Deadline and Priority-aware Congestion Control for Delay-sensitive Multimedia Streaming

Chao Zhou[1], Wenjun Wu[1], Dan Yang[1], Tianchi Huang[2,1], Liang Guo[1], Bing Yu[1]
[1]Kuaishou, Beijing, China
[2]Dept. of Computer Science and Technology, Tsinghua University
{zhouchao,wuwenjun05,yangdan07,huangtianchi03,guoliang,yubing}@kuaishou.com

ABSTRACT

Most applications of interactive multimedia require the data to arrive within the specific acceptable end-to-end latency (i.e., meeting deadline). To avoid efforts being wasted, the content must reach the destination before the deadline. In our work, we propose DAP (Deadline And Priority-aware congestion control) to achieve high throughput within acceptable end-to-end latency, especially to send high-priority packets while meeting deadline requirements. DAP is mainly composed of two modules: i) the *scheduler* decides which packet should be sent at first w.r.t the reward function with fully considering the packets' priority, deadline, and current network conditions. ii) the *deadline-sensitive congestion control module* transmits packets with high efficiency while guaranteeing the end-to-end latency. Specifically, we propose an improved packet-pair scheme to adjust the best congestion window (corresponding to the Bandwidth-Delay Product) and to update the instant sending rate by current queue length. Experimental results demonstrate the significant performance of our scheme and DAP ranks first in both the training phase and final phase of the *ACM MM 2021 Grand Challenge: Meet Deadline Requirements*.

CCS CONCEPTS

• **Information systems** → *Multimedia streaming*.

KEYWORDS

Congestion Control, Delay-sensitive Multimedia Streaming

ACM Reference Format:
Chao Zhou, Wenjun Wu, Dan Yang, Tianchi Huang, Liang Guo, Bing Yu. 2021. Deadline and Priority-aware Congestion Control for Delay-sensitive Multimedia Streaming. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3474085.3479208

1 INTRODUCTION

Recent years have seen explosive growth in the requirement of live or interactive multimedia applications, such as VR, AR, video conferencing apps, etc [7]. Different from recent conventional multimedia services (e.g., video-on-demand (VOD) streaming [1, 2]), the quality of experience (QoE) of such applications is greatly influenced by the contents marked by the different priority and deadline requirements. For example, in the tile-based 360 video streaming [18], each tile's deadline is its playback time minus local processing time. And the priority of each tile can be set according to the user's viewpoint. In such scenarios, the priority of the tiles in the viewpoint is higher than the others [17]. Users will feel satisfied if the tiles with higher priority are successfully transmitted on time. Otherwise, missing the deadlines of tiles in the viewpoint will severely degrade the users' experience [16].

In this study, we model the aforementioned tasks as the *delay-sensitive multimedia streaming*, in which the data are often transmitted in blocks (e.g., chunks in live video streaming, frames in video conferencing) [14, 16]. A block, consisting of several packets, has its priorities and deadline requirements. It's quite challenging for a congestion control algorithm to ensure users' QoE for delay-sensitive multimedia streaming since the algorithm has to **i)** decide which data block to send first, aiming to achieve a higher priority score on the blocks that satisfying the deadline requirements, and **ii)** consider how to adapt the current network bandwidth with high throughput and low end-to-end delay (§2).

Nevertheless, off-the-shelf congestion control algorithms (CCAs) suffer from several key issues in delay-sensitive multimedia streaming. On the one hand, recent popular CCAs, such as CUBIC [9] and BBR [5], increase the data in *flight* (data sent but not yet acknowledged) to detect the bottleneck size of the current network. However, such operations may cause high queueing delay or even congestion, and eventually, the blocks will possibly miss the deadline. Meanwhile, delay-based congestion controls (e.g., Copa [4]) and real-time rate control algorithms (e.g., GCC [6]) send data packets conservatively in pursuit of end-to-end target latency, resulting in low bandwidth utilization. On the other hand, existing schemes lack support for the deadline requirement, and very few control approaches can customize priority scheduling in their services [16]. Salsify [8] allows the sender to send frames with different frame types, while it requires a specific video encoder. Rebera [13] dynamically sends or drops the encoded frames for avoiding congestion events. However, it seldom considers the heterogeneous deadline requirements on video and audio frames. To that end, how to design a novel congestion control scheme that helps tame the complexity of delay-sensitive transmission tasks? (§3.1)

