The ACM Multimedia 2019 Live Video Streaming Grand Challenge

Gang Yi
Tsinghua University
icarus.eagle@outlook.com

Dan Yang
Beijing University of Posts and Telecommunications
bupt-steven@foxmail.com

Abdelhak Bentaleb
National University of Singapore
bentaleb@comp.nus.edu.sg

Weihua Li
PowerInfo
powerinfo@263.net

Yi Li
PowerInfo
tiger_li@263.net

Kai Zheng
HuaWei
kai.zheng@huawei.com

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

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

Yong Cui*
Tsinghua University
cuiyong@tsinghua.edu.cn

ABSTRACT

Live video streaming delivery over Dynamic Adaptive Video Streaming (DASH) is challenging as it requires low end-to-end latency, is more prone to stall, and the receiver has to decide online which representation at which bitrate to download and whether to adjust the playback speed to control the latency. To encourage the research community to come together to address this challenge, we organize the Live Video Streaming Grand Challenge at ACM Multimedia 2019. This grand challenge provides a simulation platform onto which the participants can implement their adaptive bitrate (ABR) logic and latency control algorithm, and then benchmark against each other using a common set of video traces and network traces. The ABR algorithms are evaluated using a common Quality-of-Experience (QoE) model that accounts for playback bitrate, latency constraint, frame-skipping penalty, and rebuffering penalty.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

live video streaming, DASH, ABR, QoE, low latency

ACM Reference Format:
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 (MM '19), October 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3343031.3356083

*Corresponding author

MM '19, October 21–25, 2019, Nice, France

ACM ISBN 978-1-4503-6889-6/19/10…$15.00
https://doi.org/10.1145/3343031.3356083

1 INTRODUCTION

Recently, a new breed of video services that support interactive live video streaming has become tremendously prevalent. These video services allow users to broadcast live videos over the Internet, interact with their viewers and have many applications including journalism, online education, virtual reality (VR) [7, 8]. Compared to the video-on-demand (VoD) services, live video streaming requires a low end-to-end delay for real-time interaction between the broadcasters and the viewers while still maintaining low rebuffering ratio and high video quality. Achiving a good QoE [4] while ensuring low latency delivery has become an upcoming challenge.

To ensure high viewer's QoE, video streaming services use Dynamic Adaptive Video Streaming over HTTP (DASH) [2] which comprises an essential element at the player side, the ABR algorithm. ABR algorithms are leveraged to dynamically select the suitable bitrate level for future segments to be downloaded considering some heuristics such as download throughput and buffer occupancy level. Many state-of-the-art ABR algorithms have been proposed to deal with the dynamic network condition. These algorithms are either buffer-based (*e.g.,* BBA [5], BOLA [13], QUETRA [15]), rate-based (*e.g.,* FESTIVE[6]) or quality-based. The latter includes reinforcement learning-based algorithms (*e.g.,* Pensieve [10]) and model predictive control algorithms (*e.g.,* MPCDASH [16]). The existing video streaming system and ABR algorithms, however, inherently mismatch the properties of live video streaming scenario. DASH, which is the predominant form of video delivery today, cannot start playback until the entire video segment is fully downloaded, resulting in an end-to-end delay longer than a segment duration. Such delay, typically in the order of seconds, is unacceptable for live video delivery. Furthermore, live video is generated and delivered in real-time, and therefore, the ABR algorithm can only access a few seconds of video ahead of time and can utilize less information to make the optimal decisions.

To encourage the research community to come together and address the challenges of live video streaming, we organize a new Live Video Streaming Challenge[1] at ACM Multimedia 2019. This competition aims at improving the quality of live video delivery

[1] https://www.aitrans.online/MMGC/

where the main task is to design an ABR algorithm that adaptively decides on video bitrate and glass-to-glass latency control to attain the optimal QoE. We provide an open source simulator based on a frame-level live video streaming system similar to the MPEG Common Media Application Format (CMAF; ISO/IEC 23000-19) [1, 3] standard with DASH, a set of video traces, a set of network condition traces, and a common evaluation QoE metric. The participants use our system to implement and evaluate their ABR algorithms for live video delivery. We hope that the system will serve as a common tool for researchers to benchmark their algorithms with each other, and thus contribute towards reproducible research.

