DataPlanner: Data-budget Driven Approach to Resource-efficient ABR Streaming

Yanyuan Qin[1], Chinmaey Shende[1], Cheonjin Park[1], Subhabrata Sen[2], Bing Wang[1]
[1]University of Connecticut [2]AT&T Labs - Research

ABSTRACT

Over-the-top video (OTT) streaming accounts for the majority of traffic on cellular networks, and also places a heavy demand on users' limited monthly cellular data budgets. In contrast to much of traditional research that focuses on improving the quality, we explore a different direction—using data budget information to better manage the data usage of mobile video streaming, while minimizing the impact on users' quality of experience (QoE). Specifically, we propose a novel framework for quality-aware Adaptive Bitrate (ABR) streaming involving a per-session data budget constraint. Under the framework, we develop two planning based strategies, one for the case where fine-grained perceptual quality information is known to the planning scheme, and another for the case where such information is not available. Evaluations for a wide range of network conditions, using different videos covering a variety of content types and encodings, demonstrate that both these strategies use much less data compared to state-of-the-art ABR schemes, while still providing comparable QoE. Our proposed approach is designed to work in conjunction with existing ABR streaming workflows, enabling ease of adoption.

CCS CONCEPTS

• **Information systems** → Multimedia streaming.

KEYWORDS

Adaptive Bitrate (ABR) streaming; Data budget; Resource efficiency.

ACM Reference Format:
Yanyuan Qin, Chinmaey Shende, Cheonjin Park, Subhabrata Sen, and Bing Wang. 2021. DataPlanner: Data-budget Driven Approach to Resource-efficient ABR Streaming. In *12th ACM Multimedia Systems Conference (MMSys'21), Sept. 28-Oct. 1, 2021, Istanbul, Turkey.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3458305.3459596

1 INTRODUCTION

Video streaming over cellular networks provides users convenient streaming experience anytime anywhere. On the other hand, video streaming is one of the most bandwidth consuming applications: streaming just one-hour High Definition (HD) video can consume a significant portion of a user's monthly data plan. In addition, video streaming traffic imposes significant load on cellular networks.

Reducing the data usage of mobile video streaming while minimizing the impact on users' QoE is the key to enable users to consume more good-quality content within their specific monthly data budgets. In addition, reduced data usage per session translates to reduced load on cellular networks, and hence potentially better QoE for other users sharing the same cellular infrastructure.

In this paper, we explore reducing data usage in the context of ABR streaming, the *de facto* over-the-top video streaming technology in industry, for the popular Video-on-Demand (VOD) use case. In ABR streaming, for each video, the server creates a *track ladder*, consisting of multiple independent *tracks*, each encoding the same content, but differing in frame rate, encoding bitrate, resolution, or perceptual quality. Each track is further divided into a series of *segments*, each containing data for a few seconds' (typically 2-10 seconds) worth of playback. For each segment position in the video, the encoding bitrates and hence the perceptual quality generally progressively increase from lower to higher tracks. During streaming playback, the ABR client leverages an adaptation logic to dynamically determine which quality variant (i.e., from which track) to fetch for each segment position in the video. For good QoE, the ABR streaming logic needs to account for and balance across the conflicting goals of maximizing quality, minimizing rebuffering, and minimizing quality changes [11, 22, 28, 33].

Existing approaches in industry to reducing data usage are either *service-based* or *network-based* [37]. A service-based approach is typified by a video service providing a user multiple options via the client player user interface; users can choose an option with a lower data usage at the cost of lower quality experience. The selected option is mapped to a specific track in the track ladder, and the ABR track selection during playback is constrained to selecting from variants at or below that track (see §2). In a network-based approach, a mobile network operator caps the maximum network bandwidth available to a video session to a fixed value [3, 40].

Existing ABR adaptation schemes in literature (§7) focus mainly on maximizing the video quality under the network bandwidth constraints, and not on limiting data usage. One exception is [37], which assumes a user specified target quality and proceeds to avoid segments whose qualities exceeding the target quality, leading to bandwidth savings. The approach, however, still does not provide a user an explicit control of data usage. For the same target quality specified by the user, the amount of data downloaded for one video can be significantly higher than that for another video (see §2).

Overall, the existing solutions to limiting video streaming data usage, however, may have difficulty in achieving a good balance between QoE and associated data usage, as they variously do not account for one or more important factors such as different video

The work of Yanyuan Qin was done while he was a PhD student at the University of Connecticut.

genres and encoding technologies, complexity and quality variability across different scenes in a video track, and different QoE vs. bitrate tradeoffs for different ABR track ladders. They also do not consider overall data usage in the rate adaptation decision.

In contrast to these existing approaches, in this paper we explore a different direction—using data budget information to better manage the data usage of mobile video streaming, while minimizing the impact on users' QoE. Specifically, we propose a novel framework for quality-aware ABR video streaming involving a *per-session data budget constraint*, which provides a knob to explicitly constrain the amount of data that can be used for a streaming session. This problem is challenging since we need to account for all the factors discussed above. In addition, to be readily deployable, the technique needs to work in conjunction with existing ABR streaming workflows. Our work resolves the above challenges and makes the following main contributions:

• We present a client-based framework, DataPlanner (DP), for incorporating per-session data budget into ABR streaming (§2). DP dynamically rations data usage, accounting for the remaining data budget and segments in a streaming session, and can be easily retrofitted into existing ABR streaming workflows.

• Under the above framework, we develop two novel computationally light-weight strategies (§3) for guiding ABR track selection: DP-Q for the scenarios where per-segment perceptual quality is known beforehand, and DP-T for the scenarios where such information is not available. For the former case, the strategy determines a target quality (and hence the respective target tracks) for the remaining segments, while the latter strategy directly determines the target tracks, taking account of the characteristics of state-of-the-art variable bitrate (VBR) encodings, including H.264 [19], H.265 [18] and VP9 [35].

• We evaluate the performance of integrating each of DP-Q and DP-T with two existing state-of-the-art ABR schemes, RobustMPC [43] and CAVA [36] (§4). In addition, we implement DP-Q and DP-T in a popular open-source ABR player software, Exoplayer [15], on the Android platform, which is widely used in commercial players [16] (§5). Evaluations across a wide range of cellular network conditions, using different videos covering diverse content types, codecs, and track ladder settings demonstrate that DP-Q and DP-T can successfully balance QoE while satisfying the data budget constraint—they achieve QoE that is close to that of running the original ABR schemes in standalone mode, but with substantially lower data usage. For instance, when the network bandwidth is high and the data budget is relatively low, the original scheme uses on average 93% and 90% more data than DP-Q and DP-T respectively, but with comparable percentage of low-quality segments, and higher rebuffering.

• Our results highlight the importance of designing DP schemes carefully to avoid impacting QoE (§2 and §4). A naive application of data-budget constraints as exemplified by a strawman scheme that is similar to the current practice is not sufficient. While obeying the data budget, it can lead to ABR adaptation decisions with significant adverse QoE implications. In contrast, a carefully designed DP scheme with the same data budget should be able to realize significantly improved QoE, by considering both the data budget and the QoE dimensions. Our designed DP schemes achieve much better quality for segments with complex scenes than the strawman scheme: the quality is up to 16 points higher measured by Video Multimethod Assessment Fusion (VMAF) phone model [24], a state-of-the-art perception quality metric.

• Based on the evaluation results, we discuss two other practical aspects related to data planning: (i) how to set per-session data budget based on the video characteristics, and (ii) using DP strategies in conjunction with network-based bandwidth caps to both deliver good QoE and constrain data usage (§6).

2 MOTIVATION AND SOLUTION FRAMEWORK

In this section, we first describe the benefits of introducing per-session data budget in ABR streaming, and then the problem setting. After that, we outline a simple strawman data planning strategy and point out its limitations. Finally, we present a high-level framework for quality-aware ABR streaming under a per-session data budget.

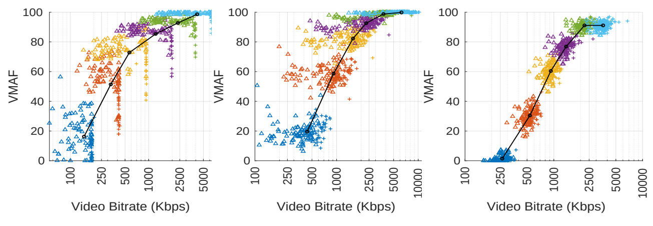
2.1 The Case for Per-session Data Budget

A per-session data budget allows a user to explicitly control the amount of data consumed in a video streaming session. For the user, it provides upfront clarity at the beginning of a session about the estimated data usage for watching the particular video. This in turn enables a user to better reason about and select the specific data budget setting (e.g., select a higher or lower budget) for the session in a more informed way, taking into account factors like their remaining cellular data budget for the month, and how much data they are willing to expend for the particular session (more in §6). In addition, a per-session data budget opens the way for the ABR streaming system to explicitly factor this constraint in the track selection decisions throughout the session. A carefully designed scheme would judiciously plan and ration the budget over time, subject to varying available network bandwidths across the session and varying content complexities across the video, while steering the selection towards delivering good QoE.

