RDladder: Resolution-Duration Ladder for VBR-encoded Videos via Imitation Learning

Lianchen Jia[1], Chao Zhou[2,5], Tianchi Huang[1], Chaoyang Li[1], Lifeng Sun[1,3,4,5]
[1]Department of Computer Science and Technology, Tsinghua University [2]Beijing Kuaishou Technology Co., Ltd.
[3]BNRist, [4]Key Laboratory of Pervasive Computing (Tsinghua University), Ministry of Education, China
[5]*Corresponding Authors. {jlc21@mails.,sunlf@}tsinghua.edu.cn, zhouchao@kuaishou.com*

***Abstract*—With the rapid development of the streaming system, a large number of videos need to be transcoded to multiple copies according to the encoding ladder, which significantly increases the storage overhead than before. This scenario demonstrates the prosperity of streaming media but also presents new challenges about achieving the balance between better quality for users and less storage cost. In our work, we observe two significant points. The first one is selecting proper resolutions under certain network conditions can reduce storage costs while maintaining a great quality of experience. The second one is the segment duration is critical, especially in VBR-encoded videos. Considering these points, we propose RDladder, a resolution-duration ladder for VBR-encoded videos via imitation learning. We jointly optimize resolutions and duration using neural networks to determine the combination of these two metrics considering network capacity, video information, and storage cost. To get more faithful results, we use over 500 videos, encoded to over 2,000,000 chunks, and collect real-world network traces for more than 50 hours. We test RDladder in simulation, emulation, and real-world environments under various network conditions, and our method can achieve near-optimal performance. Furthermore, we discuss the influence between the RDladder and ABR algorithms and summarize some characteristics of RDladder.**

I. Introduction

Recent years have seen a rapid increase in the streaming network traffic, and online video streaming has become the prevalent way of video consumption, especially under the COVID-19 [1]. The most widely used technology for video streaming is HTTP Adaptive Streaming (HAS). On the server-side, raw videos are pre-chunked to fix duration, such as 4s or 2s recommended by [2], and pre-encoded different resolutions to several bitrates or quality levels. On the client-side, users use the adaptive bitrate algorithm (ABR) to dynamically adapt the video streaming quality based on the current network conditions to achieve a higher quality of experience (QoE).

The encoding ladder guides the server to encode the raw videos to different bitrates or quality levels and try to get the higher QoE using the proper setting. Previous works mainly consider the relationship between the resolution and the bitrate, but we find other factors also significantly affect video streaming. First, we find that not all resolutions are necessary. The users of the edge cluster seldom have the chance to watch the resolution whose bitrates are much lower or higher than the bandwidth. Therefore, choosing the appropriate combination of resolutions can reduce the overhead of storage in video servers, which is more critical in current prosperous streaming media. In addition, past works use recommended fixed video chunk duration, which is indeed an excellent duration setting after extensive experiments and analysis by many researchers. However, different duration have their own advantages and disadvantages, especially in VBR-encoded videos. VBR-encoded methods encode simple scenes with fewer bits and complex scenes with more bits [3]. Supposing the duration is short, such as 1s, additional overheads will be introduced, which causes the short segment duration will have more keyframes than the long segment duration in the same quality level. In addition, longer video segments can reduce the number of requests from the client, which is significant to overall download time in the scenario with high link latency and large throughput.

Previous works have also noticed part of these two critical factors in the video stream performance and overhead of the video servers, such as [4] [5]. However, it is hard to set the fixed rules to select proper resolutions and duration as the diversity of network and video content.

So in this context, we propose RDladder, a per-title resolution-duration ladder that jointly optimizes for resolutions and duration for VBR-encoded videos. RDladder leverages imitation learning to train the neural network (NN) to determine the combination of the resolutions and duration, which has the potential to automatically find the most suitable ladder and avoid too much manual engineering. We do not focus on replacing the traditional encoding ladder, but we argue that before we optimize the resolution-quality levels pairs, we should first consider what resolution-duration combinations we need. So we consider the transcoding process as the sequential process and select the proper resolution-duration pairs with network conditions, video information, and storage cost. We design an objective function including the QoE and the storage cost to balance the trade-off between the high video quality and less storage cost, and determine the end time via the objective function. We collect more than 500 videos and encoded them to over 2,000,000 chunks for RDladder to learn, and we use over 50 hours real-world network traces and use different bandwidth utilization to increase the diversity of network conditions.

To the best of our knowledge, our study is the first to explore the new design space of joint optimization of resolutions and duration in video transcoding. In general, we summarize the contributions as follows:

- We analyze two significant factors that affect streaming performance, resolutions and duration. Our analysis shows that selecting only partial proper resolutions and taking full use of the advantage of different duration can achieve great QoE with less storage cost.

- Based on our analysis, we design RDladder, using neural networks for joint optimization of resolutions and duration to get resolutions-duration ladders.
- We test RDladder in simulation, emulation, and real-world environments with various videos and network conditions. Our results show that RDladder can achieve near-optimal performance. Furthermore, we discuss the influence between RDladder and ABR algorithms and summarize some characteristics of the results of RDladder.

II. Related Work

A. ***Encoding ladder***

The purpose of the encoding ladder is to find the optimal encoding setting to provide the highest quality for the user under limitations such as the network bandwidth or storage cost.

The fixed encoding ladders are recommended by the platforms of the video, such as [6] [7] [8] to gain high video quality. However, these ladders can not make full use of the information of video and network. The complexity of the content is the aspect that gets the most attention. In the work [9], the authors hope to determine resolution-bitrate pairs that closely reflect the convex hull of R-D curves. The authors in [10] try to find the optimal selection of the encoding ladder based on the network setting and the client status. Recently, the cost of the storage is getting more and more attention, such as [4] which comprehensively considers the influence of video features, network information, and the storage cost to find a better per-chunk ladder. The work of [11] uses the information of player feedback behavior, and the work of [12] considers the influence of different codecs.

These works focus on optimizing the pair of the resolutions and the bitrates. However, with the development of streaming media, the number of videos that need transcoding has increased significantly. If the video servers use the recommended resolutions to transcode these massive amounts of videos, the computing and storage cost are hard to accept. So, in our work, we argue that before the traditional works finding the most suitable bitrate for every resolution, finding the optimal combination of resolutions and duration is more significant.

B. ***Duration in streaming system***

Previous works to make full use of different duration mainly focus on duration switching and variable duration.

It is the most straightforward idea to encode the video to different duration in multiple copies. The work [5] analyzes the data of Youtube by catching the data packet when the video plays. They find that Youtube switches the duration to reduce data waste. Some other works consider the method to switch the duration to improve the QoE, such as [13] (use mathematical method) and [14] (consider the buffer). However, these works have to store more video chunks, which is hard to accept by the video content provider.