In this paper, we propose *Deadline And Priority-aware congestion control (DAP)*, a deadline and priority aware approach for delay-sensitive multimedia streaming (§3). DAP is mainly composed of two modules, i.e., the scheduler and the deadline-sensitive congestion control module. Technically, given a list of blocks with different priorities and deadline requirements, the scheduler leverages

MM '21, October 20–24, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3479208

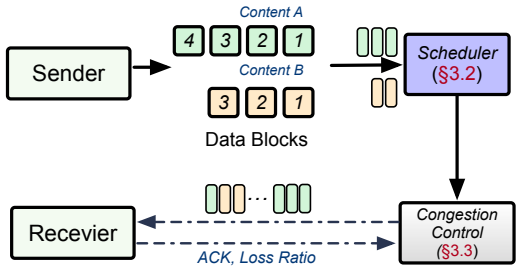


Figure 1: The System Overview of Delay-sensitive Multimedia Streaming.

a reward function to help determine which packet of which block should be sent at first (§3.2). Here the reward function is determined by several critical features, including packets' priority, deadline, and current network status. The proposed congestion control module can be viewed as the renascences with modified packet-pair methodologies [12], as it estimates the current Bandwidth-Delay Product (BDP) by sending the appropriate packet-pairs and controls the queue length by adjusting the congestion window size (§3.3). Subsequently, from the long-term perspective, the goal of DAP is to pick the high-priority packets of blocks within the acceptable deadline as much as possible, and it sends the packets with high throughput and low end-to-end delay.

We evaluate DAP over different scenarios, including various deadline-requirements, background flows, and network traces (§4). Using trace-driven emulation, we demonstrate that the DAP consistently outperforms state-of-the-art congestion control schemes on the overall QoE by 30.31%-43.5% (§4.2). Through further ablation studies, we show that both the two modules of DAP help improve the performance from different perspectives. Moreover, we give feasible suggestions for enhancing other congestion control approaches (§4.3). Finally, DAP ranks first in the both training phase and final phase of ACM MM 2021 Grand Challenge: *Meet Deadline Requirements* [3], which shows the advantages of our scheme.

2 SYSTEM OVERVIEW

The typical system workflow of delay-sensitive multimedia streaming is illustrated in Figure 1. As shown, the system consists of a sender and a receive. The sender leverages a *scheduler* and a *congestion control module* to deliver blocks, in which the block has its priority and deadline requirements. Once a block is sent to the transport layer by the sender side, it will be integrated into a scheduled queue. Then the scheduler picks the proper packet to send, where the packets are segmented from blocks. The congestion control module sends the packets and collects useful information such as ACK and packet loss for avoiding congestion events. Finally, the network states (eg., receive rate, RTT, and loss ratio) are fed back to the scheduler for deciding for the next time slot.

3 DAP MECHANISM

We propose DAP (Deadline And Priority-aware congestion control) to achieve high throughput while meeting acceptable end-to-end latency. The aim is to maximize the quality of experience (QoE) of all blocks (§3.1). Our proposed scheme includes two parts: i) a scheduler used to decide which packet should be sent at first according to our designed reward function, in which the packets' priority, deadline, and current network conditions are considered (§3.2). ii) a deadline-sensitive congestion control algorithm used to transmit packets in high efficiency while preserving the end-to-end latency (§3.3).

3.1 Problem formulation

Here we refer to the QoE model specified by ACM MM 2021 Grand Challenge: *Meet Deadline Requirements* [3]. For block n, we denote its priority as P_n. Therefore, when block n arrives before its deadline, the QoE improvement can be expressed as $\phi(P_n)$. And the block with high priority, which is significant to QoE, corresponds to high $\phi(P_n)$. What's more, if block n misses its deadline, it will deteriorate QoE which is expressed as $\psi(P_n)$ with discount factor μ. Thus, the QoE of block n can be written as

$$QoE_n = \phi(P_n) * meet_n - \mu * \psi(P_n) * (1 - meet_n), \tag{1}$$

where $meet_n$ is a binary-state value and indicates whether block n arrives before its deadline (i.e., $meet_n = 1$ means block n met deadline, $meet_n = 0$ means block n missed deadline).

