The ACM Multimedia 2021 Meet Deadline Requirements Grand Challenge

Jie Zhang
Tsinghua University
zhangjie19@mails.tsinghua.edu.cn

Junjie Deng
Beijing University of Posts and Telecommunications
junjay.deng@bupt.edu.cn

Mowei Wang
Tsinghua University
wang.mowei@outlook.com

Yong Cui
Tsinghua University
cuiyong@tsinghua.edu.cn

Wei Tsang Ooi
National University of Singapore
ooiwt@comp.nus.edu.sg

Jiangchuan Liu
Simon Fraser University
jcliu@cs.sfu.ca

Xinyu Zhang
University of California San Diego
xyzhang@ucsd.edu

Kai Zheng
Huawei Technologies
kai.zheng@huawei.com

Yi Li
PowerInfo
tiger_li@263.net

ABSTRACT

Delay-sensitive multimedia streaming applications require their data to be delivered before a deadline to be useful. The data transmitted by these applications can usually be partitioned into blocks with different priorities, assigned based on the impact of a block on the Quality of Experience (QoE) if it misses its delivery deadline. Meet their deadline requirements is challenging due to the dynamics of the network and these applications' high demand on network resources. To encourage the research community to address this challenge, we organize the "Meet Deadline Requirements" Grand Challenge at ACM Multimedia 2021. This grand challenge provides a simulation platform onto which the participants can implement their block scheduler and bandwidth estimator and then benchmark against each other using a common set of application traces and network traces.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

deadline, delay-sensitive streaming, scheduler, bandwidth estimator

ACM Reference Format:
Jie Zhang, Junjie Deng, Mowei Wang, Yong Cui, Wei Tsang Ooi, Jiangchuan Liu, Xinyu Zhang, Kai Zheng, and Yi Li. 2021. The ACM Multimedia 2021 Meet Deadline Requirements Grand Challenge. 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/10.1145/3474085.3479229

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM '21, October 20–24, 2021, Virtual Event, China
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3479229

1 INTRODUCTION

Delay-sensitive multimedia streaming applications, such as live or interactive media applications, require a tight end-to-end latency (i.e., the time between the data is generated and the time it is rendered). For example, live video broadcasting requires end-to-end latency on the order of 5 seconds, to support interactions between broadcasters and viewers. Real-time communication (RTC) such as video conferencing has stricter latency constraints and the latency of these applications cannot exceed a few hundred milliseconds to enable natural interaction [3]. If the content cannot be delivered by the deadline (i.e., missing deadline), the delivery efforts may be wasted [21]. Such applications usually transmit data in application-level blocks (e.g., chunks in live video streaming, frames in video conferencing). When a block misses its deadlines, the users' Quality of Experience (QoE) degrades.

Delay-sensitive multimedia applications usually utilize methods such as bitrate adaptation to match the bitrate of a block to transmit to the network bandwidth. There are, however, obstacles in producing blocks that precisely matches the network condition and achieving the optimal results in terms of meeting a block's deadline. A typical example of such obstacle is that the encoder should not change the target bitrate frequently to avoid flickering effect [17]. Therefore, it is also important to design content delivery services that consider the deadline of each block and minimize blocks that miss deadline. Internet research community and standardization organizations are increasingly focusing on data transmission optimization of delay-sensitive applications. QUIC [15] address some problems such as head-of-line blocking and unnecessary multi-RTT handshake latency. The datagram extension to QUIC [18] allows unreliable delivery service to reduce the latency. However, none of them carefully consider the deadline requirements. Designing such deadline-aware delivery service faces three primary challenges: (i) the perception of network dynamic is often lagging and imprecise in Internet, (ii) when the rate of blocks produced by applications exceeds the network capacity, sequentially delivering the blocks will cause accumulative delay which may cause a series of deadlines missed, (iii) blocks may have conflicting attributes like deadline, priority and size that need to be considered together.