Another aspect of the research of duration is the variable duration. The work [15] proposes to divide the video segments near the position of the I-frame, which can reduce the introduction of additional I-frame. Recent work [16] proposes a new method to produce the variable duration for the streaming system. They consider the network conditions and the ABR algorithm, and for each video, they brute-force search all combinations of each frame to gain the maximum of the QoE. These works use variable duration to reduce the size of video segments. However, they are not flexible enough compared with duration switching, which can not fully use the advantage of different duration.

For our work, we still consider using duration switching, which makes most of the benefits from the different duration, and our system does not need to store multiple copies, which avoids the shortcoming of previous works.

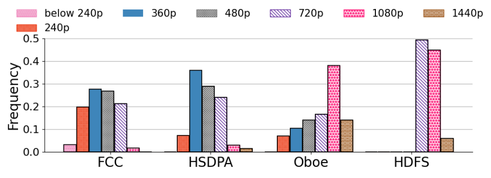
III. Motivation

A. ***Analysis about resolution selection***

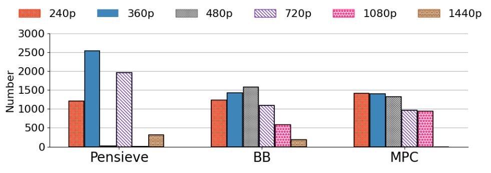
The selection is limited under the constraint of network conditions and ABR algorithms We plot the distribution of the highest resolution under average bandwidth using four network trace datasets, FCC [17], HSDPA [18] , Oboe [19] and HDFS [20] in Figure 1(a). We divide the average bandwidth into seven classes using the bitrate ladder of Pensieve [21] (a popular ABR algorithm). We can see that in FCC and HSDPA, the users seldom have a chance to watch 2k videos, and in HDFS, video encoded from 240p to 480p is probably not very useful for the streaming system. Through the above analysis, we can find that not all resolution candidates are necessary under certain network conditions.

Then we demonstrate that some resolutions are rarely selected by specific ABR algorithms. In Figure 1(b), we show the result in FCC, using three ABR algorithms (Pensieve, Buffer-base (BB) [22], MPC (In this work, we use Robust-MPC) [23]). Due to the algorithmic mechanism and certain network conditions, the selected probability of all resolutions is different. The 1440p is seldom chosen by these three ABR algorithms, and the 1080p is hardly chosen by Pensieve.

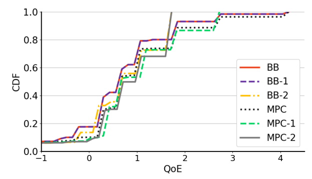
Properly removing some resolution candidates has an acceptable influence on the streaming performance Through the above analysis of the network conditions and the ABR algorithms, a straightforward idea is to remove the resolution with the least probability of being selected. We performed experiments on FCC to calculate their QoE using the function recommended by Pensieve as Figure-1(c). As Pensieve is trained with all resolution candidates, we use BB and MPC as the ABR algorithm in this experiment. As we have analyzed above, the 1440p and 1080p resolutions are the least likely to be selected by MPC and BB in FCC, so we remove 1440p called $MPC-1$ and $BB-1$, and remove 1440p and 1080p called $MPC-2$, $BB-2$ respectively. As we can see, the QoE of these six tracks is close in most cases, especially only



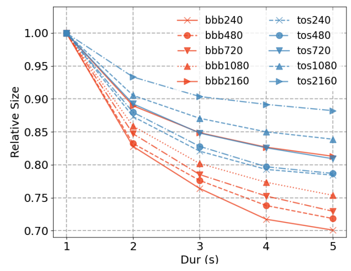
(a) The distribution of the highest resolution under average bandwidth



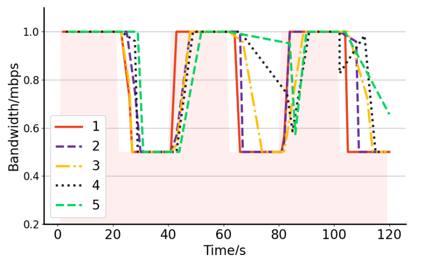
(b) The selection of ABR in FCC



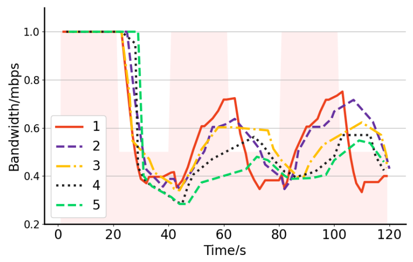
(c) The QoE removing some resolution levels



(d) The relationship between the duration and the size in VBR



(e) The prediction results using *past* prediction



(f) The prediction results using *robust* prediction

Fig. 1. Analysis about resolutions and duration

removing 1440p. So it is acceptable for QoE to remove some resolutions properly in most cases.

B. Analysis about the segment duration

The advantages of long duration Intra-frames(I-frames) are usually larger than predicted or bidirectionally predicted frames. In the HAS system, the chunk must start with I-frames, as this video chunk does not have any frames to reference. So, a short duration will introduce additional I-frames to guarantee this rule.

We choose two free videos commonly used in research, Big Buck Bunny (BBB) and Tears Of Steal (TOS), to encode. We scaled the original dimensions to 2160p, 1080p, 720p, 480p, 240p using bicubic and filter the frame rate to 24fps using the filter of FFmpeg[1]. We encode these videos with five different duration from 1s to 5s separately, i.e., $dur\epsilon\{1, 2, 3, 4, 5\}$, and we also use four different Constant Rate Factor (CRF), i.e., $crf\epsilon\{16, 22, 28, 34\}$ to get the average results in different video quality level. We plot the relative video size compared with the video with 1s duration in Figure-1(d). We can see a significant size reduction as the duration increases. The 240p and 480p reduced nearly 20%-30% size when using 5s duration,

So in this experiment, we can see the longer duration can reduce the video size in the same quality level, which gives the client more chance to get higher resolution videos.

The advantages of short duration Although the longer duration can reduce the video's size, the choice of the duration is challenging in the streaming system.