A data-budget aware ABR scheme can also take advantage of the diminishing gain in quality with increasing encoding bitrate that is common across a wide range of codecs (e.g., H.264, H.265 and VP9) and state-of-the-art ABR track ladders of commercial streaming services (e.g., used by YouTube and Netflix) [9, 37]. Table 1 lists seven H.264 and H.265 videos that are encoded by YouTube or by us (see more details in §4.1). Fig. 1 plots the perceptual quality as measured by the VMAF phone model [24] versus bitrate (in log scale) for three VBR videos. The first two videos, ED (Elephant Dream) and AF (American Football), are H.264 encoded animation and sport videos, respectively, and the third one, BB (Basketball), is an H.265 encoded sport video. They have 104, 135 and 136 segments, respectively, each of around 5-sec duration, and 6 tracks. In Fig. 1, each point corresponds to a segment, the segments in the same track are represented in the same color, and the black curve shows the average quality versus average encoding bitrate for each track. For all three videos, we clearly see that as the bitrate reaches a certain level, increasing it further only leads to little improvement of perceptual quality—only a VMAF difference of 6 or more is considered to be noticeable to a viewer [9, 34] and VMAF

Table 1: Average bitrate (in Mbps) with the peak-to-average bitrate ratio in parenthesis for seven videos.

	H.264 encodings				H.265 encodings		
	BBB	ED	ToS	AF	AF	BB	FH
144p	0.10 (1.3)	0.10 (1.3)	0.09 (1.2)	0.29 (1.9)	0.24 (2.0)	0.16 (1.4)	0.14 (1.6)
240p	0.22 (1.3)	0.23 (1.3)	0.20 (1.1)	0.61 (1.9)	0.45 (2.0)	0.34 (1.4)	0.30 (1.7)
360p	0.34 (1.9)	0.39 (1.6)	0.34 (1.4)	1.06 (1.9)	0.74 (2.0)	0.58 (1.3)	0.56 (1.7)
480p	0.73 (1.8)	0.84 (1.6)	0.76 (1.3)	1.54 (1.9)	1.03 (2.0)	0.86 (2.0)	0.86 (1.8)
720p	1.37 (1.9)	1.62 (1.7)	1.42 (1.4)	2.51 (2.2)	1.58 (1.9)	1.39 (2.0)	1.51 (2.0)
1080p	2.42 (2.0)	2.86 (1.6)	2.31 (1.4)	4.15 (2.4)	2.51 (1.8)	2.24 (1.9)	2.67 (2.1)



(a) ED (H.264) (b) AF (H.264) (c) BB (H.265)

Figure 1: Perceptual quality and bitrate tradeoff. The segments in the same track are represented in the same color; the six colors correspond to the six tracks. Triangles are for Q1-Q3 segments (with simple scenes; see §3.2) and crosses are for Q4 segments (with complex scenes).

values above 80 are considered as good quality [4, 25]. Take the ED video as an example: many segments in track 3 (360p) have VMAF values between 60 and 80 (considered as fair to good quality), many segments in track 4 (480p) have VMAF values exceeding 80 (considered as good to excellent quality), and higher tracks (tracks 5 and 6) only have slightly higher VMAF values at the cost of significantly higher bitrate. Given the above characteristics of this video, a data-budget aware ABR scheme would have a strong incentive to prefer the track 4 variant of a segment over its track 5 variant due to its marginally lower quality but significantly lower data usage. While the specific bitrate vs. QoE tradeoff for track selection may differ across different videos and different track ladder schemes, the overall diminishing returns behavior shown in Fig. 1 holds throughout. We describe how our DP framework and schemes incorporate and utilize such trends in §3 and §6.

Last, while an earlier work [37] also aims to reduce data usage, it assumes a target perceptual quality level and attempts to deliver the ABR content at that quality. It does not provide an explicit control on per-session data usage as in our approach. As an example, suppose the target perceptual quality is VMAF 80. Then consider 8-min long videos in Table 1, with the approach in [37], the data usage is up to 29 MB and 258 MB for BBB and AF (H.264), respectively, the latter being 9× as high as the former. In contrast, with our DP approach, a user selects a per-session data budget, and the resultant data consumption is bounded by that selected data budget.

Determining per-session data budget. In general, the data budget for a particular session can be determined by a policy mechanism. Specifically, we envision a simple User Interface (UI) that tells the user (i) the remaining cellular data for the month, and (ii) some options to select from, e.g., excellent quality with x MB data consumption, good quality with y MB data consumption, fair quality with z MB data consumption. The mapping of the choices to the data budget values needs to account for the specifics of a video. In particular, different videos have different content complexity; the codecs and track ladder designs used to create the ABR tracks can also lead to significantly different bitrates per track. For instance, for the videos in Table 1, if the policy is to watch a 10-minute video with a data budget of 75 MB (translates to an average bitrate of 1 Mbps), then it can be very limiting for certain videos, while not being a significant constraint for others. In §6, we discuss creating such data budget mappings based on insights from our evaluation results, and highlight the differences of our mappings from existing practice.

2.2 Problem Setting

Recall that in this paper, we focus on the popular VOD streaming use case. At the origin server side, a typical VOD library consists of videos from different genres and with different content complexities (e.g., talking heads vs. fast-paced action), each converted into a set of ABR tracks with different resolutions, frame rates, bitrates or quality levels, using the ABR track ladder design adopted by the particular streaming service. At the beginning of a streaming session, the client downloads a manifest file from the server. Important information related to data-usage requirements, such as track encoding bitrates, and per-track and per-segment sizes, can be easily gleaned from a combination of (i) information already included in the manifest file today in many cases, and (ii) additional information that the server can either piggyback in the manifest file or via an auxiliary information exchange [36, 37].

Let D denote the data budget (in bytes) for the streaming session, which is selected by the user when the session starts (see §2.1 and §6). *Our goal is to design data planning strategies so that the total data consumed (application-level data) is bounded by D, while still maintaining good QoE.*

One challenge is that commercial ABR client players tend to be sophisticated software systems with complex inter-dependencies, and vary in the extent of modifications/new feature additions that are allowed. While some players like Exoplayer [15] are open-source and therefore can be customized (this still involves deep technical domain expertise and complex engineering), others (e.g., AVFoundation [2]) are closed proprietary systems and limit modifications to a few configuration changes via a few fixed APIs.

Given such ground realities, to be useful in practice, any data planning strategy design needs to focus on being able to work in conjunction with existing player platforms and ABR adaptation logic, and still provide value. Our focus therefore is on designing such plug-and-play data planning strategies that can be easily inserted into the existing streaming workflows. The role of such a

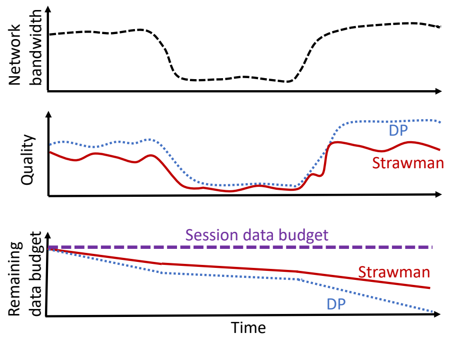


Figure 2: Illustration of the limitations of the strawman track-capping strategy.

strategy would then be to supplement and guide an ABR adaptation scheme towards more desirable choices to optimize the overall QoE, while adhering to the allocated session data budget D.

2.3 Simple Strawman Strategy

Consider a video with K tracks. Let S_k denote the size (in bytes) of track k, i.e., the sum of the segments in that track, $k = 1, \ldots, K$. Given a session data budget D, a simple strategy to satisfy the constraint is to determine the highest track, k^*, such that the corresponding size $S_{k^*} \leq D$, and limit the ABR track selection logic to tracks $k = 1, \ldots, k^*$. Specifically, for a segment position, if the selected track from an adaptation logic is i, then the actual selected track is $\min(k^*, i)$. This strategy, which we refer to as *track-capping*, is similar to what many commercial players do. For instance, the YouTube smartphone app provides an option "Play HD on Wi-Fi only", which only selects tracks with resolution no more than 480p over cellular networks, even when the network bandwidth is significantly higher. Similarly, Amazon Prime Video app provides multiple options ("Data saver", "Good", "Better", "Best"), which are realized by capping the top track [37].

While the above strategy is easy to implement, it has two key limitations: (i) it can lead to highly variable quality even under ample network bandwidths, since segments in the same track can have very different qualities (e.g., complex scenes have lower qualities than simple scenes) [36, 37], and (ii) it does not use any leftover data budget effectively, leading to poorer QoE than what can be achieved for the same network conditions. Fig. 2 shows an example. Suppose that the top track selected by the strawman scheme (based on the data budget) is track 3. Initially, the network bandwidth is high, above the bitrate of track 3, and hence track 3 is selected for the first 40 segments. These segments, although are at the same track level, are of significantly different quality (see Fig. 1). Later on, the network bandwidth drops significantly, and the ABR logic can only select track 2 for the next 10 segments, leading to significantly lower data usage than what the strawman scheme has anticipated. When the network bandwidth improves later on, the strawman scheme still caps the top track to track 3, despite the much higher remaining data budget than what it anticipated originally. As a result, when the session ends, the total data usage is significantly below the specified data budget. A more carefully designed DP strategy should significantly outperform the strawman scheme (see §2.4).