In this paper, we give a detailed description of our simulator platform (§ 2), present the datasets (§ 3) that are made available, describe the task of the grand challenge (§ 4), introduce our QoE model, discuss how each submission is evaluated (§ 5), and provide an overview of the submissions (§ 6).

2 LIVE VIDEO STREAMING SIMULATOR

The simulator platform used in this grand challenge is modeled after a commercial system deployed by *PowerInfo*[2]. It simulates the interaction between the video player and a content delivery network (CDN) server. Figure 1 shows the typical workflow of such a system: A streamer captures and generates a live video stream through a video source (*e.g.,* a mobile phone), and then uploads it to a transcoding server, which encodes the video into multiple representations, each with different bitrate levels and resolutions. Thereafter, each representation is pushed to a live origin server and then to the edge servers. The client issues pull requests to one of the edge CDN nodes, indicating which representation to download. The corresponding representation is then delivered to the client.

2.1 Reducing End-to-End Latency

Lowering the end-to-end latency is the most urgent requirement in live streaming scenarios. To reduce the latency, unlike the segment-level pull-based DASH, our simulator adopts *push-based frame-level delivery*, where the CDN node pushes video frames to the client for playback, as soon as the frames of the requested representation arrive at the CDN node. This is similar to CMAF [1] standard with DASH. Thus the latency of requesting each frame (*e.g.,* half of RTT) and waiting the entire segment downloaded for playback can be eliminated. The ABR algorithm makes the bitrate and delay control decision periodically according to the current state (e.g., measured throughput and buffer occupancy).

Our simulator further provides two knobs to control the latency caused by the playout buffers: by adjusting the playback speed and by skipping frames. The machanism to control the playout speed [14] can be controlled by a triple (S, T, Q), where T is the target buffer level; S and Q are two buffer thresholds. When the buffer level is between S and Q, the video is played back at normal speed (1x); if the buffer is below S or above Q, the playout slows down to r_{slow} times or speed up to r_{fast} times, respectively. In our system, based on the experience of PowerInfo, we set $S = 0.5T$, $Q = 2T$, $r_{slow} = 0.95$, and $r_{fast} = 1.05$. The viewer usually cannot notice the change in playout speed in this range.

[2]http://www.powerinfo.net/

Table 1: Observations from Simulator

Params	Params Description
time (s)	Physical time
time_interval (s)	Duration in this cycle
send_data_size	The data size downloaded in this cycle
frame_time_len (s)	The time length of the frame currently
rebuf (s)	The rebuffering time of this cycle
buffer_size (s)	The buffer size in seconds
play_time_len (s)	Video playout duration in this cycle
skip_time (s)	The time length of skipped frames
end_delay (s)	Current end-to-end delay
cdn_newest_id	The newest frame id in the CDN
download_id	Current downloaded frame id
decision_flag	GoP boundary flag or I-frame flag
buffer_flag	Whether the player is buffering
end_of_video	Whether the video playback is over

The frame skipping mechanism works as follows: Under normal circumstances, the client downloads the video frames in order. When the end-to-end delay exceeds a predefined latency threshold, the client can skip frames and request to download the next I-frame, instead of the next frame.

2.2 Simulator

Based on the streaming model described, we developed a discrete event simulator to provide a re-producible environment for experimenting with ABR and latency control algorithms. Figure 2 shows the simulated interactions between the video player and the CDN server in the simulator, which includes the CDN push, the client downloads, and the video playback. The operation environment is abstracted out — the simulator reads in a video trace and a network trace to simulator the dynamics of the video source and the last-mile network condition. The simulator is also written so that the ABR and latency control algoirithms are encapsulated, and so different implementation can be easily plugged into the simulator. The simulator exposes a list of observations from the environment (listed in Table 1). The ABR and latency control algorithms can decide the next action based on these observations.