When meeting the fluctuating network, the short video can quickly adapt to the change in bandwidth. We synthesize a network trace with significant fluctuation to show the advantage of short duration. We consider the trace changed every 10s between 1mbps and 0.5mbps. We use two methods of bandwidth prediction to select the highest resolution below the predicted bandwidth at each step. One method is *past*, which considers past video chunk size and the past delay. i.e., $pre_{past} = \frac{past\_chunk\_size}{delay}$. The other method, called *robust*, which is the same as the prediction method of Robust-MPC (RMPC), considers the harmonic mean of the past five bandwidths and the max error of the past five bandwidth estimation, i.e., $pre_{robust} = \frac{harmonic\_mean}{1+max\_error}$. We show the results of prediction results compared with the actual results, the result of *past* in Figure-1(e) and the result of *robust* in Figure1(f). We can see that the short duration performs better than the long duration in both results. For *past*, a short-horizon and aggressive method, the long duration causes large rebuffering time such as the 60s-80s. And for *robust*, a relatively conservative method, can not make full use of the bandwidth. For example, in the 40s-60s, the bandwidth utilization is less than 50%.

C. The challenge in optimizing resolutions and duration

The analysis above has shown the importance of the proper resolutions and duration. However, it is not easy to find a simple way to fully use their advantages.

For the resolution selection, removing the least selected candidates is the simplest way. However, as we can see in Figure 1(c), such as MPC and $MPC-2$, we still can see clearly QoE drop when QoE beyond 2. If we want fewer candidates, the drop will be more obvious. So it is not easy to achieve the balance between the video quality and the storage cost.

As for the duration, we can see that different duration have their own advantages and disadvantages. The straightforward idea is to encode each video chunk to different duration and switch them based on the network. Although this method can surely improve the QoE in theory, it has two obvious disadvantages. The first one is the storage cost. Assuming we have m different duration, this method needs to cost m time storage overhead compared with the default one fixed duration, which is hard to accept by the video server. The

[1]https://ffmpeg.org/

second weakness is the decision complexity which significantly increases due to more choices. Just taking RMPC as an example, in the default setting with n candidates and looking 5-step ahead, the decision complexity is $O(n^5)$, but in this setting, the complexity becomes $O((mn)^5)$, which is hard to meet the requirements of real-time decision making in real deployments. As for the RL-based ABR algorithm, more action space will bring difficulties for the neural networks to learn a precise strategy.

IV. Methods

Motivated by the analysis of selecting appropriate resolutions and duration, we propose RDladder, using a neural network to select the proper combinations of resolutions and duration. In this section, we introduce three main ideas of RDladder, joint optimization of resolutions and duration, training NN via imitation learning, and an objective function to balance the trade-off between the QoE and the storage cost.

Joint optimization of resolutions and duration

To solve the large storage cost and high calculated complexity, we propose joint optimization of resolutions and duration, which means we select the most appropriate resolution-duration pairs from all candidates. In this way, we select the resolutions with specific duration and do not need to store every duration chunks for each resolution. If we select resolutions first and then find the suitable duration sets, we might miss some better results as the duration significantly influences the selection of resolution. For example, the 1080p with 5s may seldom be chosen under the restriction of bandwidth, but the 1080p with 1s may have more chance.

Traning RDladder via imitation learning

One-step or Sequential decision?

Suppose we have the resolution candidates n, the duration candidates m, then the number of these two metric combinations is mn. For each candidate, we have two choices, pick it or not, so the complexity becomes $O(2^{mn})$. To eliminate the enormous complexity, we consider the lifetime of selecting one of the candidates as a Markov Decision Process(MDP). So in every step, we choose one from the combinations that have not been chosen until the objective function determines the end signal. We can reduce the complexity from the $O(2^{mn})$ to $O(mnt)$(t is the step that is controlled by the objective function).

Furthermore, many video content providers do not transcode video to all resolutions but partial default settings. As the video becomes hot, more resolutions will be transcoded to improve QoE. So sequential decisions are closer to real-world situations.

Why imitation learning?

It is hard to find the appropriate setting for every state using specific expert rules, as the diversity of the network and the video contents. Moreover, as we have discussed above, each selection has its own pros and cons, and these features are dynamic with the network and the ABR strategies. So we propose using imitation learning [24] to automatically learn a policy to determine the combinations of resolutions and duration. Imitation learning allows the NN to interact with the environment to gain more data and use some supervised learning method to minimize the distance between the current strategy and the expert strategy, which avoids the design of reward function and brings higher data utilization compared with the reinforcement learning. In detail, at time t, the neural network interacts with the environments s using the policy π_t. Then the expert solver finds the optimal policy π_t^* at this time, and the NN calculates the loss $l(\pi_t, \pi_t^*)$. After that, the system chooses the action from the policy π_t and the environment s become s_{t+1}. We can reduce the gap between them and get a new policy $\hat{\pi}$ as Eq-1:

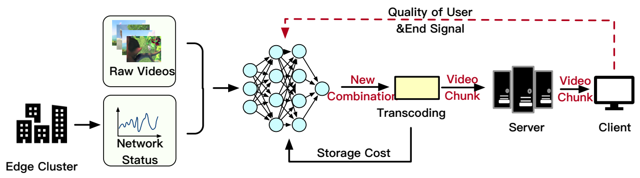


Fig. 2. RDladder System Overview

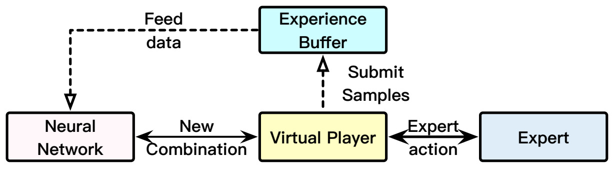


Fig. 3. Training Workflow Overview

$$\hat{\pi} = argminE_{s\sim d_{\pi}}[l(\pi_t, \pi_t^*)] \tag{1}$$

The design of objective function

In our system, we hope to balance the trade-off between the QoE and storage cost in the video server. It is hard to set the absolute weight, as we are not sure what additional size we desire to cost to achieve higher QoE in different videos and different network conditions. For example, if we consider that it is acceptable to get higher 5 QoE score with more than 50MB cost for a 200MB video, this cost might be unacceptable for a 20MB video.

So we choose the relative cost compared with the default setting as the score of the storage cost. We consider the video size with the default 4s duration and all resolutions as the base size, i.e. $base = \sum_{res \in all\_res} size_{dur=4}$. We use the weight w to balance the QoE and the additional storage cost. So, we set the objective function as :

$$sc = QoE + w(1 - size_{all}/base) \tag{2}$$

Furthermore, we use the objective function to decide whether to terminate this round of iteration. In training, for every state s, we choose one optimal combination using an expert solver from the optional candidates. If the optimal score does not improve, we consider that it is time to end. In the test, we compare the current score with the last score. If the score does not increase, we think this iteration should over.