Block completion time is not only affected by the goodput but also by the connection latency. Traditional loss-based congestion control algorithms are unsuitable for delay-sensitive applications since their probe mechanism induces a time-varying stochastic delay component into the propagation time [7]. Many RTT-based approaches have been proposed to achieve low queuing delay since the innovative work by Jain [11]. Several methods of these, such as Vegas [5], have been widely used by operating systems and applications. Some other works use one-way delay as the key signal, such as LEDBAT (over UDP) [20]. GCC employs both delay-based and loss-based controllers to achieve extremely low latency for video conferencing [7]. BBR [6] aims to maximize the goodput with minimal queue delay by probing the bottleneck bandwidth and min RTT. Sprout [24] takes a stochastic approach to contain delays while maximizing the throughput. Remy [23] employs the a-priori knowledge of the network to train a machine to learn schemes. However, none of these solutions takes the deadline requirements into consideration. If we utilize deadline or other information about blocks, higher QoE can be achieved compared to those approaches which not aware of application data attributes.

To accelerate the optimization of new applications as well as improve the users' QoE of delay-sensitive multimedia, we hope to conduct a series of "Delay-sensitive Multimedia" challenges. At ACM Multimedia 2019, we organized Live Video Streaming Grand Challenge [25] that focuses on improving QoE in low-latency live video streaming, with the team from Communication University of China (CUC) [19] winning the 2nd prize among all grand challenges.

At ACM Multimedia 2021, we organize the "Meet Deadline Requirements" challenge. This competition aims to encourage the research community to join forces and improve the ability of the delivery service modules in multimedia applications to meet deadline requirements. We provide a simulation platform (Section 2), video datasets, network datasets (Section 3), and evaluation procedure using a QoE model (Section 4). In this challenge, 45 teams submitted their algorithms (summarized in Section 5). We have open-sourced the platform and datasets [9], hoping these resources can assist everyone in optimizing the QoE of delay-sensitive multimedia.

2 BLOCKS DELIVERY SIMULATOR

In this challenge, we focus on the delay-sensitive applications which send data as blocks. Typically, data delivery of these block-based applications works in the following way: the sender continuously generates blocks and adds them to a delivery queue to send them as soon as possible, expecting them to arrive at the receiver before their deadlines. Since the network is dynamic and has limited resources, it is necessary to decide which block to send first at each moment and how fast it can be sent.

2.1 Architecture

The simulator provided for this grand challenge simulates the workflow above. It is structured into three components: the Solution, the Environment, and the QoE model, as shown in Figure 1. The Environment is the main part of the simulator, driven by discrete events. Before the simulation, one or more Senders, Links, and Receivers will be created according to the Environment setting. Each Sender simulates a list of application traces while each Link is simulated based on a network trace. These traces will be used to simulate the data patterns of applications and the dynamics of the network condition. The simulation settings also need to specify a Solution for each sender. The Solution contains a Block Scheduler and a Bandwidth Estimator. The Bandwidth Estimator controls the sending rate of the flow, trying to maintain a small queue delay while making full use of available bandwidth. Each time the sender is about to send a packet, the Block Scheduler will decide which block should be sent. More details about the Solution component of the simulator can be found in Section 4. After each simulation, the simulator will evaluate the QoE of every pair of application peers. The QoE model used in our platform is described in Section 4.2.

Table 1: Observations from Simulator

Params	Params Description
time(s)	Physical time
block_id	ID of the block, globally unique
priority	Priority of the block
deadline(ms)	Deadline of the block
create_time	The time when the block was created
block_size(bytes)	Size of the block
miss_ddl	Whether the block missed deadline
packet_id	ID of the packet, globally unique
event_type	Packet loss or arrival
inflight	The number of inflight packets
payload(bytes)	Size of payload in the packet
packet_size(bytes)	Size of the packet
offset	The offset of the payload in the block
retrans	Whether this is a retransmission packet

Figure 2 illustrates the structure of a typical Environment in this challenge. There are usually several senders and receivers sharing a link. Blocks are generated based on the application traces. After the scheduling of the Block Scheduler and the Bandwidth Estimator, the data leaves the sender as packets and enters the Link. The total available bandwidth and minimal delay of the Link vary according to the network trace. If the data arrival rate exceeds the bandwidth, the packets waiting to be sent will be temporarily stored in a Drop Tail FIFO queue. When the last packet of a block arrives at the receiver, the completion time of this block can be calculated and then used for QoE calculations.