An earlier version of the simulator was tested in an ABR algorithm competition called AITrans[3] in December 2018. Around 138 teams from China, USA and Japan have participated in this competition. The set of teams come from both academia and industry including Tsinghua, UCLA, and Alibaba. The participants are asked to develop an ABR algorithm which will be tested through both the simulator and a live video streaming system. Then the performance of the ABR algorithms is evaluated by a QoE model. The relative performance ranking of submitted ABR algorithms is mostly consistent when tested with the simulator and the live system, which supports the efficacy of our simulator.

[3]https://www.aitrans.online/

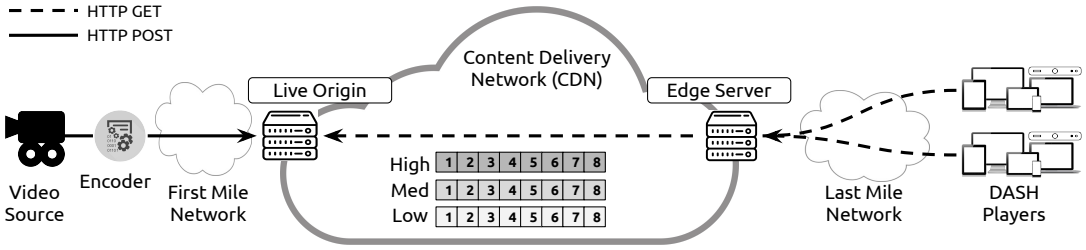

Figure 1: Universal framework of live streaming.

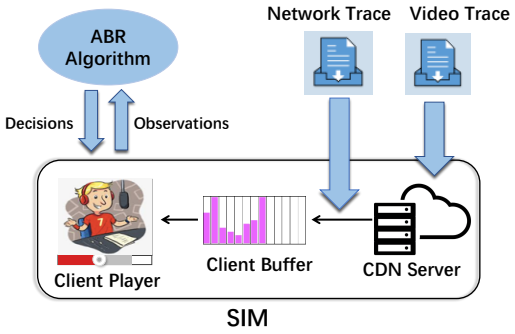

Figure 2: Overview of live video streaming simulator.

3 DATASET DETAILS

The dataset used in this challenge consists of two parts: the video traces and the network traces. Each video trace and network trace can be combined in the simulator to simulate the process of watching live video in a certain network condition.

3.1 Video Traces

The video traces contain frame-level traces of video sequences encoded with IPPP frame structure. There are six video sequences consisting of gaming, sports, and live shows scenarios. Figure 3 shows sample screenshots from these sequences. The video traces are generated by re-encoding a high quality source video (at 4 Mbps) using a H.264/MPEG-4 codec into four representations, at bitrate levels {500, 850, 1200, 1850} kbps and resolutions {240, 360, 480, 720}p. The traces are stored in text format, with each line corresponding to a frame in the video representation. Each line contains: (i) the timestamp of the video frame, (ii) the size of the video frame in bits, and (iii) a 1-or-0 flag indicating if the frame is an I-frame (if 1) or a P-frame (if 0).

3.2 Network Traces

The network trace records the network condition between a CDN server and the client. To test whether the ABR algorithm can obtain good QoE in a wide range of network environments, we provide a large-scale network dataset which covers a relatively broad set of network conditions. 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*.

Table 2: Three categories of network traces

Category	Range of average throughput
Strong Network	[2.0, 4.0] (Mbps)
Medium Network	[1.0, 2.0] (Mbps)
Weak Network	[0.2, 1.0] (Mbps)

We only considered original traces where the average throughput is less than 4 Mbps and the minimum throughput is above 0.2 Mbps. (ii) We leveraged the methods from Pensieve [10] to create a synthetic dataset. We design a dataset to cover a relatively broad set of network conditions, with average throughputs of 0.2 – 4 Mbps.