V. Design of RDladder

In this section, we describe the proposed system RDladder in detail. RDladder's basic system is illustrated in Figure 2. The NN gets the raw video from the content provider, collects

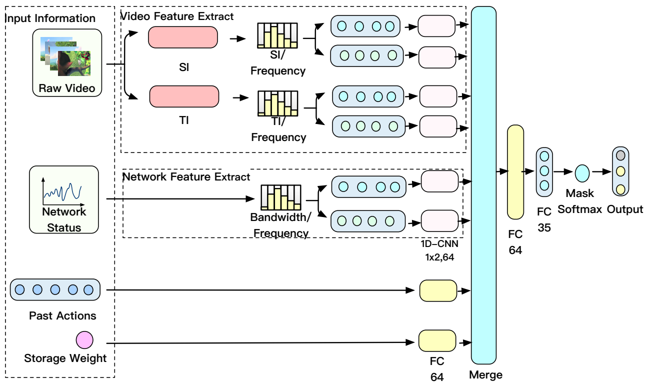


Fig. 4. Neural Network Architecture Overview

the network distribution from the edge cluster, and gets the storage limitations from the video server. Then the NN determines one new combination of the resolution and duration, which guide the video transcoding. These new video chunks are sent to the video server and played in the client. The client feedbacks the QoE and determines whether to end. Figure 3 shows the training workflow overview, which is mainly composed of a NN, a virtual player, an expert, and an experience buffer. In this section, we start by introducing the RDladder's NN module. Then we explain the virtual player's setting and the expert solver's design. Finally, we illustrate the basic training methodology.

A. *NN Architecture Overview*

RDladder's NN architecture is illustrated in Figure-4, and we now explain the compositions of NN, including its inputs, output, and network architecture.

Inputs

- **Video Information** Previous works have shown that the video information has a significant impact on video transcoding [10] [9] [4]. Different video contents have different scene complexity, which causes a large deviation in video size in the same quality level under VBR-encoded videos [3]. To determine the video's scene complexity, we use two commonly used metrics, Spatial information (SI) and Temporal information (TI), to quantify the complexity of spatial and temporal [25]. To have more representative information, we do not consider each specific value of SI and TI. We prefer the statistical information that reflects their distribution. So we use the histograms of SI and TI for a video to represent its video information. Specifically, we categorized these two metrics separately within a certain number of bins (in our setting, we set the number as 35). We use the information about the bins and the frequency of each bin on behalf of the metrics.
- **Network Information** Past works and the above analysis have shown the importance of the network information [26] [27]. Same as the video information, we do not care about the fine-grained information such as the precise bandwidth at a specific time, but we hope to know its general distribution. So, we still consider using the histogram to represent the network information similarly to the video information.
- **Past Action** As we consider the transcoding process as a sequence process, we still consider the past action as part of the feature. We consider the $P_t = \{a_1, a_2, ...a_t\}$ as the feature of past action, where a_t reflects the action at step t.
- **Storage Cost** For different video servers, the storage limit is different. We set the weight as $w\epsilon[0, 30]$ to balance the QoE and the limit of the storage cost.

Output As we model the transcoding process as the sequential process, the NN outputs one new combination of the resolution and duration from the optional candidates until the system receives the end signal from the expert solver.

Network Architecture As shown in Figure 4, we use six 1D-CNN layers with channels=64 to extract the feature from the video and the network information. We utilize two 64-dim fully connected layers to extract the valuable characteristics separately for the past action and the storage weight. Then we merge these vectors and feed them to a 64-dim full connected layer and a 35-dim full connected layer. Finally, we use masked-softmax to mask the previously selected action and get the probability of each action.

B. *Virtual player*

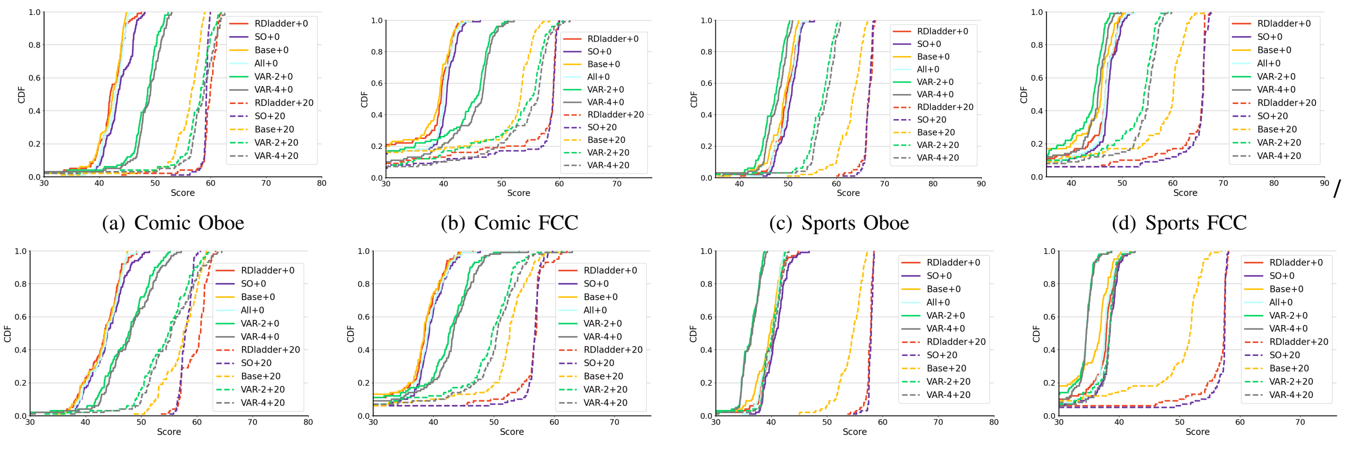
After the NN chooses a new combination, we add it to the chosen set. The virtual player uses these combinations to play videos virtually.