The deadline of block n is denoted as T_n (the block should be transmitted successfully before T_n, otherwise it will be dropped even it is received). In practice, each block is packed up into one or more packets before transmission. We assume block n is consisted of W_n packets, and each packet is arrived at time $t_i^n, \forall 1 \le i \le W_n$. Then we have

$$meet_n = \begin{cases} 1, & \text{if } t_i^n \le T_n, \ \forall 1 \le i \le W_n \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

At last, the transmission problem for the deadline requirements can be formulated into a maximization problem as $\max \sum_{n=1}^{N} QoE_n$, subject to Eq. 1 and Eq. 2, and N is the total number of blocks.

To better obtain QoE (Eq. 1), we attempt to tackle the following challenges:*i) which packet or block should be sent at each certain moment? ii) how to determine the sending rate to balance the delay and throughput?*

3.2 Scheduler

The scheduler aims to answer the first question of the above section: *which packet/block should be sent at each certain moment?* The intuition for designing a scheduler is that the block with the highest priority or least deadline should be sent at first. However, only considering one factor for scheduling is not suitable. For example, a block with the highest priority may have a very large deadline, and there is no need to send it immediately especially there are some urgent blocks. Moreover, the block size is required to schedule as a block with a big size occupies more network resources for successful transmission.

In our proposed scheduler, we propose a reward function for each block, in which the blocks' priority, size, deadline, and current network conditions are considered. We pick the block with the highest reward. Meanwhile, instead of scheduling the whole block for sending, once an ACK packet arrives, we update the rewards for all the blocks, and then only fetch the first packet of the block with the highest reward for alternative sending.

For block n, it is consisted of W_n packets. The size of i^{th} packet is $s_i^n, \forall 1 \le i \le W_n$ (in bytes) with an indicator x_i^n to denote whether

packet i has been received, i.e., $x_i^n = 1$ if packet i has arrived before deadline, otherwise $x_i^n = 0$. Besides, at a certain moment t, we assume the network bandwidth is $bw(t)$ in $kbps$ and packet loss ratio is $plr(t)$. To guarantee the packet can be transmitted successfully, re-transmission is adopted. Specifically, $i_{th}(\forall 1 \le i \le W_n)$ packet of block n will be sent (including re-transmission) k_i^n times to guarantee the equivalent packet loss ratio (the packet is still lost even under re-transmission) is no more than ε (typically, $\varepsilon = 0.01$) with

$$k_i^n = \lceil log^{\varepsilon}_{plr(t)} \rceil \tag{3}$$

Thus, to make sure that block n can arrive successfully, the remaining size (including re-transmission) that needs to be transmitted is

$$S_n = \sum_{i=1}^{W_n} s_i^n * (1 - x_i^n) * k_i^n \tag{4}$$

and it will consume time duration d_n in seconds with

$$d_n = \frac{S_n * 8}{bw(t) * 1000}. \tag{5}$$

Note that we have adopted an equivalent packet loss ratio, rather than block loss ratio, to guarantee the blocks' transmission reliability for simplicity.

Then, we define the reward for block n as

$$R_n = \frac{\phi(P_n)}{S_n} * f(T_n - t_c, d_n) \tag{6}$$

where $\frac{\phi(P_n)}{S_n}$ is the normalized QoE improvement, t_c is the current time and $T_n - t_c$ denotes the available time for transmitting block n before its deadline. $f(T_n - t_c, d_n)$ is the success ratio for block n arriving before its deadline and it is given as

$$f(T_n - t_c, d_n) = \begin{cases} \min(1, \eta * \frac{T_n - t_c}{d_n}), T_n - t_c \ge 0 \\ 0, T_n - t_c < 0 \end{cases} \tag{7}$$

where η is the discounted factor.