2.2 Simulator Implementation

Based on the structure described, we developed a discrete event simulator to provide a re-producible environment for experimenting with the Block Scheduler and the Bandwidth Estimator algorithms. The simulator exposes a list of observations from the environment, some of the observations are listed in Table 1). The algorithms in the Solution can decide the next action based on these observations.

We have open-sourced the simulator as a package [8]. The simulator package can be used alone for the simulation of delay-sensitive blocks transmission, or be used together with the tools and datasets provided in the challenge repository [9]. It contains integrated code of the event-based Engine, Links, Senders, Packets, etc. Part of Link and Sender code are modified from PCC-RL [12] proposed in [13].

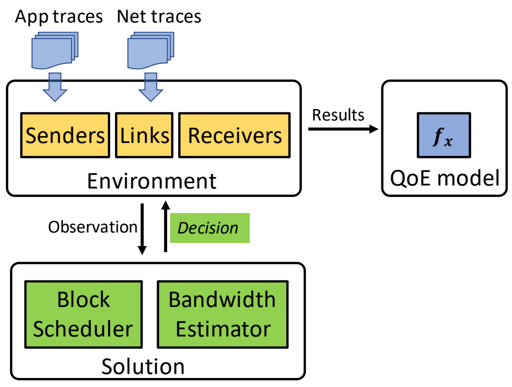

Figure 1: Overview of Blocks Delivery Simulator.

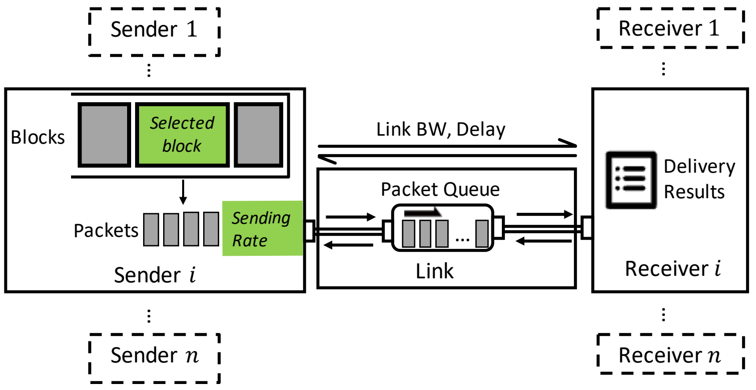

Figure 2: Structure of a typical Environment.

An earlier version of the simulator was tested in a competition called AITrans2 in October 2020. Around one hundred teams participated in this competition. The participants are asked to develop algorithms which will be tested through both the simulator and a live benchmark system. The relative performance ranking of submitted algorithms is mostly consistent when tested with the simulator and the live system, supporting the efficacy of our simulator.

3 DATASET DETAILS

The dataset used in this challenge consists of three parts: the block traces, the background traffic traces, and the network traces. Each application scenario contains one or more block traces, which can be combined with any network traces to simulate the process of using delay-sensitive applications in a certain network condition. Background traffic traces are used to simulate the complex wide area network environments and add more dynamics.

3.1 Block Traces

Figure 3 takes the RTC application as an example to illustrate the typical scenarios of various blocks. Generally, streamed data from RTC applications can be divided into three categories. The first category is the control signal, such as predicted bandwidth, target bit rate setting, etc. Control signals must arrive timely to ensure the stability of RTC application services. The second category is audio, that is, the user's voice data after noise elimination. The third category is video recorded by the camera. These three types of data have different priorities. Missing deadline events of control signal may cause severe decline of the QoE, thus these signals should have the highest priority. In most RTC applications, audio is more important than video.