We refer to previous work [23] to calculate the download time and update the buffer as Eq-3. At virtual time t_k, we first calculate the download time for the chosen chunk R_k via $\frac{d_k(R_k)}{C_k}$, where $d_k(R_k)$ is the chunk size of R_k and the C_k is the average throughput from the network trace. Then we update the buffer B_{k+1} for chunk $k+1$ by the download time, the chunk duration L_k, and the waiting time δt_k such as Round-Trip-Time(RTT).

$$\begin{cases} t_{k+1} = t_k + \frac{d_k(R_k)}{C_k} + \delta t_k \\ C_k = \frac{1}{t_{k+1} - t_k - \delta t_k} \int_{t_k}^{t_{k+1} - \delta t_k} C_t d_t \\ B_{k+1} = \left[\left(B_k - \frac{d_k(R_k)}{C_k} \right)_+ + L_k - \delta t_k \right]_+ \\ B_1 = 0 \\ B_k \in [0, B_{max}], R_k \in R, \forall k = 1, ..., N \end{cases} \tag{3}$$

We play the video in the virtual player to get the QoE. Motivated by the previous works on the analysis of the QoE [21] [23], we can see the QoE is relative to the video quality, rebuffering time, and video smoothness. In our system, we use the classical video quality assessment VQA^{log} [28] [29], which is the non-linear relationship between SSIM [30] and the mean opinion score (MOS), to measure the video quality. Motivated by the recent work [29], we set the parameters as Eq-4 to calculate the video quality for each frame:

$$VQA^{log} = 75.5 - \frac{65.4}{1 + e^{37.37*(SSIM - 0.93)}} + 24.4SSIM \tag{4}$$



(a) Comic Oboe (b) Comic FCC (c) Sports Oboe (d) Sports FCC

(e) Game Oboe (f) Game FCC (g) Nature Oboe (h) Nature FCC

Fig. 5. The Performance in Simulation Environments

We define the QoE_{mean} as the average QoE in the video time $time_{video}$ for each video. As the duration of each chunk may be different, we define $norm\_VQA^{log}(R_n)$ to calculate the smoothness, which avoids the influence of duration. The QoE function for the virtual play is described in Eq-5. We consider the weight of rebuffing penalty α as the video quality of the 4K video and the weight of smoothness penalty β as 1 recommended by [23].

$$\begin{cases} QoE_{all} = \sum_{n=1}^{N} VQA^{log} - \alpha * \sum_{n=1}^{N} rebuf\_time_n \\ \qquad -\beta * \sum_{n=1}^{N-1} (VQA^{log}(R_{n+1})/L_{n+1} \\ \qquad -norm\_VQA^{log}(R_n)) \\ norm\_VQA^{log}(R_n) = VQA^{log}(R_n)/L_n \\ QoE_{mean} = \frac{QoE_{all}}{time_{video}} \end{cases} \tag{5}$$

We consider the HYB[2], a wildly used in industrial, as the default ABR algorithm, and we use parameters recommended by the recent work [19]. In § VII, we will discuss the influence of different ABR algorithms. When the virtual player plays, we record the current time, and at time t, we check whether it is legal for the video to switch duration. For example, we have three combinations $(1s, 2s, 4s)$. Then at time $0s$, these duration are all legal. If the ABR algorithm chooses the duration of $2s$ as the first segment, at the next decision time $2s$, the $4s$ duration is illegal, so only $1s$ and $2s$ duration can be chosen.

C. *Expert Solver*

This solver gives us the best combination from the optional set and gives the signal on whether to end this iteration. For every step, the expert solver separately adds every optional combination to the selected set and uses the virtual player to play. After getting the QoE from the virtual player, the expert solver calculates the score using Eq-2 and considers the combination that gains the highest score as the expert action. The expert solver records the max score after each step; if the score does not increase in training, the solver gives the end signal to the system.

[2]https://github.com/Dash-Industry-Forum/dash.js/wiki/ABR-Logic#primary-rules

D. *Experience buffer and Loss function*

To improve the data utilization, we construct an experience pool to store the pair of states and the expert action motivated by [31]. As the pair of the current state and the expert action is not relative to the current policy, we can randomly pick the sample from the buffer during the training process, which can take full use of previous data.

For our system, we encode our action a as the one-hot n-dim vector. As the relevance of different actions is hard to quantify, we use cross-entropy error as the loss function. In imitation learning, we hope to minimize the distance between the current policy $\pi(\theta)$ and the expert policy π^*. For the current state s, we can update the model by minimizing the gap between the current policy $\pi(s, a : \theta)$ and the expert action A^*. So the loss function of RDladder can be described as Eq-6.

$$L_{RDladder} = -\sum A^* log\pi(s, a : \theta) \tag{6}$$

E. *Implementation*

We now show the detail of the implementation of RDladder. We use Tensorflow to implement the training workflow and TFlearn to construct the NN architecture. We use python to implement the instant solver and the virtual player. We set the number of duration as 5, i.e., $dur \in \{1, 2, 3, 4, 5\}$, and set the number of resolutions as 7, i.e., $res \in \{144p, 240p, 480p, 720p, 1080p, 2k\ (1440p), 4k\ (2160p)\}$, so the number of candidates is 35. For the weight of the storage cost, we randomly choose the integer from the interval [0,30] during training.

VI. Evaluation

In this section, we will show our results using various experiments. At first, we will introduce our setup of all experiments in § VI-A, including the video datasets, network datasets, and some baselines we will compare. Then we will introduce our results in the simulation experiments (§ VI-B). Next, we will discuss our trace-driven emulation experiments (§ VI-C) and real-world experiments (§ VI-D). In this section, we still use HYB as the default ABR algorithm, and the influence of different ABR algorithms will be discussed in the next section.

A. *Experimental Setup*

Video datasets We collect over 500 4K videos to provide various video content, including four categories: games, sports, comics, and nature. We use these 4k videos to scale different other resolutions (144p, 240p, 480p, 720p, 1080p, 2k) via bicubic filtering of FFmpeg with original fps. Then we use one-pass VBR to encode these videos to get different duration (1s, 2s, 3s, 4s, 5s) chunks via x264 with $crf = 16$. We calculate the Structural Similarity(SSIM) for each frame referred with the source 4k videos using FFmpeg and calculate their VQA^{log} based on the SSIM.

Network datasets We collect various real-world traces from previous work, such as FCC [17], Oboe [19], HDFS [20] and HSDPA [18]. We use 90% traces of HDFS as the train set, and we scale them from 0.5X to 3X to improve the diversity of the network information. We think these data from the average of 0.4mbps to 27mbps can reflect most of the common network environments in the real world.

Baselines To better demonstrate the results of our system, we select four baselines to compare. We use the default 4s duration and full resolutions as the All. We select some resolutions with 4s duration whose bitrates are close to the average bandwidth as $Base$. And we use the method VAR mentioned by the [15], to produce the variable duration. We consider the recommended duration by [2] and set the $max\_dur = \{2, 4\}$ of VAR called $VAR - 2$ and $VAR - 4$ respectively. Besides, we design the sequential greedy optimal method SO to show the highest score in this sequential system under given network environments. We calculate each new score for the optional combination and select the combination with the highest score for the selected set. We repeat this process until the score does not improve.

B. *Simulation Experiments*

Overall Performance In this part, we show overall results in simulation experiments. We select four different types (Comic, Sports, Game, Nature) videos that NN does not see before to test. We test these videos on Oboe trace and the FCC trace with different size weight $w \in \{0, 20\}$ and the scores (Eq-2) are illustrated in Figure 5. The $Base$ selects 4 resolutions close to the average bandwidth (480p, 720p, 1080p and 1440p in Oboe; 144p, 240p, 480p, 720p in FCC).