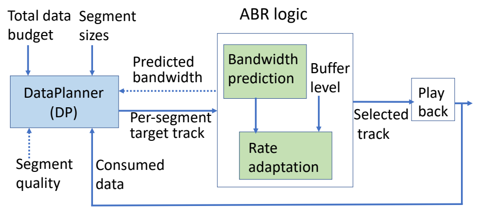


Figure 3: Framework of data-budget driven ABR streaming. Per-segment quality and predicted network bandwidth are optional inputs to DataPlanner (DP).

2.4 Proposed Data Planning Framework

Learning from the various limitations of the above strawman strategy, we next design a planning-based ABR adaptation framework that avoids such issues. The high level design is illustrated in Fig. 3. This framework includes a component called DataPlanner (DP), which interacts with the ABR logic at a client. While the ABR logic runs after each segment is downloaded to determine the track for the next segment, DP can run at a coarser time scale, e.g., every time interval Δ or after every N segments have been downloaded. At the beginning of the streaming session, DP determines the *target track* for each individual segment based on the total data budget and per-segment size (it can also use per-segment quality if that information is available; see discussion below). The target track L_i for segment i represents an upper bound on the rung in the ABR track ladder that can be selected for that segment, which is then used in conjunction with the core ABR logic to ensure that the selected track for segment i does not exceed L_i to satisfy the data budget constraint. Later on, each time DP runs, it calculates the remaining data budget as the total data budget subtracted by the amount of data that has been consumed until then, and *dynamically adjusts* the target track for each remaining segment position.

The above framework differs from the strawman strategy (§2.3) in two key aspects: (i) the target track for each segment is determined individually; at any instant, the planning approach conducts its fine-grained per-segment target track selection decision by globally considering the sizes and complexities of all the remaining video segments and the remaining data budget for the session, and (ii) the planning process runs periodically, revisiting and changing earlier decisions when appropriate, and so can better account for changing network dynamics and remaining data budgets.

Fig. 2 shows that the DP strategy (i) leads to more consistent quality when there is ample network bandwidth, and (ii) ramps up the quality to take advantage of the remaining data budget when it is higher than what was initially anticipated (e.g., due to lower network bandwidths for earlier segments). Overall, it allows a much better balance between data budget and quality than the static strawman track-capping strategy.

In the above framework, as mentioned earlier, the per-session data budget can be determined by a user through a simple UI based on the remaining monthly cellular data and other considerations (see more details in §6). We next discuss several other issues in a DP strategy design:

• **Using per-segment quality information or not?** When per-segment quality is available, the planning module can explicitly take account of the quality information to achieve similar quality across the segments, while satisfying the data budget constraint. This is important since as mentioned earlier, for the state-of-the-art codecs, segments within the same track can have highly variable quality [36, 37]. Therefore, per-segment quality provides finer-grained information that can be used to balance the quality across the selected segments. Ideally, this information can be exchanged as part of the manifest file shared with the client. In the shorter term, this can be achieved through an auxiliary information exchange. On the other hand, most services today do not share such quality information with the client. In such cases, the planning module cannot rely on per-segment quality and needs to use the available information around encoding bitrates and sizes for its decision process. In §3, we develop two strategies, one uses per-segment quality, while the other does not.

• **Frequency of planning.** One option is to run the data-budget based planning periodically, after every Δ seconds. Another option is to run it after every $N \geq 1$ segments have been downloaded. For both methods, we need to decide how often to plan by choosing Δ and N, respectively. A low value of Δ or N can lead to higher computation overhead, while a very high value can lead to slow reaction to the remaining data budget. We discuss the tradeoffs in detail in §4.

• **Interaction with ABR logic.** The data-budget based planning can influence the behavior of the ABR logic in two ways: (i) *cap-beforehand*, i.e., restricting the set of tracks to be selected in the ABR logic, or (ii) *cap-afterwards*, i.e., capping the selected track after the ABR logic. In both cases, the capping is for each decision made by the ABR logic (i.e., for each segment position), through well-defined APIs. The implementation complexity of these two approaches may differ, depending on the streaming system and the APIs that are available to the data planning module. In addition, they may lead to different performance, as we will explore in §4.

• **Interaction with bandwidth prediction.** An ABR logic may take account of the predicted network bandwidth during rate adaptation. The planning module can then access the predicted bandwidth through an API. It, however, may not choose to use the predicted bandwidth in its decision making because: (i) bandwidth prediction is only for a short-window, which is particularly true in cellular networks where bandwidth changes rapidly, while planning based on data budget operates at a longer time scale, and (ii) bandwidth prediction is already considered in the ABR logic, and hence may not need to be considered in the planning module. Alternatively, the planning module may consider the joint constraint of data budget and predicted network bandwidth. We take the former approach of not considering the predicted network bandwidth in the planning module, and leave the latter as future work.

3 DATA-BUDGET DRIVEN ABR STRATEGIES

In this section, we present two data-budget driven strategies for ABR streaming. The first strategy is for scenarios where per-segment perceptual quality information is available, while the second is for scenarios where such information is not available. For ease of exposition, our description below considers the beginning of a streaming session with a video of n segment positions and a session data budget D. The planning at each time t in the middle of the session considers the remaining outstanding segment positions $n_t < n$ and remaining data budget D_t (which is D subtracted by the amount of data that has been consumed up to time t), and can be done similarly.

3.1 DP with Per-segment Quality Information

When per-segment quality is available, we design a DP scheme that leverages such information, called DP-Q. DP-Q finds a *target quality*, q^*, so that when the quality of each segment is capped close to q^*, the total data usage is close to, but below the data budget D. After finding q^*, DP-Q further determines the target track for each segment i (see below).

We next describe how to find q^*. For segment index i, let $q_{i,\ell}$ denote the quality of the variant at track ℓ, and $s_{i,\ell}$ denote the size of this variant (in bytes). Given a quality q, for segment index i, consider the variant whose quality is closest to q, and denote its track level as $L_i(q)$, i.e., $L_i(q) = \arg\min_{\ell=1,\ldots,K} |q - q_{i,\ell}|$, where K is the number of tracks. In this way, we map a quality to a particular track for each segment. Let $S(q) = \sum_{i=1}^{n} s_{i,L_i(q)}$. That is, $S(q)$ is the sum of the sizes of the variants at track $L_i(q)$ for all n segments, which is the total data usage corresponding to quality q. Then, the target quality q^* is the maximum quality so that the total data usage is below D, i.e., $q^* = \arg\max_q S(q) \leq D$. Once the target quality q^* is determined, DP-Q determines the corresponding target track for segment i simply as $L_i(q^*)$, which will be fed to the ABR logic to limit the track selection for segment i.

In the above, the target quality is deliberately kept the same for all the segments being considered, in order to achieve a more consistent viewing quality across the segments, which is desirable. The next time DP-Q runs, it will repeat the above process, based on the remaining data budget, for the remaining segments, to obtain a new target quality, which may differ from the earlier target quality (e.g., due to a low network bandwidth condition earlier, the remaining data budget turns out to be higher than what is anticipated). As a result, the target track for a remaining outstanding segment may differ from an earlier planned value.

DP-Q can use binary search to find the target quality efficiently, as shown in Algorithm 1. In the algorithm, initially q_{low} and q_{high} represent the minimum and maximum quality of all the segments, respectively. They are then adjusted based on whether $S(q_{mid})$, i.e., the total data usage corresponding to quality q_{mid}, is below or above the data budget D, where q_{mid} is the quality in the middle of q_{low} and q_{high}. If the difference of $S(q_{mid})$ and D is less than a small positive value δ_2, q_{mid} is returned as the target quality, or q_{mid} is returned when q_{high} and q_{low} differ by less than a small positive value δ_1.

3.2 DP without Quality Information

When per-segment quality is not available, we develop a DP scheme, called DP-T, based on using ABR track-level information available in the manifest file obtained from the server (which is the only information to gauge quality). Different from DP-Q, which first determines a target quality and then per-segment target track, DP-T directly determines the per-segment target track for each of the remaining segments.

The main idea of DP-T is setting target tracks for complex scenes (i.e., high-motion or high-complexity scenes) to a higher level than

Algorithm 1: DP-Q: Find target quality via binary search.

```
Input: per-segment size, per-segment quality, data budget
  $D$, minimum quality $q_{low}$, maximum quality $q_{high}$
Output: target quality
while $q_{high} - q_{low} > \delta_1$ do
    $q_{mid} = (q_{low} + q_{high})/2$
    if $|D - S(q_{mid})| < \delta_2$ then
        return $q_{mid}$
    else if $S(q_{mid}) < D$ then
        recur for the right half, $q_{low} = q_{mid}$
    else
        recur for the left half, $q_{high} = q_{mid}$
    end
end
return $q_{low}$
```

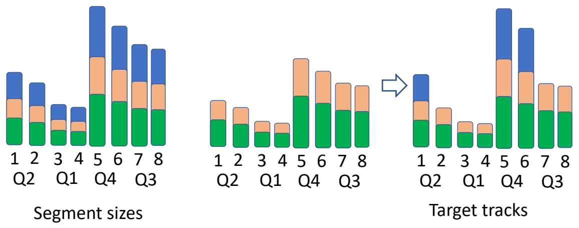


Figure 4: Illustration on how DP-T **determines per-segment target track based on data budget.**