As shown in Table 2, all of the network traces are divided into three categories: strong network, medium network, and weak network according to the average throughput. Finally, three kinds of network traces are randomly mixed to generate oscillating network traces. Each network trace is a text file containing multiple lines. Each line contains two floating-point numbers: the timestamp in seconds and the measured throughput in kbps. The time interval between each network throughput sample is 0.5 secs.

4 TASK DESCRIPTION

The task in our competition aims at obtaining good QoE while maintaining the low latency. The participants are asked to design an ABR and a latency control algorithm to achieve high QoE when their algorithms are run in the simulator given various video traces and different categories of network traces. Specifically, The algorithm interacts 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 bitrate for the next segment, the value of target buffer which controls the adaptive playout mechanism and the latency threshold which controls the frame skipping mechanism. The decision space can be shown in Table 3. Note that the decision can only take effect when it comes to the GoP (group of pictures) boundary.

5 QOE EVALUATION

To evaluate the given ABR and latency control algorithms, our simulator records the frame rate, rebuffering time, end-to-end delay, and bitrate switches while simulating the live streaming and playback. These data are then fit into a QoE model, which we constructed based on the QoE model by Mao et al. [10]. We introduce two additional terms into the QoE model to consider the needs of live video

(a) Gaming. (b) Sports. (c) Live Show.

Figure 3: Three categories of video traces.

Table 3: Decision Space for Algorithm

Decision	Value Range
Bitrate	{500,850,1200,1850} kbps
Target Buffer	{0,1}
Latency Threshould	[0,∞]

streaming: the latency penalty, and the frame skipping penalty. The new QoE model can be expressed as:

$$QoE = \sum_{n=1}^{N}\sum_{m=1}^{M}(\bar{\beta}R_{n,m} - \gamma T_{n,m} - \delta L_{n,m} - \theta S_{n,m}) - \sum_{n=1}^{N-1}\bar{\alpha}|R_{n+1} - R_n|$$

where N is the number of GoPs; M is the number of frames in each GoP; $R_{n,m}$,$T_{n,m}$, $L_{n,m}$ and $S_{n,m}$ represent the bitrate, rebuffering time, latency and frame-skipping time of frame m in GoP n, respectively. The coefficients $\bar{\beta}, \gamma, \delta, \theta$, and $\bar{\alpha}$ can be set according to the application scenario and user preference.

To tune the QoE model, we ran multiple experiments to test the performance of simple ABR algorithms. These experiments lead us to set the coefficients as follows for the grand challenge: (i) The weight of bitrate, $\bar{\beta}$, is set to the playout time of this frame; (ii) The weight of rebuffering time, γ, is set to be the max bitrate level (1850 kbps in this challenge); (iii) To encourage the participants to adaptively utilize our latency control mechanisms and accomplish lower latency, we set different degrees of punishment towards different latency. In this challenge, if the end-to-end delay is lower than 1 sec, the weight of the latency penalty, δ is 0.005. Otherwise, δ is 0.01; (iv) The weight of frame-skipping time, θ is set to the lowest bitrate level. Thus, the penalty of frame-skipping is lower than rebuffering; (v) To decrease the number of bitrate switch, we set a small penalty, the weight of bitrate switch, $\bar{\alpha}$ to 0.02.

In this challenge, we evaluated the participants' algorithms in simulation given the all possible 12 combinations between the three video trace categories and four types of network traces by using the above QoE model. For each network type, we record the average QoE scores obtained by each algorithm under three video traces and twenty network traces of the corresponding type. Therefore, there are four ranking lists, each corresponds to a type of network trace.

6 OVERVIEW OF SUBMISSIONS

A total of 59 teams from China, USA, Canada, and Japan participated in our challenge. Most of the submitted algorithms are variants of existing ABR algorithms. Their modifications for live streaming indicate that current algorithms cannot be directly applied in the live scenario. For example, the buffer-based algorithms (*e.g.,* BBA [5]) lack sufficient scheduling space due to the small playback buffer caused by the low-latency constraint. The quality-based algorithms (*e.g.,* MPC [16] and Pensive [10]) need the future segment information as decision inputs.

