client performance was aggregated and anonymized to generate comparative performance metrics. In both experiments, the results consistently

TABLE II
Reduction in rebuffering with different network types (%)

Network Type	WiFi	Cellular
Normalized rebuffering duration	1.48	2.49
Normalized rebuffering count	6.17	6.44

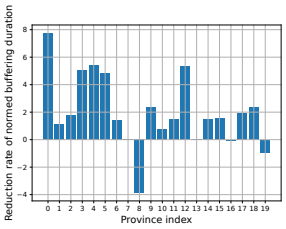

Fig. 24. Reduction in rebuffering duration with different provinces.

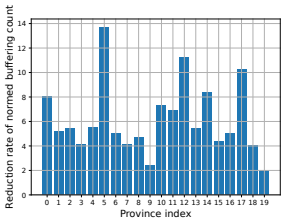

Fig. 25. Reduction in rebuffering count with different provinces.

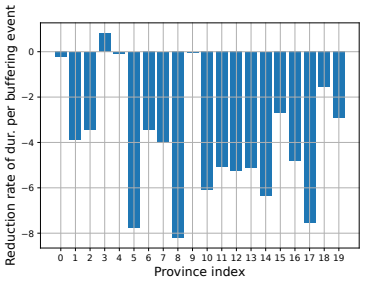

Fig. 26. Reduction in mean rebuffering duration for each buffering event with different provinces.

demonstrated the benefits of BBR-E2E compared to BBR in optimizing playback rebuffering in short video services. In this study, we report details from the second experiment, including aggregated client QoE data from different provinces and networks.

Since both rebuffering duration and rebuffering count negatively impact users' quality of experience during video playback, it is important to optimize both performance metrics. However, it is evident that as watch duration increases, there is a tendency for rebuffering count and rebuffering duration to also increase. To eliminate potential bias by different total watch durations, we normalize the total rebuffering duration and buffering counts by dividing them by the total watch duration of all the video playback sessions. Specifically, we sum up the rebuffering duration (denoted by d_i for ith session) and rebuffering count (denoted by c_i for ith session) for each playback session, as well as the watch duration (denoted by w_i for ith playback) then the normalized playback rebuffering duration (denoted by λ)/rebuffering count (denoted by γ) by Eq. 3 and Eq. 4, respectively.

$$\lambda = \sum_i d_i / \sum_i w_i \tag{3}$$

$$\gamma = \sum_i c_i / \sum_i w_i \tag{4}$$

TABLE III
PROPORTIONS OF VIDEO QUALITY REQUESTED BY END USERS (%)

Algorithm	1080P	720P	540P	480P	360P	unknown
BBR	8.47	60.61	27.82	0.12	0.08	2.90
BBR-E2E	8.48	60.62	27.81	0.12	0.08	2.89

Regarding the normalized rebuffering duration, BBR-E2E achieves an average reduction of 1.6% compared to BBR. Further analysis across 20 provinces in China reveals that BBR-E2E demonstrates rebuffering reduction in 17 out of 20 regions, with reductions of up to 8% at the provincial level (as shown in Fig. 24). However, BBR-E2E exhibits subpar performance in 3 regions. This observed deviation could stem from the downsides of the optimizations outlined in Sections V-A and V-B, which may outweigh their benefits in regions characterized by diverse network conditions. This suggests that the current optimizations in BBR-E2E may not be optimally tailored to such network conditions, highlighting the need for further refinement. Additionally, comparing the performance of BBR-E2E across different network types, we observe rebuffering reduction in both cellular and WiFi networks (as depicted in Table II).

In terms of the normalized rebuffering count, BBR-E2E achieves an average reduction of 6.2% compared to BBR. Lower rebuffering counts with BBR-E2E are consistently observed across different provinces, with reductions of up to 14% at the provincial level (as illustrated in Fig. 25). Interestingly, unlike the observations in rebuffering duration, no subpar performance is noted in terms of rebuffering count across provinces. Our further investigation reveals that BBR-E2E may reduce more short-time rebuffering of video sessions but could potentially introduce long-time rebuffering as shown in Fig. 26, especially for province index #8 and #19. Furthermore, BBR-E2E demonstrates rebuffering count reductions in both cellular and WiFi networks (as depicted in Table II). These results clearly demonstrate the effectiveness of the two optimizations introduced in BBR-E2E.