Algorithm 2: DP-T: Determine per-segment target track.

```
Input: per-segment size, data budget $D$, segments that are
  classified as Q1, Q2, Q3 or Q4 beforehand
Output: per-segment target track
$\ell = 1$
while $(\sum_i s_{i,\ell} < D)$ and $(\sum_i s_{i+1,\ell} > D)$ do
    $\ell = \ell + 1$
end
$\ell^* = \ell$
Set target track for each segment to $\ell^*$
if $D - \sum_i s_{i,\ell^*} > 0$ then
    Increment the target levels of Q4 segments in increasing
      segment index order
    Let $I$ denote the corresponding increment in data usage;
      update $I$ accordingly
    Return $\ell_i^*$ for each segment $i$ when $D - \sum_i s_{i,\ell^*} - I$ is not
      sufficient for increasing the track for the next Q4
      segment
if $D - \sum_i s_{i,\ell^*} - I > 0$ then
    Increment the target level of Q1-Q3 segments in
      increasing segment index order
    Return $\ell_i^*$ for each segment $i$ when the the remaining
      data is not sufficient for increasing the track for the
      next $Q1$-$Q3$ segment
return per-segment target track $\ell_i^*$ for each segment $i$
```

those for the simple scenes (i.e., low-motion or low-complexity scenes). This is based on the findings in existing studies [36, 37]: (i) depending on scene complexity, segments in the same track can have very different qualities, and (ii) even for VBR encoding where complex scenes are encoded with more bits, the segments with complex scenes still tend to have lower quality than those with simple scenes in the same track. Therefore, allocating a higher target track to a segment with complex scenes can help increase the quality of the track version to be closer to the quality of a segment with simple scenes. For VBR encoding, which is the primary focus of this paper since it has many advantages over constant bitrate (CBR) encoding [23] and has been widely adopted by industry [42], DP-T uses the methodology in [36] to identify segments with complex scenes. Specifically, it identifies the segments with sizes falling in the largest quartile of all the segments in the track as those containing complex scenes. These segments are referred to as *Q4 segments* (i.e., segments in the 4th quartile). The rest of the segments, i.e., those falling into the lower three quartiles, referred to as *Q1-Q3 segments*, are identified as the segments containing relatively simpler scenes. Fig. 1 shows that Q4 segments (marked as crosses) indeed tend to have lower quality than Q1-Q3 segments (marked as triangles) in the same track.

DP-T works as follows. Consider n segments and data budget D. The segments have been classified as Q1, Q2, Q3 or Q4 at the beginning of the streaming session based on their sizes. DP-T first finds the highest track level ℓ^* so that the total data usage is no more than D, i.e., $\sum_i s_{i,\ell^*} \le D$, where s_{i,ℓ^*} is the size of the variant of the ith segment at track level ℓ^*. We refer to this track ℓ^* as the *base track*. The target track for all the segments is first set to this base track, and then the target track of each Q4 segment is incremented when possible. Specifically, if $D - \sum_i s_{i,\ell^*} > 0$, i.e., the data budget is not completely exhausted yet, then the difference is used to add one level to the sequence of Q4 segments, in the increasing order of playback position in the video; if the data budget is still not used up after that, then it will be used to increment the level of Q1-Q3 segments, again in the increasing order of playback position. After the above process, let ℓ_i^* denote the target track for segment i. Then ℓ_i^* will be fed to the ABR logic to limit the track selection for segment i.

Fig. 4 illustrates this algorithm. Suppose we have 8 segments, two being Q4 segments, and the others being Q1, Q2, or Q3 segments; each segment has three track levels, marked in green, orange and blue, respectively. DP-T first determines that track 2 is the highest track level so that the corresponding total data usage is below the data budget. It sets the target track for all the segments to 2. Then for the remaining unused data budget, it first increases the track for the two Q4 segments to track 3, and then uses the still remaining data to increase the track level of the first segment (which is a Q2 segment) to 3. The operation of DP-T is summarized in Algorithm 2.

4 PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed DP strategies and their interaction with ABR schemes using extensive simulation. All the simulations are driven by real-world time-varying network bandwidth traces collected from commercial cellular networks, which allows repeatable experiments and evaluation of different schemes under identical settings. Evaluation using our implementation in a real player is deferred to §5. In the following, we first describe the evaluation setup and then the results.

4.1 Evaluation Setup

Methodology. The goal of the evaluation is to understand the performance of combining DP strategies with ABR schemes under diverse settings. The performance will depend on many factors, including the ABR video characteristics (codec, track ladders, encoding bitrate and qualities), network conditions, and data budget. To understand the complex tradeoffs and allow performance comparisons across the different videos and ABR track ladder settings, we use the following methodology: for a given video, we set the data budget and scale the network bandwidth traces relative to a *reference track* from that video's ABR track ladder.

For the reference track, we focus on middle rungs in the track ladder, in our case, tracks 3 (360p) and 4 (480p), which have quality (VMAF) ranging from fair to good. Denote the size of a reference track (including all the segments in the track) of a video as S and the average bitrate of that track as b. Then the data budget is set to $r_d \times S$ and each network bandwidth trace is scaled so that the average bandwidth is $r_n \times b$, where $r_d \geq 1$ is referred to as *data budget scale factor*, and $r_n \geq 1$ is referred to as *network bandwidth scale factor*.

The above evaluation methodology is designed to allow us to easily reason about how the performance depends on the quantitative relation among total data budget, the sizes (and their corresponding qualities) of different tracks, and the available network bandwidth. For example, for a given video, by ranging the data budget around a particular reference track, we can understand how a DP scheme is able to guide ABR rate adaptation within different degrees of tightness of the data budget. Note that the above methodology is mainly for experimental clarity and provides a way to explore the large space of the experiments as described above. Our DP strategies can work with any data budget, and for any network bandwidth conditions.

Videos. We consider four H.264 encoded and three H.265 encoded VBR videos (see Table 1). Three H.264 videos, BBB (Big Buck Bunny), ED (Elephant Dream), and ToS (Tears of Steel), are encoded by YouTube. The fourth H.264 video, AF (American Football), and the three H.265 videos, AF, BB (Basketball) and FH (Forza Horizon), were encoded using `ffmpeg` [12]. Specifically, we used the "three-pass" encoding following the specification by Netflix [10], with CRF values of 28 for H.264 and 31 for H.265 (default settings in `ffmpeg`). All the videos are around 10-minute long, with segment durations either 5 or 5.33 seconds. They represent a wide range of genres (animation, sci-fi movie, sports and racing car game). Two videos (BB and FH) have frame rates of 30 fps; the others have frame rates of 24 fps.

Data budget. As mentioned earlier, the data budget for a video is set relative to a reference track, i.e., track 3 (360p) or 4 (480p) in our case. The data budget scale factor, r_d, is varied in a wide range, from 1.0 to 1.8, to explore different degrees of tightness of the data budget relative to the size of the reference track.

Network traces. We collected a total of 42 hours of network traces over two large commercial LTE networks in the US. Our traces consist of per-second network bandwidth measurements, which are collected on a phone, by recording the throughput of a large file downloaded from a well-provisioned server. They cover a diverse set of scenarios, including different times of day, locations, and movement speeds (stationary, walking, local driving, and highway driving). We divide the collected traces into 15-minute long traces. The evaluation in the rest of the section is performed using 100 randomly chosen traces. As mentioned earlier, for each video, we scale the network traces using either track 3 or 4 of the video as the reference track. Specifically, we vary the network bandwidth scale factor r_n in a wide range, from 1.2 to 4.0, to explore different degrees of tightness of the available network bandwidth relative to the bitrate of the reference track. For a particular network trace and r_n, although the average network bandwidth is higher than the average bitrate of the reference track, the instantaneous network bandwidth can be very low due to the highly dynamic network conditions in cellular networks.

Data planning strategies. We evaluate three strategies, the strawman track-capping strategy, henceforth referred to as DP-Strawman, and the two strategies that we develop, DP-Q and DP-T. For all these three strategies, we evaluate two ways to interact with the ABR logic, i.e., cap-beforehand and cap-afterwards (see §2.4). Both DP-Q and DP-T periodically recalculate the target tracks, and we evaluate two ways to trigger the calculation: (i) after every N segments have been downloaded, where N is varied from 1 to 60, and (ii) after every $\Delta = 20$ seconds.

ABR schemes. We consider two state-of-the-art ABR schemes: (i) *RobustMPC* [43], which is a well-known scheme based on model predictive control, and has been shown to outperform its more aggressive variants, MPC and FastMPC, under more dynamic network bandwidth settings [29], and (ii) *CAVA* [36], which is a more recent control-based scheme that was designed to explicitly take account of the characteristics of VBR videos. For both schemes, we use the harmonic mean of the average download throughout for the past 5 segments as the bandwidth prediction, as it has been shown to be robust to measurement outliers [20, 43]. The maximum client-side buffer size is set to 100 seconds. This is reasonable for VOD streaming, and is consistent with existing practices that set the maximum buffer size to hundreds of seconds [14, 17, 42]. The track for the first segment is selected to be track 1. The player starts the playback when two segments are downloaded into the buffer, following the practice in commercial services [42].

Perceptual quality. DP-Q uses per-segment quality, which is measured using the VMAF [24] phone model, a state-of-the-art perceptual quality metric. The VMAF value of a segment is the mean VMAF value of all the frames in the segment. To calculate the VMAF values for a video, we need a high-quality reference video. For the videos with publicly available high quality raw videos, we use the corresponding raw video as the reference video. For the other videos, we use the highest resolution track (4K resolution) as the reference video.