Now, whenever a packet can be sent according to our congestion control algorithm introduced in the next section, the rewards of all blocks are updated and the first packet of block n^* is fetched to sent with $n^* = \underset{n}{\text{argmax}} R_n$.

3.3 Deadline-sensitive congestion control

Then, we move to the second question: *how to determine the sending rate?* Excessive sending rate will cause high queuing delay even congestion, while too low sending rate means waste of resources, both will harm the experience of users.

When the in-flight is no more than the Bandwidth-Delay Product (BDP), no queuing delay occurs and the lowest end-to-end delay is achieved. In other words, too small in-flight will lead to low bandwidth utilization. Thus, in this work, we consider adjusting the congestion window (sending rate) to control the in-flight towards the estimated BDP.

At any certain time t, when a packet is acked or lost (by packet loss detection mechanism), the binary-state indicator $b(t)$ is updated that if $b(t) = 1$, it denotes the network is blocked and no packet will be sent, otherwise, at least $F(t)$ packets, named chunk, will be sent simultaneously. $F(t)$ is used to make sure that at least two packets are acknowledged in order (first sent, first acknowledged). Generally, $F(t)$ is set to 2 if no packet disorder happens during transmission, and it is increased (generally no larger than 6) according to the disorder level.

Therefore, for the chunk sent at time t, there are $F(t)$ packets with sending time $st_i(t)$ and acknowledgement time $at_i(t), \forall 1 \le i \le F(t)$. Since all the $F(t)$ packets are sent simultaneously, we have $st_i(t) \approx st_j(t), \forall 1 \le i, j \le F(t)$. Average transmission time for a single packet $\tau(t)$ can be approximately expressed by the time difference of acknowledgements:

$$\tau(t) = \frac{\sum\limits_{i,j} \frac{at_j(t) - at_i(t)}{j-i}}{\sum\limits_{i,j} \Gamma(i,j)}, \ \forall 1 \le i < j \le F(t) \ \& \ at_i(t) < at_j(t) \tag{8}$$

where disordered packets are ignored, and $\Gamma(i,j) = 1$ if $i < j$, otherwise $\Gamma(i,j) = 0$. Then, the best congestion window, under which the in-flight is equal to the BDP, is given as

$$cwnd^* = \frac{RTT_{min}(t)}{\tau(t)} \tag{9}$$

where $RTT_{min}(t)$ is the estimated minimal RTT at time t.

With the congestion window $cwnd^*$, the packets are sent by solving the following sub-problems: i) whether the network is congested? i.e., how to determine the value of $b(t)$ at time t? ii) how many packets should be sent simultaneously?

First, we use IFP to denote the in-flight packets and it is initialized to zero. Whenever a packet is sent, we plus one to IFP. Meanwhile, IFP minus one when a packet's acknowledgement arrives or it is detected lost. At time t, $b(t)$ is updated as

$$b(t) = \begin{cases} 1, \ if \ \ IFP >= cwnd^* \\ 0, \ else \end{cases} \tag{10}$$

And then, the number of packets should be sent is give:

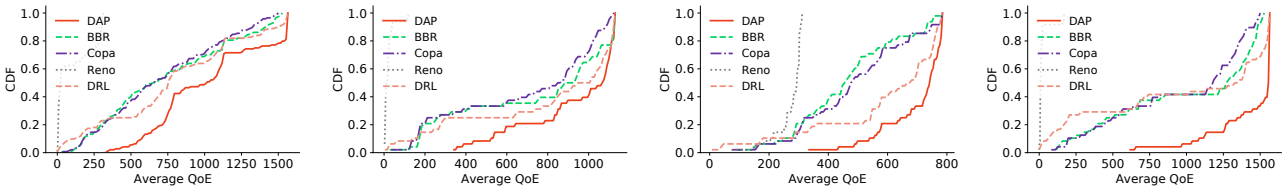
$$NP(t) = \begin{cases} max(F(t), cwnd^* - IFP), & if \ \ b(t) = 0 \\ 0, & if \ \ b(t) = 1 \end{cases} \tag{11}$$

From Eq. 11, we can see that $NP(t)$ packets can be sent simultaneously. Here $max(F(t), cwnd^* - IFP)$ is to make sure that at least two packets are acknowledged in order. Though it may send more packets when $F(t) >= cwnd^* - IFP$ at time t, it doesn't matter in the following round, fewer packets will be sent for compensation automatically. Besides, when $F(t) < cwnd^* - IFP$, pacing is adopted to avoid too big burst.