When $w = 0$, in some conditions, the performance of $RDladder + 0$ is much better than $Base + 0$, such as Figure 5(d) 5(g) 5(h), which indicates $RDladde + 0$ chooses better combinations even if we do not consider the benefit of video size. Besides, we can find that in some conditions, $RDladder + 0$ can be even better than $All + 0$, such as Figure 5(e) (score from 42 to 45) and Figure 5(h) (score from 32 to 37), which shows the duration switch provides more benefit using fewer resolutions in this conditions.

We can see in these experiments that the performance of $RDladder$ is close to the SO, especially when $w = 20$. For VAR, we can see that its performance is close to the $SO + 0$ in most cases, but if the variable duration can save many sizes compared with the fixed encoding, it achieves excellent

TABLE I
RESULTS OF SIMULATION EXPERIMENTS

Type	Score	QoE	Rebuf/s
RDladder+0	34.47 (± 3.98)	34.47 (± 3.98)	13.73 (± 4.39)
RDladder+20	53.13 (± 3.78)	33.88 (± 3.74)	13.30 (± 4.08)
Base+0	30.12 (± 3.83)	30.12 (± 3.83)	22.24 (± 4.39)
Base+20	44.49 (± 3.76)	30.12 (± 3.83)	22.24 (± 4.39)
SO+0	35.94 (± 3.69)	35.94 (± 3.69)	12.48 (± 4.13)
SO+20	54.00 (± 3.77)	35.36 (± 3.76)	12.48 (± 4.23)
VAR-2+0	35.74 (± 3.00)	35.74 (± 3.00)	29.80 (± 9.60)
VAR-2+20	43.35 (± 5.32)	35.74 (± 3.00)	29.80 (± 9.60)
VAR-4+0	37.35 (± 3.47)	37.35 (± 3.47)	25.34 (± 7.74)
VAR-4+20	44.95 (± 5.83)	37.35 (± 3.47)	25.34 (± 7.74)

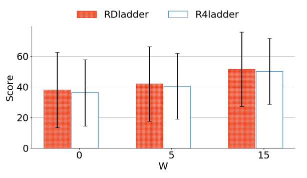


Fig. 6. The Results of Duration Switch

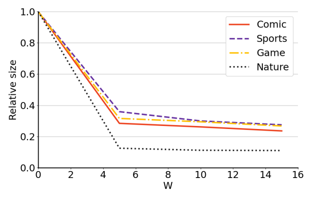


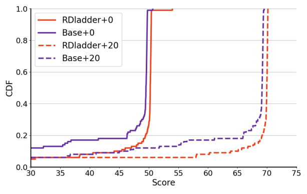
Fig. 7. The Relative Size When w Changes

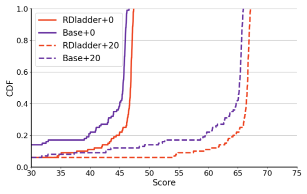
performance, such as $VAR - 2 + 0$ compared with $SO + 0$ in Figure 5(f) and Figure 5(e). However, when we consider the storage cost in $w = 20$, the score of VAR indeed increases but is significantly lower than the $RDladder$. We show the details in FCC, including the average number and the standard deviation of the score, QoE, and rebuffing time in Table-I. We can see when $w = 0$, RDladder improves 13% score compared with $Base$ and when $w = 20$, RDladder improves 19%, 22%, 18% compared with $Base+20$, $VAR-2+20$, $VAR-4+20$ respectively.

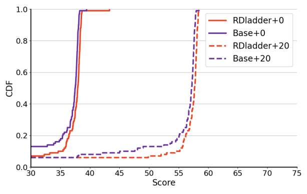
The Influence of Duration Switch To show the influence of the duration, we design a new baseline, R4ladder, which has the same resolutions as the RDladder, but the duration is all 4s. We test them both in 100 traces of FCC and calculate their average scores with $w \in \{0, 15\}$. We compare each score and show the results in Figure-6. We can see that the RDladder is better than R4ladder in most scenarios, especially when the weight of size w is small.

The Influence of size weight w To show the size change in different size weights w, we change w from 0 to 15 and calculate the relative size compared with the ladder when $w = 0$. As shown in Figure-7, we can see clearly that as w increases, the relative size can even decrease to 20%. It means that if constrained by strict storage overhead, only two or three ladders carefully selected according to the video content and network condition can achieve satisfactory results.

Sequential Performance Furthermore, as we consider the transcoding as a sequential process, we compare the performance of $RDladder$ and $Base$ in the same number of conditions as Figure 8. We set the ladder number as two, and we can see that the RDladder is much better than $Base$ (240p and 480p).


(a) Comic


(b) Sports


(c) Game

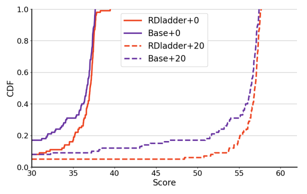
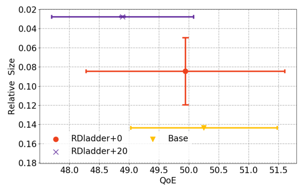
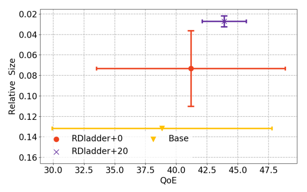
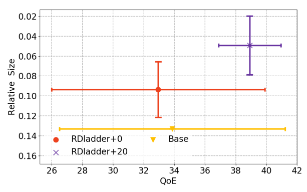

(d) Nature

Fig. 8. The Sequential Performance in FCC with 2 ladders


(a) Comic HSDPA


(b) Sports HSDPA


(c) Game HSDPA

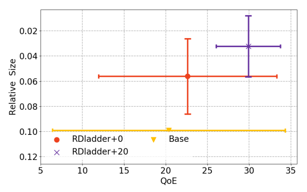
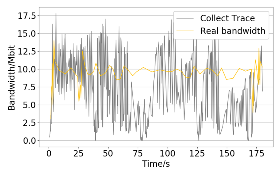

(d) Nature HSDPA

Fig. 9. The Performance in Emulation Environments

C. Emulation Experiments

We construct an emulation environment to test its performance. In detail, we use mahimahi [32], a truthful emulator to reflect the network traffic to emulate the network between the video server and the client player. The client player requests the chunks from the video server via the ABR algorithms. Then the player plays videos virtually to calculate the download time, update the current buffer, and record the video quality and rebuffering time. After receiving the request from the client, the video server sends the data packets of the same size as the video chunk controlled by the congestion control algorithm Cubic [33]. We compare $RDladder+0$ and $RDladder+20$ with the default setting with 3 resolutions (240p, 480p and 720p) $Base$ in HSDPA traces and use 4 videos to play. We run ten traces and calculate its average number and the standard deviation, illustrated in Figure 9. We can see $RDladder+0$ achieve a better average QoE than $Base$ in most conditions, improve 11% QoE, and reduce 35% size on average. The relative size of $RDladder+20$ compared with the default setting is significantly reduced by 77% on average with the cost of acceptable influence of QoE in most cases.