Performance metrics. We use six metrics: five for measuring different aspects of QoE (four related to quality, measured using VMAF phone model, and one on rebuffering), and one for measuring data usage. (i) *Quality of Q4 segments*: measures the quality of Q4 segments, which contain complex scenes. (ii) *Quality of all segments*: captures the quality of all the segments. (iii) *Low-quality segment percentage*: measures the percentage of the segments that were selected with low quality (specifically with VMAF values below 40 [24]). The reason for using this metric is because human eyes are sensitive to bad quality segments [32]. (iv) *Rebuffering duration*: measures the total rebuffering/stall duration in a streaming session. (v) *Average quality change per segment*: defined as $\sum_i |q_{i+1} - q_i|/n$, where q_i is the quality of the ith segment and n is the number of segments. (vi) *Data usage*: measures the total amount of data downloaded for a streaming session. All metrics are computed with respect to the delivered segments, i.e., those that have been downloaded and played back.

4.2 RobustMPC with Data Budget

We first report the performance of RobustMPC with the three data planning strategies. Unless otherwise stated, we use cap-afterwards; for DP-Q and DP-T, the per-segment target track is calculated after every five segments are downloaded. We evaluate the impact of other options later in this section.

Comparison of strategies. We first use the results for one video to show the main observations, and then show the results for other videos. Fig. 5 shows the performance of RobustMPC combined with the three data planning strategies, along with the original RobustMPC scheme, when using the ED video, reference track of 360p, $r_n = 4.0$, and $r_d = 1.6$. The results are obtained from 100 simulation runs, each using a randomly chosen LTE trace. We make the following three main observations:

• While the overall quality is necessarily reduced when using DP-Q and DP-T to constrain the data usage, the quality degradation is low: the percentage of segments with VMAF values above 70 is 76% and 78% under DP-Q and DP-T, respectively, only slightly lower than that of the original scheme (85%); for Q4 segments, the percentages are comparable to that the original scheme (70% and 66% vs. 68%). More importantly, DP-Q and DP-T help to steer the ABR scheme away from choosing very low-quality segments, leading to similar or even fewer low-quality segments compared to the original scheme (which uses much more data). Specifically, for all segments, the percentages of low quality segments under RobustMPC, RobustMPC+DP-Q, and RobustMPC+DP-T are 6%, 4%, and 4%, respectively; for Q4 segments, the corresponding values are 10%, 6%, and 6%. The comparable or lower percentage of low-quality segments under DP-Q and DP-T is because they guide the ABR scheme to choose fewer segments with very high quality (that tend to have higher bitrate), which helps to maintain a higher player buffer level, thus leading to lower frequency of choosing low-quality segments. Another benefit of the data planning is that the amount of rebuffering is much lower than that of the original scheme (close to 0 vs. 5 seconds of average rebuffering for the original scheme).

• Both DP-Q and DP-T achieve significantly better QoE than DP-Strawman. For Q4 segments, the median qualities under DP-Q and DP-T are 76 and 77, respectively, 11 and 12 VMAF points higher than that under DP-Strawman; for all (Q1-Q4) segments, the corresponding improvements are 5 and 7 VMAF points, respectively. DP-Q and DP-T also outperform DP-Strawman in achieving lower average quality change per segment. The data usage of DP-Q and DP-T is bounded by the specified data budget, while DP-Strawman underuses the data budget. In summary, while DP-Strawman also satisfies the data budget constraint and leads to close to zero rebuffering as DP-Q and DP-T, it leads to significantly more adverse impact on QoE than DP-Q and DP-T.

• DP-T achieves similar performance as DP-Q for all the performance metrics (also see the results in Table 2). This result demonstrates that even when per-segment quality information is not available, a carefully designed DP strategy that only uses coarse-grained track information such as DP-T can achieve most of the gains of data planning that uses quality information.

Results for other videos. The above results are for one video. Table 2 summarizes the results for three H.264 videos (with boldface titles) and three H.265 videos. The results are for three settings: (i) reference track (RT) of track 3 (360p), $r_n = 4.0$, (ii) RT of track 4 (480p), $r_n = 2.0$, and (iii) RT of track 4 (480p), $r_n = 4.0$, listed from the top to the bottom in Table 2. For all the three settings, the data budget scale factor r_d is 1.6. The table shows the five performance metrics across all the simulation runs. In each cell, the three values are for RobustMPC, RobustMPC+DP-Q, and RobustMPC+DP-T, marked in black, green and blue, respectively.

In the two settings with $r_n = 4.0$, we see that both DP-Q and DP-T lead to significantly lower data usage than the original ABR scheme, which uses on average 93% and 90% more data than DP-Q and DP-T respectively. Even with the lower data usage, DP-Q and DP-T still realize acceptable quality delivery: (i) for most of the cases, the median quality of Q4 segments is close or above 80 (considered as good quality [4, 25]); and (ii) the percentage of low-quality segments per session is consistently low, no more than 6% and 7% for DP-Q and DP-T respectively, in the same ballpark as the original ABR scheme.

In the setting with $r_n = 2.0$ (i.e., the average network bandwidth is tighter with respect to the reference track bitrate than when $r_n = 4.0$), perhaps somewhat surprisingly, we see that the quality when using DP-Q and DP-T is close to and sometime even better than that of the original scheme. This is because when the network bandwidth is not very high, DP-Q and DP-T help the ABR scheme to make more judicious use of the bandwidth by avoiding selecting extremely high quality segment variants, and guide the selection towards the overall good QoE (as we shall see in Fig. 7 later). In addition, they also avoid selecting very low quality segments: the percentage of low-quality segments per session is no more than 6% and 7% for DP-Q and DP-T respectively, and even lower than the corresponding value (up to 10%) for the original scheme. In this setting, because of the lower network bandwidth, the data saving compared to the original scheme is lower than that when $r_n = 4.0$.

For all the three settings, the average stall duration when using DP-Q and DP-T is in general lower than that of the original scheme, and sometimes significantly lower (see the first two settings in Table 2). Last, DP-Q and DP-T exhibit much better quality (up to 16 VMAF [24] points higher) for segments with complex scenes than DP-Strawman (not shown in the table), which reinforces the importance of carefully designing DP schemes that consider both the data budget and QoE dimensions, to realize significantly improved QoE beyond DP-Strawman.

We observe that DP-Q and DP-T are able to lead to low percentage of poor quality segments even with tighter data budget constraints. For instance, for the three settings in Table 2, when we reduce the data budget scale factor r_d from 1.6 to 1.2, the percentage of low-quality segments of DP-Q and DP-T is still within 5% of what the original scheme achieves with no data budget constraint. This behavior is because their planning aspects allow them to better

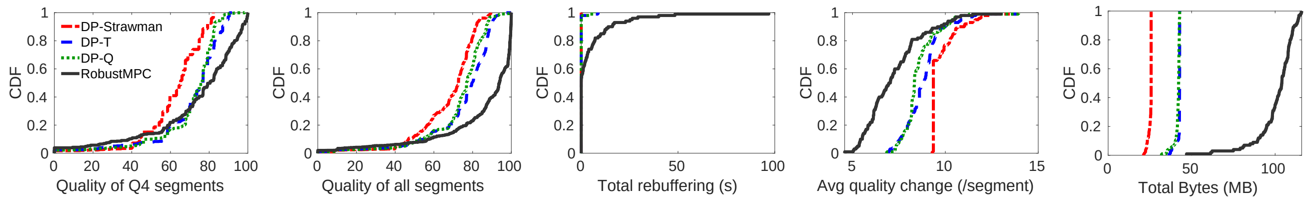


Figure 5: Performance of using data planning with RobustMPC for ED video, reference track 3 (360p), r_n = 4.0, r_d = 1.6.

Table 2: Performance comparison. The three numbers in each cell represent the results for the tuple (RobustMPC, RobustMPC+DP-Q, RobustMPC+DP-T). Reference track (RT) is 3 or 4, r_n = 2.0 or 4.0, r_d = 1.6.

	Video	Median Q4 seg. quality	Low-qual. segs (%)	Stall durat. (s)	Avg. quality change	Data usage (MB)
RT 3, r_n = 4.0	**BBB**	86, 82, 82	7, 4, 4	6, 0, 0	7, 9, 10	78, 32, 33
	ED	79, 76, 77	6, 4, 4	5, 0, 0	7, 9, 9	99, 41, 42
	ToS	82, 74, 76	7, 6, 7	7, 1, 0	8, 10, 10	101, 41, 41
	AF	98, 85, 87	3, 3, 3	1, 0, 0	5, 7, 7	178, 105, 106
	BB	98, 79, 79	5, 4, 4	1, 0, 0	4, 6, 6	162, 78, 78
	FH	92, 72, 76	6, 4, 4	4, 0, 0	5, 6, 6	188, 84, 84
RT 4, r_n = 2.0	**BBB**	89, 92, 92	6, 5, 5	4, 1, 2	7, 6, 6	84, 67, 69
	ED	82, 88, 88	6, 4, 4	5, 1, 1	7, 6, 6	106, 83, 87
	ToS	88, 90, 89	7, 4, 4	5, 1, 1	7, 5, 6	111, 87, 91
	AF	92, 92, 92	7, 5, 5	4, 1, 2	7, 6, 7	150, 137, 140
	BB	93, 92, 92	9, 6, 7	4, 2, 2	6, 5, 5	134, 112, 113
	FH	84, 84, 84	10, 6, 6	6, 1, 1	7, 5, 5	153, 124, 125
RT 4, r_n = 4.0	**BBB**	99, 92, 92	3, 2, 2	0, 0, 0	4, 4, 4	121, 72, 72
	ED	97, 89, 88	2, 2, 2	1, 0, 0	4, 5, 5	162, 89, 90
	ToS	98, 91, 90	3, 2, 2	1, 0, 0	3, 4, 4	161, 92, 94
	AF	98, 93, 96	2, 2, 2	0, 0, 0	4, 5, 6	187, 145, 147
	BB	99, 92, 92	3, 2, 2	0, 0, 0	3, 4, 4	176, 116, 116
	FH	96, 84, 84	3, 2, 2	1, 0, 0	3, 4, 4	225, 129, 129

ration the data budget across the session as well as make better use of network bandwidth variability.