4 EVALUATION

4.1 Methodology

Testbed. We use a trace-driven simulator provided by ACM MM 2021 Grand Challenge [3] to test and optimize DAP. The simulator is written by Python, which can evaluate the solution via both no background flow and competition with background flow. The block datasets have two parts: block traces and background flows. The block traces divide the test case into 3 scenarios: 1) fixed deadline with three different priorities, 2) real-time multimedia (video and audio), and 3) dynamic priority and deadline requirement. The background flow has three different patterns: the web flow (e.g., Google Search), the video flow (e.g., YouTube), and the real-time



(a) All Scenarios (b) S. 1: fixed deadline, dynamic priority (c) S. 2: real-time multimedia (d) S. 3: dynamic deadline and priority

Figure 2: Comparing the performance of DAP with existing CCAs in three representative scenarios provided by [3]. Results are evaluated with QoE metrics described in Eq.1, plotted by CDF curves.

flow (e.g., Game Online Streaming). Furthermore, we adopt the network traces dataset provided by [3], which includes 118 network traces generated from 36 different patterns, bandwidth ranging from 0.1Mbps to 40Mbps. For each scenario, we sweep through all considered conditions to evaluate each scheme.

Baselines. We take recent representative CCAs as the baselines, including Reno [10], a typical loss-based approach; Copa [4], a delay-based heuristic, which computes the target sending rate by estimating minimum delay; and BBR [5], a hybrid approach that builds an explicit model via available bandwidth and RTT. Moreover, we also train a neural network model via the deep reinforcement learning (DRL) method PPO [15]. We use a similar state and action space with Aurora [11]. The policy is learned by maximizing the QoE objective function (Eq. 1).

4.2 DAP vs. Baselines

Trace-driven Results. Figure 2 plots the CDF curve of QoE across all scenarios respectively. Here we find that DAP outperforms existing baselines over all considered video scenarios, with the improvements on average QoE of 43.5% (Copa), 39.12% (BBR), and 30.31% (DRL). It makes sense that DAP leverages the deadline & priority-based scheduler and the deadline-aware congestion control, which can not only guarantee the overall miss rate but also reduce the end-to-end latency. With further analysis, we find that DAP improves 26.34% on scenario 1, 51.05% on scenario 2, and 42.97% on scenario 3 compared with BBR. The performance of BBR heavily depends on the scenario services. For example, in scenario 2 (real-time multimedia streaming scenario), BBR has to estimate the maximum bandwidth via filling the pipe, resulting in poor performance. The same conclusions can be found in the comparison results of Copa (i.e., 35.12% on scenario 1, 42.72% on scenario 2, and 49.93% on scenario 3). Copa is a delay-based scheme that performs well in real-time multimedia streaming but it fails to achieve acceptable performance in Scenario 1 and Scenario 3.

Final Competition Results. Moreover, DAP is well behaved in the unseen networks and requirements. MM Grand Challenge consists of a training phase and a final phase. Our team (Kwai2021) ranks first with an overall score of 5012.26 [3], which yields the effectiveness and generalization capacity of DAP.

4.3 Ablation Study

Comparison of different schedule strategies. Table 1 reports the overall QoE of DAP and baselines with different schedule policies, where the results are collected under all scenarios. As shown, we find that our proposed scheduler can significantly help various algorithms improve the performance, not only DAP but also

Table 1: Performance of using different schedule policies.