(a) Mobile Cellular Networks

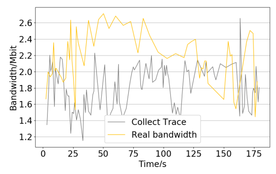
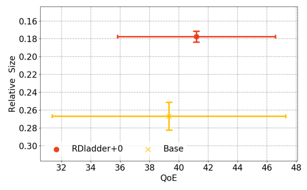

(b) Public WiFi

Fig. 10. The First 180s of Network


(a) Mobile Cellular Networks

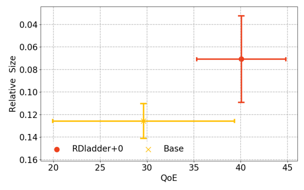

(b) Public WiFi

Fig. 11. The Performance of Real World

D. Real world Experiments

We deploy our systems in the real world. At first, we collect some bandwidth traces from a video player over 10 minutes. Then RDladder considers the network conditions and the video content to decide the combination of the resolution and duration. The server uses the setting from the RDladder to transcode the video to the chunk. We plot the first 180s of the collected network bandwidth and real network during playing in the mobile cellular network and public WiFi in Figure 10. We compare $RDladder$ with $Base$ which has 3 resolutions (240p, 480p, 720p in WiFi and 480p, 720p, 1080p in cellular) as Figure 11. We can see $RDladder$ achieve a relatively higher QoE and much less size. RDladder reduces 53% size and improves 4.7% QoE in cellular, and reduces 44% size and improves 36% QoE in WiFi. It is worth noticing there is a gap between the collected trace and the real bandwidth, but RDladder achieves great performance in both conditions.

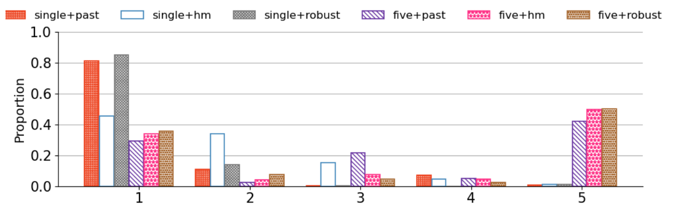


Fig. 12. The Proportion of Each Duration Selected by Different ABR Algorithms

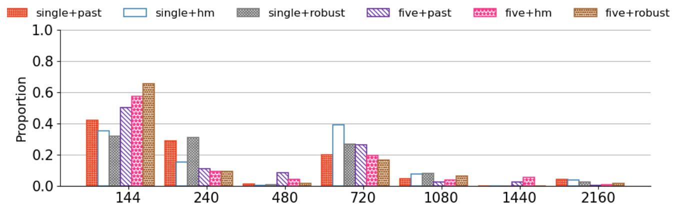


Fig. 13. The Proportion of Each Resolution Selected by Different ABR Algorithms

TABLE II
THE DETAILS OF 6 ABR ALGORITHMS IN OBOE

Type	Score	QoE	Mean of Dur/s	Mean of Ladder Length	Mean of Video Resolution Levels
Single-past	62.55(±16.04)	53.57(±13.72)	1.36(±0.83)	3.3(±1.18)	1.59(±1.79)
Single-hm	64.25(±18.6)	55.48(±16.59)	1.83(±0.9)	3.29(±1.27)	1.97(±1.73)
Single-robust	61.57(±15.84)	52.62(±13.25)	1.15(±0.47)	3.27(±1.17)	1.62(±1.59)
Five-past	64.37(±17.34)	55.15(±15.35)	3.43(±1.59)	3.62(±1.2)	1.52(±1.6)
Five-hm	61.16(±17.11)	52.56(±14.73)	3.5(±1.75)	3.42(±1.18)	1.53(±1.8)
Five-robust	60.62(±14.02)	51.38(±11.79)	3.4(±1.84)	3.45(±1.4)	1.17(±1.7)

TABLE III
TOP 5 OF THE SELECTED LADDER IN OBOE

Type	Ladder
Single-past	(1s, 144p), (1s, 720p), (1s, 240p), (2s, 1080p), (2s, 2160p)
Single-hm	(1s, 144p), (2s, 720p), (3s, 720p), (2s, 1080p), (1s, 240p)
Single-robust	(1s, 144p), (1s, 240p), (1s, 720p), (1s, 1080p), (2s, 720p)
Five-past	(5s, 720p), (3s, 720p), (5s, 144p), (1s, 144p), (3s, 480p)
Five-hm	(5s, 720p), (1s, 144p), (5s, 144p), (3s, 144p), (1s, 480)
Five-robust	(5s, 144p), (1s, 144p), (5s, 720p), (5s, 1080p), (1s, 240p)

VII. THE INFLUENCE BETWEEN RDLADDER AND ABR ALGORITHM

In this section, we will discuss the influence between RDladder and ABR algorithm. We consider the framework of MPC and change the method of bandwidth prediction and look-ahead horizon to get different ABR algorithms. We consider three different prediction methods $past$,$robust$, and hm, the first two methods have been introduced in § III-B and the hm method is used to calculate the harmonic mean of the past five bandwidth. Moreover, we also consider two look-ahead horizons, single-step and five-step. We run these six algorithms in Oboe with $w \in \{0, 20\}$ and use different bandwidth utilization 0.5 and 1 to improve the network diversity. We plot the proportion of each duration selected by different ABR algorithms in Figure 12, each resolution selected in Figure 13, the detail of these six algorithms in Table II and we show the top 5 of the selected ladder in Oboe trace in Table III. We summarize four characteristics as follows:

- RDladder improves the performance of the ABR algorithm in the client. As shown in Table II, We can see that the performance of only the 1-step look-ahead horizon is close and even better than the 5-step horizon. And we find that in most cases, only 2 or 3 combinations recommended by RDladder can achieve great performance.
- Although the mean of the duration is close to the recommended fixed duration 4s [2], 1s and 5s duration are the most popular duration, as shown in Figure 12. We find that the 5-step horizon prefers more long duration than the single-step horizon. The reason behind this is long duration has more advantages in a long-term view, so it prefers long duration to provide more buffer, which improves the video quality level in the future.
- In Figure 13, the popular resolution is 144p, 720p, and 240p, but the middle resolution has less chance of being chosen. Low resolutions are selected more due to the network's limitation. And 720p is the highest resolution that the client can choose in most of these network conditions. So, we can see that the ladder prefers the low resolution to avoid rebuffing and cooperates with the high resolution to achieve higher video quality.
- We find that different ABR algorithms prefer different ladders, as shown in Table III. We can see that the long look-ahead horizon prefers a conservative strategy and lower resolution than the short look-ahead horizon, which has larger influence than prediction methods.