In addition to rebuffering, video bitrate and startup delay are two other crucial performance metrics that significantly impact users' quality of experience in short video services. In our short video service, video quality (encoded with variable bitrate but fixed quality) remains constant during playback, thus eliminating the need for adaptive video bitrate logic for a single video. However, we offer multiple versions of each video with varying quality, and the video player selects the appropriate quality based on historical download bandwidth, where higher recent throughput may result in higher video quality in subsequent video requests. In our experiment, we observed negligible differences (below 0.01%) in the distribution of each video quality (e.g., 720p and 1080p) watched by users in our two A/B test groups (as shown in Table III).

Furthermore, during extensive A/B testing, we did not observe any significant performance gains or losses in terms of initial startup delay. We speculate that the effectiveness of the video preloading algorithm plays a significant role, potentially mitigating the potential gains introduced by TCP congestion control mechanisms. As initial startup delay remains an important metric, further work is warranted to explore new ways to optimize it.

VII. CONCLUSION AND FUTURE WORKS

In our study, we conducted a pilot investigation focused on optimizing congestion control algorithms specifically for short video services. By leveraging feedback from the video player, we proposed an enhanced algorithm called BBR-E2E, which improves the performance of the existing BBR algorithm. Through a large-scale experiment, which represents the first known experiment at the 100 million users and trillion video sessions level, we demonstrated that BBR-E2E can substantially improve the streaming performance of real-world short-video services.

Our A/B experiments demonstrated that BBR-E2E effectively reduces the overall rebuffering duration by 1.6% (and rebuffering count by 6.2%). Notably, certain areas experienced even more remarkable improvements, with reductions of up to 7.8% in rebuffering duration and 13.7% in rebuffering count. These findings shed light on the under-explored research field of low throughput scenarios, which play a critical role in optimizing the performance of short video services in production networks.

The insights gained from our extensive A/B experiments, which leverage collaborative optimization between user-end devices and edge servers, provide valuable guidance for industry practitioners. Moreover, they pave the way for future exploration and advancements in this field, benefitting both the industry and academic researchers.

Moving forward, our ongoing work focuses not only on further optimizing the performance of short video services based on existing feedback information, such as network type and video bitrate, but also on investigating additional metrics from video players, including rebuffering and video segment priority. Additionally, we recognize the significance of addressing the optimization of the initial startup delay of short videos as another key performance metric, which we plan to explore in future research endeavors.

VIII. ACKNOWLEDGEMENT

We would like to thank our shepherd Arvind Narayanan and the anonymous reviewers for their suggestions in improving this paper. This work was supported in part by the HKSAR RGC General Research Fund (GRF/14219022).

REFERENCES

[1] T. Newsroom, "Thanks a billion!" 2021. [Online]. Available: https://newsroom.tiktok.com/en-us/1-billion-people-on-tiktok

[2] Grand View Research, "Short video platforms market size, share & trends analysis report," 2030, report ID: GVR-4-68040-050-8, Number of Pages: 110, Format: Electronic (PDF). [Online]. Available: https://www.grandviewresearch.com/industry-analysis/short-video-platforms-market-report

[3] Vision Research Reports, "Short video platforms market size, share, growth, trends — report 2023-2032," https://www.visionresearchreports.com/short-video-platforms-market/40148, 2023, accessed on January 13, 2024.

[4] H. Zhang, Y. Ban, Z. Guo, Z. Xu, Q. Ma, Y. Wang, and X. Zhang, "Quty: Towards better understanding and optimization of short video quality," in *Proceedings of the 14th Conference on ACM Multimedia Systems*, 2023, pp. 173–182.

[5] Y. Zhang, Y. Liu, L. Guo, and J. Y. Lee, "Measurement of a large-scale short-video service over mobile and wireless networks," *IEEE Transactions on Mobile Computing*, 2022.

[6] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[7] S. Vargas, G. Gunapati, A. Gandhi, and A. Balasubramanian, "Are mobiles ready for bbr?" in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 551–559.

[8] V. Arun and 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.