	Deadline-First	Priority-First	Proposed Scheduler
Reno	20.9 ± 26.92	269.11 ± 141.04	327.1 ± 131.31
Copa	684.11 ± 365.46	878.42 ± 260.46	890.08 ± 249.54
BBR	731.69 ± 385.07	892.29 ± 283.61	904.19 ± 270.14
DAP	753.22 ± 380.95	914.33 ± 254.43	**924.39 ± 244.78**

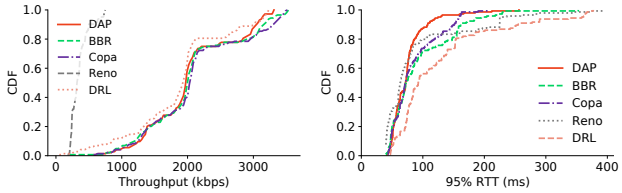


Figure 3: Comparison of different CCAs. We use our proposed scheduler for all algorithms.

other schemes. Surprisingly, one counter-intuitive fact is that the priority-first scheduler performs only 1%-21% lower than our proposed scheduler. We consider discussing it in future work.

Analysis of Congestion Control Algorithms. Figure 3 shows the CDF of the average throughput and 95% RTT. We can clearly see that although DAP doesn't provide the highest throughput (slightly lower than the best scheme Copa by 3.23%), it enormously reduces the end-to-end latency by 11.16%-40.29% compared with baselines. Such observations prove that transmitting streams with high throughput and low latency does impact the QoE. Moreover, as much as the DRL-based method obtains both low throughput and high latency, it also reaches acceptable QoE over all scenarios (see in Figure 2). One of the underlying reasons is that the DRL-based approach is trained via maximizing the QoE objective function rather than network metrics, and thus, it may learn another strategy that diverges from our actual demand. While such results also shed light on other possible ways to fulfill users' experience.

5 CONCLUSION

In this paper, we proposed a novel deadline and priority-aware congestion control approach, namely DAP. DAP was a delay-sensitive scheme that leverages a scheduler to pick the packet with high priority in deadline requirement and designed a novel deadline-sensitive congestion control algorithm to achieve high throughput and low latency. DAP ranked first in the both training and final phase of the ACM MM 2021 Grand Challenge. For future work, we attempt to deploy DAP in the real world and tap the potential of utilizing it in the cloud gaming scenario.

REFERENCES

[1] 2019. DASH Industry Forum | Catalyzing the adoption of MPEG-DASH. https://dashif.org/

[2] 2019. HTTP Live Streaming. https://developer.apple.com/streaming/.

[3] AITrans. 2021. ACM Multimedia 2021 Grand Challenge: Meet Deadline Requirements. https://www.aitrans.online/MMGC2021/

[4] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical delay-based congestion control for the internet. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 329–342.

[5] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue* 14, 5 (2016), 20–53.

[6] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–12.

[7] Cisco. 2017. Cisco Visual Networking Index: Forecast and Methodology, 2016-2021. https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[8] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 2018. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 267–282.

[9] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[10] Van Jacobson. 1988. Congestion avoidance and control. *ACM SIGCOMM computer communication review* 18, 4 (1988), 314–329.

[11] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A deep reinforcement learning perspective on internet congestion control. In *International Conference on Machine Learning*. PMLR, 3050–3059.

[12] Srinivasan Keshav. 1995. Packet-pair flow control. *IEEE/ACM transactions on Networking* (1995), 1–45.

[13] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. 2016. Real-time bandwidth prediction and rate adaptation for video calls over cellular networks. In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–11.

[14] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. 2017. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*. 183–196.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[16] Hang Shi, Yong Cui, Feng Qian, and Yuming Hu. 2019. Dtp: Deadline-aware transport protocol. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019*. 1–7.

[17] Chenglei Wu, Ruixiao Zhang, Zhi Wang, and Lifeng Sun. 2020. A Spherical Convolution Approach for Learning Long Term Viewport Prediction in 360 Immersive Video. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 14003–14040.

[18] Mengbai Xiao, Chao Zhou, Yao Liu, and Songqing Chen. 2017. Optile: Toward optimal tiling in 360-degree video streaming. In *Proceedings of the 25th ACM international conference on Multimedia*. 708–716.