VIII. DISCUSSION

How general is RDladder's model? We have collected a large amount of network and video data in the hope of reflecting as much as possible on the real-world streaming environments. However, there is still much unknown about deep neural networks. Besides, as shown in § VI-D, we can see the gap between the collected network traces and the real network in play. Although we use the high-dimensional representation of the network, it is hard to guarantee its performance in any conditions. One possible approach is to add redundant candidates, so how to balance the performance safety and the additional size may be an interesting problem.

Is RDladder the optimal solution? We believe RDladder is a great solution to optimize the duration and resolution. However, we think there is still much room to improve for the final QoE as current ABR algorithms do not fully use the advantage of the selected resolutions and duration switch. For example, a longer duration will likely meet bandwidth fluctuations. If RDladder provides the same resolution with different duration, the ABR algorithm will select the higher QoE in its horizon but ignore the different risks. So a better risk-aware ABR may make better use of the duration switch.

IX. CONCLUSION

In this work, we analyze the importance of resolution optimization and duration switch, and propose joint optimization of resolutions and duration. So we design RDladder, a resolution-duration ladder for VBR-encoded videos via imitation learning. We consider the information about the video, network, and storage cost. We test our system in simulation, emulation, and real-world environments and achieve near-optimal results. In addition, we discuss the influence between the RDladder and the ABR algorithm and introduce some interesting characteristics with different ABR algorithms.

Acknowledgment We thank the anonymous INFOCOM reviewers for their constructive feedback. This work was supported by NSFC under Grant 61936011, Beijing Key Lab of Networked Multimedia, and the Kuaishou-Tsinghua Joint Project.

References

[1] sandvine, "sandvine-releases-covid-19-global-internet-phenomena-report," https://www.sandvine.com/press-releases/sandvine-releases-covid-19-global-internet-phenomena-report, 2020, [Online; accessed 7-May-2020].

[2] bitmovin, "Optimal adaptive streaming formats mpeg-dash hls segment length." https://bitmovin.com/mpeg-dash-hls-segment-length/, 2015, [Online; accessed 1-April-2022].

[3] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, "Abr streaming of vbr-encoded videos: characterization, challenges, and solutions," in *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, 2018, pp. 366–378.

[4] T. Huang and L. Sun, "Optimized bitrate ladders for adaptive video streaming with deep reinforcement learning," in *Proceedings of the SIGCOMM'20 Poster and Demo Sessions*, 2020, pp. 46–48.

[5] A. Mondal, S. Sengupta, B. R. Reddy, M. Koundinya, C. Govindarajan, P. De, N. Ganguly, and S. Chakraborty, "Candid with youtube: Adaptive streaming behavior and implications on data consumption," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2017, pp. 19–24.

[6] Google, "Recommended upload encoding settings," https://support.google.com/youtube/answer/1722171, 2022, [Online; accessed 1-April-2022].

[7] Apple, "Http live streaming (hls) authoring specification for apple devices," https://developer.apple.com/documentation/http_live_streaming/, 2022, [Online; accessed 1-April-2022].

[8] Twitch, "Broadcasting guidelines," https://stream.twitch.tv/encoding/, 2022, [Online; accessed 1-April-2022].

[9] J. De Cock, Z. Li, M. Manohara, and A. Aaron, "Complexity-based consistent-quality encoding in the cloud," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1484–1488.

[10] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, and P. Frossard, "Optimal selection of adaptive streaming representations," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 11, no. 2s, pp. 1–26, 2015.

[11] C. Chen, Y.-C. Lin, S. Benting, and A. Kokaram, "Optimized transcoding for large scale adaptive streaming using playback statistics," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 3269–3273.

[12] Y. A. Reznik, X. Li, K. O. Lillevold, A. Jagannath, and J. Greer, "Optimal multi-codec adaptive bitrate streaming," in *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2019, pp. 348–353.

[13] L. Bedogni, M. Di Felice, and L. Bononi, "Dynamic segment size selection in http based adaptive video streaming," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2017, pp. 665–670.

[14] B. J. Villa and P. E. Heegaard, "Group based traffic shaping for adaptive http video streaming by segment duration control," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2013, pp. 830–837.

[15] S. Schwarzmann, N. Hainke, T. Zinner, C. Sieber, W. Robitza, and A. Raake, "Comparing fixed and variable segment durations for adaptive video streaming: a holistic analysis," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 38–53.

[16] M. Licciardello, L. Humbel, F. Rohr, M. Grüner, and A. Singla, "Prepare your video for streaming with segue," *arXiv preprint arXiv:2202.09112*, 2022.

[17] M. F. B. Report, "Raw data measuring broadband america 2016," https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016, 2016, [Online; accessed 19-July-2016].

[18] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 2013, pp. 114–118.

[19] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 44–58.

[20] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: Improving bitrate adaptation in the dash reference player," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 2s, pp. 1–29, 2019.

[21] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[22] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.

[23] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "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*, 2015, pp. 325–338.

[24] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[25] P. ITU-T RECOMMENDATION, "Subjective video quality assessment methods for multimedia applications," 1999.

[26] C. Kreuzberger, B. Rainer, H. Hellwagner, L. Toni, and P. Frossard, "A comparative study of dash representation sets using real user characteristics," in *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2016, pp. 1–6.

[27] Y. A. Reznik, K. O. Lillevold, A. Jagannath, J. Greer, and J. Corley, "Optimal design of encoding profiles for abr streaming," in *Proceedings of the 23rd Packet Video Workshop*, 2018, pp. 43–47.

[28] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on image processing*, vol. 15, no. 11, pp. 3440–3451, 2006.

[29] H. Zhang, X. Jiang, and X. Lei, "A method for evaluating qoe of live streaming services," *international Journal of computer and electrical engineering*, vol. 7, no. 5, p. 296, 2015.

[30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[32] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate {Record-and-Replay} for {HTTP}," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, 2015, pp. 417–429.

[33] 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.