Impact of data budget. For all the three data planning strategies, as expected, increasing the data budget under the same network setting leads to higher quality. Fig. 6 shows the impact of data budget when the reference track is 4 (480p) and network bandwidth scale factor r_n = 4.0 under DP-T; the results under DP-Q show similar trend. The data budget scale factor r_d is varied from 1.0 to 1.8. For clarity, only the results for r_d = 1.0 and 1.6 are shown in the figure; the results for r_d = 1.2 are between those of r_d = 1.0 and 1.6; the results for r_d = 1.8 are close to those of r_d = 1.6. For comparison, the results for the original scheme without data constraint are also shown in the figure. When r_d = 1.6, the quality under RobustMPC+DP-T is closer to that of the original scheme compared to the case when r_d = 1.0, while still using much less data than the original scheme. Even when r_d = 1.0, the percentage of low-quality segments under RobustMPC+DP-T is already similar to that of the original scheme with no data budget constraint.

Fig. 6 also shows the results for DP-Strawman with r_d = 1.0. When r_d = 1.0, although DP-Strawman and DP-T set the same target track for all the segments at the beginning of the session, DP-T leads to significantly better performance than DP-Strawman. This is because during the session DP-T dynamically adjusts the target tracks for the remaining segments based on the remaining data budget, which takes account of the past network dynamics, while DP-Strawman makes no such subsequent adjustment.

Impact of network bandwidth. We now fix the data budget, and investigate the performance of DP-Q and DP-T when increasing the available network bandwidth. As the network bandwidth increases, he data budget becomes the dominant constraint in limiting the track selection, and the data usage becomes closer to the specified data budget. Fig. 7 shows an example when the reference track is 360p, data budget is fixed to r_d = 1.6, and the network bandwidth scale factor r_n increases from 1.5 to 4.0. For clarity, only the results for DP-T and the original scheme when r_n = 2.0 and 4.0 are shown in the figure; the results for DP-Q are similar as those for DP-T. As expected, for both RobustMPC and RobustMPC+DP-T, all the performance metrics improve when increasing r_n from 2.0 to 4.0. When r_n = 2.0, RobustMPC+DP-T consumes less data, has much lower rebuffering and is much better in steering away from the low quality segments than RobustMPC (comparing the two dashed lines in each figure). When the network bandwidth is higher (r_n = 4.0), the data saving from RobustMPC+DP-T is much more significant, while still achieving acceptable quality and significantly lower rebuffering compared to RobustMPC.

Impact of data planning interval. In all the results presented so far, the planning interval for DP-Q and DP-T is set to five segments (using a planning interval of 20 seconds leads to similar results). To evaluate the impact of planning interval, we set it to 1, 5, 20, or 60 segments. A very large planning interval can lead to delayed updates of the target tracks for the remaining segments and slower adjustment to network bandwidth dynamics, which can lead to lower quality and lower data usage. When the planning interval is very small (being 1 segment), we observe similar quality as that when it is set to 5 or 20 segments, but slightly larger quality changes per segment in some settings (since the target tracks for the remaining segments may change frequently). Fig. 8 shows an example for reference track 3 (360p), r_n = 4.0, r_d = 1.6. The difference across different planning intervals is small since the network bandwidth is high.

Fig. 8 also plots the results for DP-Strawman. We see that DP-T significantly outperforms DP-Strawman even with a planning interval of 60 segments (DP-T only does two plannings in this case). This is because DP-T determines the target track for each segment individually, while DP-Strawman chooses the same target track for all the segments.

Interaction with ABR logic. So far, all the results are for cap-afterward. In cap-beforehand, the search space of the ABR scheme

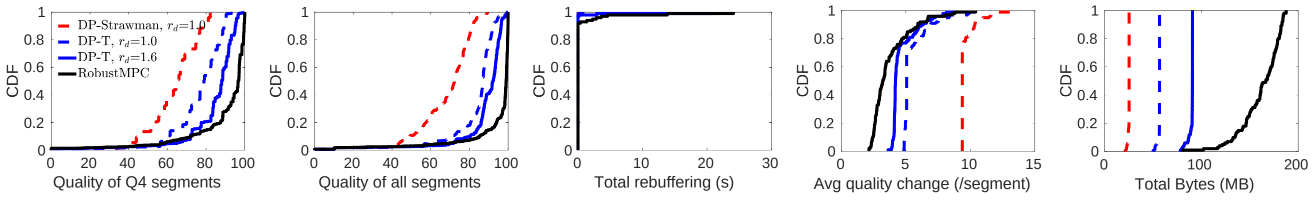


Figure 6: Impact of data budget when using DP-T **with RobustMPC for ED video, reference track 4 (480p),** $r_n = 4.0$.

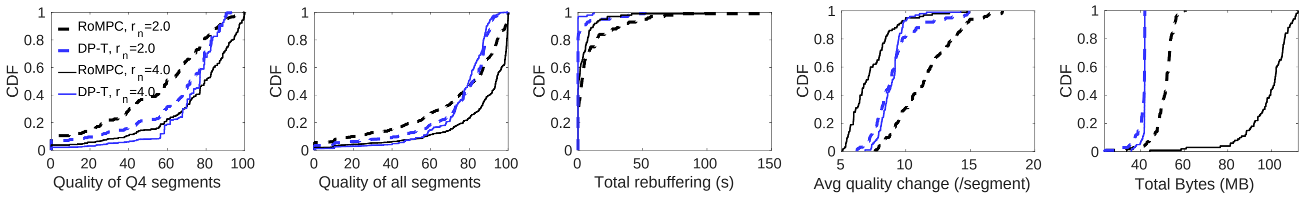


Figure 7: Impact of network bandwidth when using DP-T **with RobustMPC for ED video, reference track 3 (360p),** $r_d = 1.6$. **The two dashed lines are for** $r_n = 2.0$ **and the two solid lines are for** $r_n = 4.0$.

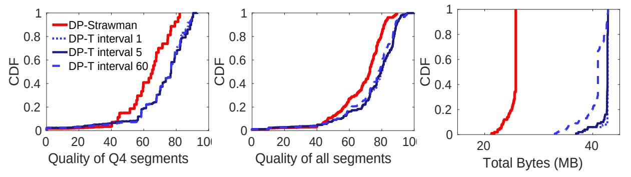


Figure 8: Impact of data planning interval for ED video, reference track 3 (360p), $r_n = 4.0$, $r_d = 1.6$.

is restricted to a smaller set, which can lead to lower running time. On the other hand, the restricted search space may lead to inferior track selection. We see that when combining DP-Q or DP-T with RobustMPC, the performance of cap-beforehand is worse than that of cap-afterward in certain settings. For CAVA, the results in these two cases are similar. The different observations across these two ABR schemes are due to their different design: when deciding the track for the current segment, both schemes consider several segments in the future, CAVA requires that the track levels for these segments to be the same (for smoothness), while RobustMPC does not have such a requirement. In general, when combining DP-Q or DP-T with a particular ABR scheme, the impact of cap-beforehand and cap-afterward should be tested to decide what to use. Usually, cap-afterward is more desirable as it leads to easier deployment with existing ABR logic.

4.3 CAVA with Data Budget

Fig. 9 plots the results for CAVA with DP-Strawman, DP-T, and DP-Q, along with the original scheme, for ED video, reference track of 360p, $r_n = 4.0$, and $r_d = 1.6$. Table 3 summarizes the performance for 6 videos under the same settings as those in Table 2. Across all the cases, the predominant trend is that, for a given setting, the performance of the CAVA based schemes (standalone and with data planning) is somewhat better than the corresponding RobustMPC based schemes. This is not unexpected given the very different designs of RobustMPC and CAVA. On the other hand, the high-level trends observed here are similar to those for RobustMPC:

Table 3: Performance comparison. The three numbers in each cell represent the results for the tuple (CAVA, CAVA+DP-Q**, CAVA+**DP-T**). Reference track (RT) is 3 or 4,** $r_n = 2.0$ **or** 4.0, $r_d = 1.6$.