[9] H. Jiang, Q. Li, Y. Jiang, G. Shen, R. Sinnott, C. Tian, and M. Xu, "When machine learning meets congestion control: A survey and comparison," *Computer Networks*, vol. 192, p. 108033, 2021.

[10] T. Zhang and S. Mao, "Machine learning for end-to-end congestion control," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 52–57, 2020.

[11] S. Zhu, T. Karagioules, E. Halepovic, A. Mohammed, and A. D. Striegel, "Swipe along: a measurement study of short video services," in *Proceedings of the 13th ACM Multimedia Systems Conference*, 2022, pp. 123–135.

[12] N. T. Phong, T. T. Huong, P. N. Nam, T. C. Thang, and D. Nguyen, "Joint preloading and bitrate adaptation for short video streaming," *IEEE Access*, 2023.

[13] Z. Li, Y. Xie, R. Netravali, and K. Jamieson, "Dashlet: Taming swipe uncertainty for robust short video streaming," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 1583–1599.

[14] H. Zhang, Y. Ban, X. Zhang, Z. Guo, Z. Xu, S. Meng, J. Li, and Y. Wang, "Apl: Adaptive preloading of short video with lyapunov optimization," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, 2020, pp. 13–16.

[15] S.-Z. Qian, Y. Xie, Z. Pan, Y. Zhang, and T. Lin, "Dam: Deep reinforcement learning based preload algorithm with action masking for short video streaming," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 7030–7034.

[16] C. Zhou, S. Zhong, Y. Geng, and B. Yu, "A statistical-based rate adaptation approach for short video service," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2018, pp. 1–4.

[17] D. Wei, J. Zhang, H. Li, Z. Xue, Y. Peng, and R. Han, "Multipath smart preloading algorithms in short video peer-to-peer cdn transmission architecture," *IEEE Network*, 2023.

[18] J. He, M. Hu, Y. Zhou, and D. Wu, "Liveclip: towards intelligent mobile short-form video streaming with deep reinforcement learning," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2020, pp. 54–59.

[19] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, I. Swett, J. Iyengar, V. Vasiliev, and V. Jacobson, "Bbr congestion control: Ietf 99 update," in *Presentation in ICCRG at IETF 99th meeting*, 2017.

[20] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Wanna make your tcp scheme great for cellular networks? let machines do it for you!" *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 265–279, 2020.

[21] W. Yang, Y. Liu, C. Tian, J. Jiang, and L. Guo, "Gemini: Divide-and-conquer for practical learning-based internet congestion control," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[22] C.-Y. Yen, S. Abbasloo, and H. J. Chao, "Computers can learn from the heuristic designs and master internet congestion control," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 255–274.

[23] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, 2018, pp. 731–743.

[24] "setsockopt(2) - linux manual page," urlhttps://linux.die.net/man/2/setsockopt, [Online; accessed May 15, 2024].

[25] D. Scholz, B. Jaeger, L. Schwaighofer, D. Raumer, F. Geyer, and G. Carle, "Towards a deeper understanding of tcp bbr congestion control," in *2018 IFIP networking conference (IFIP networking) and workshops*. IEEE, 2018, pp. 1–9.

[26] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of bbr congestion control," in *2017 IEEE 25th international conference on network protocols (ICNP)*. IEEE, 2017, pp. 1–10.

[27] S. Zhu, T. Li, X. Ma, Y. Zhu, T. Zhang, S. Liu, H. Wang, and K. Xu, "Polycc: Poly-algorithmic congestion control," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 1129–1131.

[28] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," Network Working Group, RFC 2988, November 2000. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2988

[29] "tcconfig," https://tcconfig.readthedocs.io/en/latest/, [Online; accessed May 15, 2024].

[30] Wikipedia contributors, "Box plot," https://en.wikipedia.org/wiki/Box_plot, [Online; accessed May 15, 2024].

[31] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling bufferbloat in 3g/4g networks," in *Proceedings of the 2012 Internet Measurement Conference*, 2012, pp. 329–342.

[32] "iperf - the ultimate speed test tool for tcp, udp and sctp," https://iperf.fr/iperf-download.php, [Online; accessed May 15, 2024].