The block traces indicate the attributes of the blocks. We select a variety of scenarios to evaluate the overall performance of the submitted algorithms. We generate these traces in two steps. Firstly, we record some actual attributes of data sent in the running of several applications, such as time and size of each video or audio frame of WebRTC recording from RecordRTC [14]. Secondly, we generate block traces according to the characteristics we want to evaluate. For instance, there is no need to make any changes to the raw files when the purpose is to simulate the original application; but we complicate the patterns to evaluate the performance for more complex applications that may appear in the future. For example, we add small high-priority blocks into WebRTC traces to imitate control signals in applications that need more interactivity. The finished traces are stored in text format, with each line corresponding to a block of an original or hypothetical application. Each line contains: (i) the timestamp, (ii) the size in bytes. Block information with different deadlines or priorities are stored in different files. Priority and deadline are indicated by name of the file.

3.2 Network Traces

To test whether the algorithms can obtain a good QoE in a wide range of network environments, this challenge evaluates submitted solutions over a large corpus of network traces. Each network trace is a text file containing multiple lines. Each line contains four floating-point numbers: the timestamp in seconds, the measured throughput in kbps, the loss rate and the one-way delay in milliseconds. The time interval between each network throughput sample is 1 sec. The network trace is generated in two ways: (i) We concatenated randomly selected traces from the collected real network traces in WiFi and 4G scenarios provided by PowerInfo. We only considered the original traces where the average throughput is less than 150 Mbps and the minimum throughput is above 8 Mbps. (ii) We leveraged the methods from Pensieve [16] to create a synthetic dataset. We design a dataset to cover a relatively broad set of network conditions, with average throughputs of 8 − 150 Mbps. We use RTTs between 0 ms and 100 ms in steps of 20 ms. These throughputs and RTTs cover the majority of global network conditions reported in [10] and [22]. The loss rate of each trace will be set to either 0.00 or 0.01.

3.3 Background Traffic Traces

To simulate the complex environments in the wide area network, we replay dynamical cross traffic by using background traffic traces. These traces are recorded from three different flows: live broadcast streaming from douyu.com; video on demand from the bilibili player, which based on ijkplayer [4], one of the most popular players in open-source community; browsing a variety of complex Web content on bilibili.com. The background traffic traces are optional

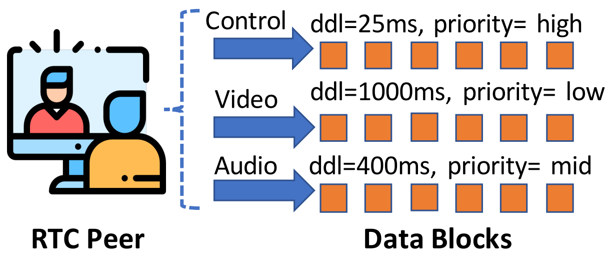

Figure 3: A typical framework of RTC application.

for each simulation, different from block traces and network traces, which are required by every simulation. In this challenge, we test the performance of the algorithm in an exclusive link and the performance when there is dynamic background traffic.

4 TASK AND EVALUATION

For the 2021 Multimedia Grand Challenge, we encourage the participants to take on the challenge of designing transmission mechanisms for delay-sensitive applications to improve quality of experience. We provide participants with an open-source platform, consist of the simulator, datasets and algorithms demos. A calculation method that converts block completion into QoE is provided to evaluate the score after simulation.

4.1 Task Description

The participants are asked to design block scheduling and bandwidth estimation algorithms to achieve high QoE when their algorithms are run in the simulator given various block traces and different categories of network traces, while there may be some background traffic traces used.

We select two parts that have great impacts on the meeting deadline of blocks: the simulator (i) determines the sending rate using the bandwidth estimator and (ii) selects the block to send at each moment using a block scheduler. **1. Bandwidth Estimator:** Bandwidth estimation has a great impact on the sending rate of blocks and the delay of path, which in turn affects if the blocks could meet their deadlines. The bandwidth estimation algorithm needs to decide the sending rate of each moment using input signals. **2. Block Scheduler:** Each block has properties such as deadline and priority. Block scheduler needs to balance these tradeoffs and decide which blocks should be sent at a certain moment.