	Video	Median Q4 seg. quality	Low-qual. segs (%)	Stall durat. (s)	Avg. quality change	Data usage (MB)
RT 3, r_n = 4.0	**BBB**	92, 83, 85	3, 2, 2	2, 0, 1	4, 5, 6	71, 33, 33
	ED	85, 77, 77	2, 2, 2	2, 0, 0	5, 6, 6	90, 42, 42
	ToS	87, 76, 79	5, 3, 4	1, 0, 0	5, 6, 7	91, 42, 42
	AF	99, 88, 94	3, 2, 2	0, 0, 0	4, 5, 7	174, 106, 106
	BB	98, 79, 86	5, 4, 4	2, 0, 0	4, 5, 6	154, 78, 78
	FH	94, 73, 77	4, 2, 2	1, 0, 0	4, 4, 4	179, 85, 85
RT 4, r_n = 2.0	**BBB**	93, 92, 93	3, 3, 3	2, 2, 2	4, 4, 4	77, 67, 68
	ED	88, 88, 87	2, 2, 2	2, 1, 2	4, 4, 4	97, 81, 84
	ToS	90, 90, 90	4, 3, 3	1, 1, 1	5, 4, 5	100, 86, 88
	AF	96, 96, 97	6, 5, 5	1, 1, 1	6, 6, 6	141, 134, 136
	BB	94, 92, 93	8, 7, 7	2, 2, 2	5, 5, 5	126, 111, 112
	FH	90, 85, 89	7, 5, 6	2, 1, 1	5, 4, 4	145, 124, 126
RT 4, r_n = 4.0	**BBB**	99, 93, 95	1, 1, 1	1, 0, 0	2, 3, 3	124, 72, 72
	ED	97, 89, 89	0, 0, 0	1, 0, 0	3, 4, 4	156, 89, 90
	ToS	98, 91, 91	2, 1, 1	0, 0, 0	2, 4, 4	157, 93, 94
	AF	99, 97, 98	2, 2, 2	0, 0, 0	3, 4, 5	193, 147, 147
	BB	99, 92, 93	2, 2, 2	1, 0, 0	2, 4, 4	174, 116, 116
	FH	97, 86, 89	1, 1, 1	1, 0, 0	2, 3, 3	224, 130, 131

• Given a data budget, DP-Q and DP-T guide the ABR scheme to steer away from very low-quality segments. Using DP-Q and DP-T leads to acceptable quality, lower rebuffering and much lower data usage compared to the original ABR scheme.

• Both DP-Q and DP-T significantly outperform DP-Strawman.

• Despite lacking per-segment quality information, DP-T achieves similar performance as DP-Q.

5 EXOPLAYER RESULTS

Implementation. We implemented the three data planning strategies, DP-Strawman, DP-Q and DP-T, in ExoPlayer [15] (v2.11.7)

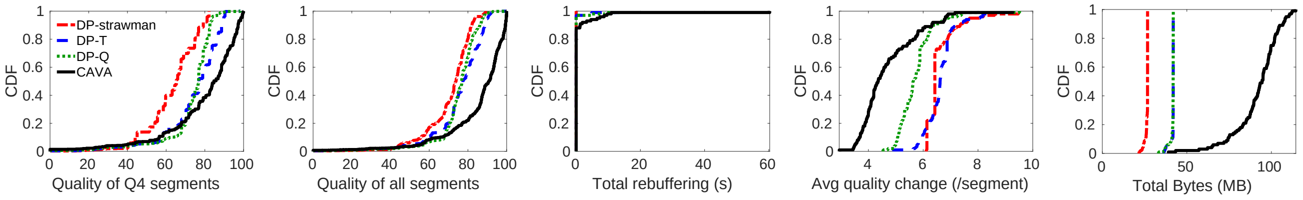


Figure 9: Performance of using data planning with CAVA for ED video, reference track 3 (360p), $r_n = 4.0$, $r_d = 1.6$.

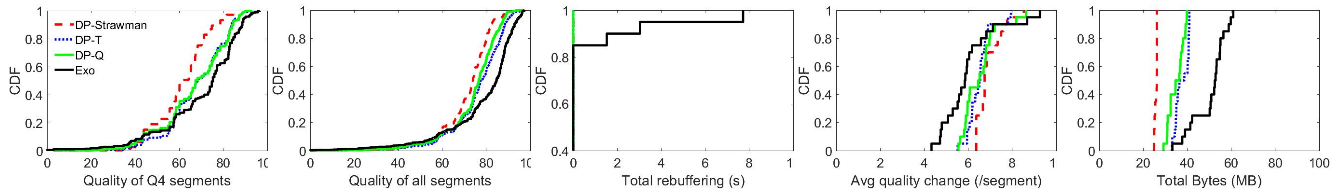


Figure 10: Performance of using data planning in ExoPlayer for ED video, reference track 3 (360p), $r_d = 1.6$, $r_n = 4.0$.

with the DASH [13] protocol. The reason for choosing ExoPlayer is that it is a popular application-level media player for the Android platform and is widely used in commercial services [16]. The implementation involves adding three modules, one for each of the three strategies. For each module, the main changes are in ExoPlayer's `UpdateSelectedTrack` function so that the caps for the tracks are applied after the track selection by the default ExoPlayer logic. We also instrumented ExoPlayer to provide detailed information about the ABR streaming process and its performance. The segment size information is obtained directly from the ABR manifest file received from the server. For DP-Q, we piggybacked the quality information of each segment in the manifest file (see discussion in §2.4).

While Exoplayer for much of its life had avoided the technique of segment replacement (SR) [42], the version that we use (v2.11.7) combines SR with the ABR logic. SR can replace an earlier downloaded segment with a higher quality variant later on (e.g., when the buffer level is high or network bandwidth improves). It essentially attempts to improve QoE at the cost of additional data usage and potential risk of underrunning the buffer during SR (if the network bandwidth drops again). Hence, its design involves complex tradeoffs. Since we are seeking opportunities for achieving good QoE under strict data constraints, we consider using data planning in the absence of SR in this paper, and therefore disable SR in ExoPlayer. A study of DP in the presence of SR is left as future work.

Evaluation setup. We compare the performance of the default ExoPlayer ABR logic in standalone mode, referred to as *Exo*, with Exo+DP-Strawman, Exo+DP-Q, and Exo+DP-T using experiments conducted on an Android phone (Samsung Galaxy S7 Edge, OS 8.0.0). For repeatability and enabling apples-to-apples comparison across the schemes under identical conditions, we deploy our own ABR streaming origin server, and use `tc` [27] to create time-varying network bandwidths (on the path from the server to the client) that are reminiscent of real cellular conditions. Specifically, we use 20 LTE network traces (see §4.1) to drive the experiments. All the experiments use the ED video. Following the same experimental methodology in §4.1, the average network bandwidth and data budget are set based on reference track 3, with data budget scale factor $r_d = 1.6$ and in two network settings, $r_n = 4.0$ and $r_n = 6.0$.

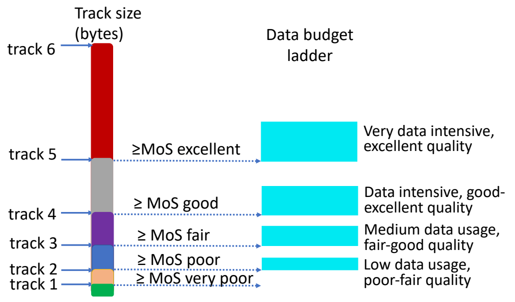

Track size (bytes)
Data budget ladder
track 6
track 5
track 4
track 3
track 2
track 1
≥MoS excellent
≥ MoS good
≥ MoS fair
≥ MoS poor
≥ MoS very poor
Very data intensive, excellent quality
Data intensive, good-excellent quality
Medium data usage, fair-good quality
Low data usage, poor-fair quality

Figure 11: An approach to determining per-session data budget based on data budget ladder.

Evaluation results. Both DP-Q and DP-T are computationally light-weight. For 104 segments, their running times in ExoPlayer on the test phone were 55.2 ms and 48.8 ms, respectively (and expected to be faster for newer devices). Fig. 10 plots the performance of the various schemes when $r_d = 1.6$ and $r_n = 4.0$ (the results for $r_n = 6.0$ show similar trends). Note that the data usage reported here only considers application-level data, and is slightly lower than the actual data usage (our measurements show that the total network protocol related overhead is within 3%). The overall observations are consistent with the results from the simulations. Exo in its standalone mode has the highest quality, while leads to the highest data usage and the highest amount of stalls. Combining DP-Q and DP-T with Exo leads to significantly lower rebuffering and data usage (bounded by the data budget), with little degradation in quality. Combining DP-Strawman and Exo leads to the lowest data usage, however, with noticeable degradation in quality.

6 OTHER PRACTICAL ASPECTS

As shown in our evaluation, the actual impact of data planning on QoE depends on the video track ladder, ABR logic, data budget, and network conditions. Based on insights from our evaluation, we next discuss two other practical aspects related to data planning.