Nine of the top ten teams submitted algorithms that use reinforcement learning. After analyzing all the ABR algorithms and their rankings, we found some common insights that are helpful to enhance ABR algorithms for live streaming. First, reinforcement learning for continuous control (*e.g.,* DDPG [9]) performs better than reinforcement learning for discrete action control (*e.g.,* DQN [11], A3C [12]) because the value of latency threshold is continuous. Second, since in live streaming scenario one cannot look ahead to know the properties of the future video segments, being able to accurately predict future source video information (*e.g.,* the segment size) based on the video information available.

7 SUMMARY

Through the ACM Multimedia 2019 Live Video Streaming Grand Challenge, we have provided a simulation platform that allows researchers to benchmark ABR and latency control algorithms against each other. We hope that this effort allows the community to push the envelope of live video streaming algorithms. The simulation platform, along with the video and network traces, are available online, at https://github.com/NGnetLab/Live-Video-Streaming-Challenge.

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 (NSFC) No.61872211, (ii) the NExT++ project, supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@Singapore Funding Initiative, and (iii) the Singapore Ministry of Education Academic Research Fund Tier 2 under MOE's official grant number MOE2018-T2-1-103.

REFERENCES

[1] 2018. *Information technology – Multimedia application format (MPEG-A) – Part 19: Common media application format (CMAF) for segmented media.* Standard ISO/IEC 23000-19:2018. International Organization for Standardization and International Electrotechnical Commission, Geneva, CH. https://www.iso.org/standard/71975.html

[2] Akamai. 2016. dash.js. https://github.com/Dash-Industry-Forum/dash.js/

[3] Abdelhak Bentaleb, Christian Timmerer, Ali C. Begen, and Roger Zimmermann. 2019. Bandwidth Prediction in Low-latency Chunked Streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '19)*. ACM, New York, NY, USA, 7–13. https://doi.org/10.1145/3304112.3325611

[4] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, and Zhang Hui. 2011. Understanding the Impact of Video Quality on User Engagement. In *Acm Sigcomm Conference*.

[5] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2015. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *SIGCOMM* 44, 4 (2015), 187–198.

[6] J. Jiang, V. Sekar, and H. Zhang. 2014. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. *IEEE/ACM Transactions on Networking* 22, 1 (Feb 2014), 326–340. https://doi.org/10.1109/TNET.2013.2291681

[7] Yoon Jung Park, Yang Yoonsik, Hyocheol Ro, Jinwon Cha, Kyuri Kim, and Tack Don Han. 2018. ChildAR-bot: Educational Playing Projection-based AR Robot for Children. 1278–1282. https://doi.org/10.1145/3240508.3241362

[8] Teemu Kämäräinen, Matti Siekkinen, Jukka Eerikäinen, and Antti Ylä-Jääski. 2018. CloudVR: Cloud Accelerated Interactive Mobile Virtual Reality. 1181–1189. https://doi.org/10.1145/3240508.3240620

[9] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[10] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *SIGCOMM*. ACM, 197–210.

[11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. 1928–1937. https://arxiv.org/pdf/1602.01783.pdf

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[13] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2016. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. (01 2016).

[14] Eckehard Steinbach, Niko Farber, and Bernd Girod. 2001. Adaptive playout for low latency video streaming. In *Proceedings 2001 International Conference on Image Processing*, Vol. 1. IEEE, 962–965.

[15] Praveen Kumar Yadav, Arash Shafiei, and Wei Tsang Ooi. 2017. QUETRA: A Queuing Theory Approach to DASH Rate Adaptation. In *Proceedings of the 25th ACM International Conference on Multimedia (MM '17)*. ACM, New York, NY, USA, 1130–1138. https://doi.org/10.1145/3123266.3123390

[16] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *SIGCOMM*, Vol. 45. ACM, 325–338.