These two algorithms interact with the simulator. It takes the observations as shown in Table 1 from the environment as inputs. According to these states, the algorithm decides on the pacing rate or congestion window and the id of the block to be sent for the next moment. Participants will design and develop these two transmission layer algorithms and implement them in the simulator. The goal is to optimize the QoE of multiple delay-sensitive application scenarios with the traces we provide. The grand challenge provides tools to run the simulation and visualize the results, helping participants to run and improve their solutions.

4.2 QoE Model

There exists significant variants in QoE for multimedia applications, such as video streaming [16]. The quality perceived by a user is generally improved when blocks arrive timely. Deadline misses may degrade the QoE. For example, missing the deadline of a video streaming chunk may cause rebuffering, which should be penalized [26]. For real-time Communication, if a block miss the deadline (e.g., several hundred milliseconds) for only one participant, there will be a severe negative impact on the QoE of the whole group [1]. Thus, we model the QoE as:

$$QoE = \sum_{n=1}^{N} \phi(P_n) \times meet_n - \mu \sum_{n=1}^{N} \psi(P_n) \times (1 - meet_n)$$

where $meet_n$ is a binary-state value and indicates whether the $block_n$ arrives before the deadline (i.e., $meet_n = 1$ means $block_n$ met its deadline, $meet_n = 0$ means $block_n$ missed its deadline); P_n denotes the priority of $block_n$; $\phi(P_n)$ expresses the QoE increase if $block_n$ arrives before its deadline. The block with high priority, which is significant to user's experience, corresponds to high $\phi(P_n)$; $\psi(P_n)$ penalizes the deadline missing event of $block_n$.

In this challenge, we consider a simplified choice of $\phi\,(P_n)\,, \psi(P_n)$ and μ :

$$\phi\,(P_n) = P_n, \quad \psi\,(P_n) = P_n, \quad \mu = 1$$

By using the above QoE model, we evaluated the participants' algorithms in a simulation given combinations of traces. For each block scenario, we randomly select tens of network traces from both collected and synthetic ones. Each selected case runs once in an exclusive link, and runs three times respectively under three background traces. Thus every solution is evaluated under hundreds of combinations.

5 OVERVIEW OF SUBMISSIONS

In this grand challenge, 45 teams submitted their solutions. All the top ten teams use heuristic algorithms. They all utilized both network signals and the properties of blocks to decide which block to send. For bandwidth estimator, most of their algorithms are variants of existing algorithms such as BBR [6] and Copa [2]. The most popular network signal is delay. Almost all top teams use delay in both two parts of solution. This indicates the significance of delay signals for optimization of deadline requirements.

The highest scored learning-based algorithm is ranked 13th place on the leader board. Two key factors may have the most effect on the relatively poor performance of learning-based algorithms in this competition: (i) The evaluation process combines multiple application scenarios, background traffic scenarios, and network scenarios. The low generalizability makes it is difficult for learning-based algorithms to obtain great overall performance; and (ii) The simulator supports packet-level actions, which enable quick response to new events. Thus the implicit prediction advantage from learning-based algorithms can bring little improvement.

ACKNOWLEDGMENTS

This work is funded in part by (i) National Key R&D Program of China (No.2018YFB1800303) and National Nature Science Foundation of China (No.61872211), (ii) the Singapore Ministry of Education Academic Research Fund Tier 1 (T1 251RES2038).

REFERENCES

[1] Doreid Ammar, Katrien De Moor, Min Xie, Markus Fiedler, and Poul Heegaard. 2016. Video QoE killer and performance statistics in WebRTC-based video communication. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*. IEEE, 429–436.

[2] 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 2018, Renton, WA, USA, April 9-11, 2018*, Sujata Banerjee and Srinivasan Seshan (Eds.). USENIX Association, 329–342. https://www.usenix.org/conference/nsdi18/presentation/arun

[3] Mario Baldi and Yoram Ofek. 2000. End-to-end delay analysis of videoconferencing over packet-switched networks. *IEEE/ACM Transactions On Networking* 8, 4 (2000), 479–492.

[4] bilibili. 2021. *ijkplayer.* Retrieved June 8, 2021 from https://github.com/bilibili/ijkplayer

[5] Lawrence S. Brakmo and Larry L. Peterson. 1995. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on selected Areas in communications* 13, 8 (1995), 1465–1480.

[6] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: congestion-based congestion control. *Commun. ACM* 60, 2 (2017), 58–66.

[7] 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.

[8] AItrans Competition. 2021. *The competition system for MMGC 2021.* Retrieved June 2, 2021 from https://pypi.org/project/simple-emulator/

[9] AItrans Competition. 2021. *Meet-Deadline-Requirements-Challenge.* Retrieved June 2, 2021 from https://github.com/AItransCompetition/Meet-Deadline-Requirements-Challenge

[10] Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, and Ramesh Govindan. 2016. An Internet-Wide Analysis of Traffic Policing. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 468–482. https://doi.org/10.1145/2934872.2934873

[11] Raj Jain. 1989. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM SIGCOMM Computer Communication Review* 19, 5 (1989), 56–71.

[12] Nathan Jay. 2019. *Reinforcement learning resources for PCC.* Retrieved May 10, 2021 from https://github.com/PCCproject/PCC-RL/tree/master

[13] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. 2019. A Deep Reinforcement Learning Perspective on Internet Congestion Control. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3050–3059. http://proceedings.mlr.press/v97/jay19a.html

[14] Muaz Khan. 2019. *WebRTC Demos, Experiments, Libraries, Examples.* Retrieved July 4, 2021 from https://www.webrtc-experiment.com/

[15] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 183–196. https://doi.org/10.1145/3098822.3098842

[16] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication.* 197–210.

[17] Pengpeng Ni, Ragnhild Eg, Alexander Eichhorn, Carsten Griwodz, and Pål Halvorsen. 2011. Flicker Effects in Adaptive Video Streaming to Handheld Devices. In *Proceedings of the 19th ACM International Conference on Multimedia* (Scottsdale, Arizona, USA) *(MM '11)*. Association for Computing Machinery, New York, NY, USA, 463–472. https://doi.org/10.1145/2072298.2072359

[18] Tommy Pauly, Eric Kinnear, and David Schinazi. 2018. An Unreliable Datagram Extension to QUIC. *Internet Engineering Task Force.(September 2018). draft-pauly-quicdatagram-00* (2018).

[19] Huan Peng, Yuan Zhang, Yongbei Yang, and Jinyao Yan. 2019. A Hybrid Control Scheme for Adaptive Live Streaming. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 2627–2631. https://doi.org/10.1145/3343031.3356049

[20] Sea Shalunov, Greg Hazel, Janardhan Iyengar, Mirja Kuehlewind, et al. 2012. Low extra delay background transport (LEDBAT). In *RFC 6817.*

[21] 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.

[22] Martino Trevisan, Danilo Giordano, Idilio Drago, Maurizio Matteo Munafò, and Marco Mellia. 2020. Five Years at the Edge: Watching Internet From the ISP Network. *IEEE/ACM Transactions on Networking* 28, 2 (2020), 561–574. https://doi.org/10.1109/TNET.2020.2967588

[23] Keith Winstein and Hari Balakrishnan. 2013. Tcp ex machina: Computer-generated congestion control. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 123–134.

[24] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13).* 459–471.

[25] Gang Yi, Dan Yang, Abdelhak Bentaleb, Weihua Li, Yi Li, Kai Zheng, Jiangchuan Liu, Wei Tsang Ooi, and Yong Cui. 2019. The ACM Multimedia 2019 Live Video Streaming Grand Challenge. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 2622–2626. https://doi.org/10.1145/3343031.3356083

[26] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication.* 325–338.