Setting per-session data budget. Following §2.1, we next present an approach for determining per-session data budget choices for a specific video. This approach considers the track ladder of the

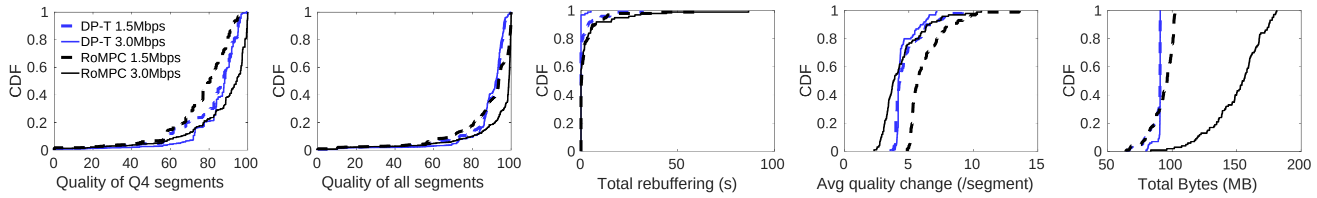


Figure 12: Impact of network-based bandwidth cap when using and not using data planning strategies, for ED video, reference track 4 (480p), $r_n = 4.0$, $r_d = 1.6$. The two dashed lines are for cap 1.5 Mbps and the two solid lines are for cap 3.0 Mbps.

specific video, as illustrated by an example in Fig. 11. As in existing practice in commercial services, it first roughly maps tracks to quality levels, e.g., mapping track 1 to very poor quality, track 2 to poor quality, track 3 to fair quality, etc. But instead of just statically capping the track selection to a particular level as in existing practice, our approach (i) determines a *data budget ladder* based on the mapping, and (ii) uses dynamic data planning strategies such as DP-Q or DP-T to judiciously guide rate adaptation. Specifically, the data budget ladder is first determined as illustrated in Fig. 11. The lowest rung in the data budget ladder (e.g., low data usage, poor-fair quality) maps to data usage between track 2 and 3 for this particular video, i.e., the data budget is set to $D = r_d \times S_2$, where $r_d \geq 1$ and S_2 is the size of track 2. Similarly, the second lowest rung in the data budget ladder (e.g., medium data usage, fair-good quality) maps to data usage between track 3 and track 4 with the data budget set to $D = r_d \times S_3$. The other rungs in the data budget ladder can be defined in similar ways. In the above, the parameter r_d allows a finer-level decision of data budgets within the rung (e.g., based on the remaining monthly data plan, the content of the video, etc.). As we have shown in §4, a larger r_d leads to better performance at the cost of higher data usage. On the other hand, even when $r_d = 1.0$, our proposed strategies, DP-Q and DP-T, can significantly outperform DP-Strawman. We emphasize that the specific numbers in the above only serve as illustrative examples; the actual mapping may be different for another video depending on its ABR track ladder, but the basic concept still holds.

Setting network bandwidth cap. Recall that one practice in industry for limiting data usage is to cap the network bandwidth to a fixed value for video streaming sessions (see §1). This approach can be potentially combined with client-based DP strategies to both control the peak network bandwidth usage as well as per-session data usage, while still delivering good QoE. As a first step, we explore how to configure the network bandwidth cap in conjunction with our DP strategies. Our evaluation shows that, for a given data budget, when combining a DP strategy with the ABR logic, as the network bandwidth increases, the data usage will still be bounded by the data budget, while the quality improves and rebuffering decreases (see Fig. 7). This trend implies that when a cellular network provider caps the network bandwidth, from both the data usage and QoE perspectives, it is more desirable to cap the bandwidth to a higher level and rely on the client-side data budget constraint to limit the data usage. We next demonstrate this point using an example in Fig. 12. The figure shows the results for four settings: RobustMPC and RobustMPC+DP-T, with either 1.5 or 3.0 Mbps network bandwidth cap. Without DP-T, the standalone ABR scheme with the lower 1.5 Mbps network cap uses much less data than that when the network cap is 3.0 Mbps. This benefit, however, comes at the cost of much worse quality. When combined with DP-T (data budget $r_d = 1.6$, reference track of 480p), the ABR scheme with 3.0 Mbps cap leads to similar data usage as the two cases with 1.5 Mbps cap, while achieving visibly better quality. Overall, RobustMPC+DP-T with 3.0 Mbps cap achieves the best balance in QoE and data usage, demonstrating the benefits of combining DP with a higher network bandwidth cap. Note that these are illustrative settings; determining the optimal network bandwidth cap to deploy in practice requires detailed analysis.

7 RELATED WORK

ABR schemes. A large number of ABR schemes have been proposed. BBA [17], BOLA [39], and BOLA-E [38] propose adaptation schemes based on client-side buffer information. QDASH [30] tries to reduce quality switches during adaptation. PANDA/CQ [26] jointly considers network bandwidth and video bitrate variability. Pensieve [29] proposes a system that generates ABR algorithms using reinforcement learning. Oboe [1] pre-computes the best possible ABR parameters for different network conditions and dynamically adapts the parameters at run-time. All of these schemes aim to maximize QoE without considering data usage at all.

Reducing data usage. Reducing data in the content encoding process has been studied in [5, 8, 21, 31, 41]. Our work considers reducing data usage for a given set of encoded tracks and can complement those efforts. QBR [7] aims to improve the efficiency of existing ABR schemes by reducing the data usage while potentially increasing QoE. The study in [6] manages the tradeoff between monthly data usage and video quality by leveraging the compressibility of videos and predicting consumer usage behavior throughout a billing cycle. Neither of the above two studies uses data budget to explicitly constrain the data usage. The work in [37] relies on a user specified target quality to reduce data usage, which does not provide users explicit control of data usage (see §2.1). In addition, it assumes that per-segment quality is available, while we also consider the case when quality information is not available.

8 CONCLUSION

We proposed a novel framework for ABR streaming using a per-session data budget constraint, and developed two strategies, DP-Q and DP-T. This framework is designed to work in conjunction with existing ABR workflows. Using a combination of extensive simulations and experiments with ExoPlayer, we demonstrated that a per-session data budget together with a carefully designed data planning strategy can achieve significantly lower data usage, while still delivering good QoE. The results are encouraging and suggest the potential of further research in this direction.

REFERENCES

[1] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-tuning Video ABR Algorithms to Network Conditions. In *SIGCOMM*.

[2] Apple. [n. d.]. AVFoundation. https://developer.apple.com/av-foundation/. ([n. d.]).

[3] AT&T. 2018. Stream Saver. (2018). https://www.att.com/offers/streamsaver.html

[4] Christos Bampis. 2018. Measuring Video Quality with VMAF: Why You Should Care. (2018). http://downloads.aomedia.org/assets/pdf/symposium-2019/slides/ChristosBampis_Netflix.pdf

[5] Chao Chen, Yao-Chung Lin, Anil Kokaram, and Steve Benting. 2017. Encoding Bitrate Optimization Using Playback Statistics for HTTP-based Adaptive Video Streaming. *arXiv preprint arXiv:1709.08763* (2017).

[6] Jiasi Chen, Amitabha Ghosh, Josphat Magutt, and Mung Chiang. 2012. QAVA: Quota Aware Video Adaptation. In *Proc. of ACM CoNEXT*. 121–132.

[7] William Cooper, Sue Farrell, and Kumar Subramanian. 2017. QBR Metadata to Improve Streaming Efficiency and Quality. In *SMPTE*.

[8] Jan De Cock and Anne Aaron. 2016. Constant-slope rate allocation for distributed real-world encoding. In *Picture Coding Symposium (PCS), 2016*. IEEE, 1–5.

[9] Jan De Cock, Zhi Li, Megha Manohara, and Anne Aaron. 2016. Complexity-based consistent-quality encoding in the cloud. In *ICIP*. IEEE.

[10] Jan De Cock, Aditya Mavlankar, Anush Moorthy, and Anne Aaron. 2016. A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications. In *SPIE, Applications of Digital Image Processing*.

[11] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the impact of video quality on user engagement. In *ACM SIGCOMM*.

[12] FFmpeg. 2017. FFmpeg Project. https://www.ffmpeg.org/. (2017).

[13] International Organization for Standardization. 2012. ISO/IEC DIS 23009-1.2 Dynamic adaptive streaming over HTTP (DASH). (2012).

[14] DASH Industry Forum. 2017. Reference Client 2.4.1. https://goo.gl/XJcciV. (2017).

[15] Google. 2016. ExoPlayer. https://github.com/google/ExoPlayer. (2016).

[16] Google. 2017. ExoPlayer: Flexible Media Playback for Android (Google I/O '17). https://youtu.be/jAZn-J1I8Eg?t=552. (2017).

[17] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc. of ACM SIGCOMM*.

[18] MulticoreWare Inc. 2018. H.265 Video Codec. http://x265.org/hevc-h265/. (2018).

[19] ITU. 2017. H.264 codec. https://goo.gl/AjvnTs. (2017).

[20] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *CoNEXT*.

[21] Ioannis Katsavounidis. 2018. Dynamic optimizer a perceptual video encoding optimization framework. https://goo.gl/zHdium. (2018).

[22] S. Shunmuga Krishnan and Ramesh K Sitaraman. 2013. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking* 21, 6 (2013), 2001–2014.

[23] TV Lakshman, Antonio Ortega, and Amy R Reibman. 1998. VBR video: Tradeoffs and potentials. *Proc. IEEE* (1998).

[24] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward A Practical Perceptual Video Quality Metric. (2016). https://goo.gl/ptjrWv.

[25] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, and Jan De Cock. 2018. VMAF: The Journey Continues. (2018). https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12

[26] Zhi Li, Ali Begen, Joshua Gahm, Yufeng Shan, Bruce Osler, and David Oran. 2014. Streaming video over HTTP with consistent quality. In *ACM MMSys*.

[27] Linux. 2014. tc. https://linux.die.net/man/8/tc. (2014).

[28] Yao Liu, Sujit Dey, Fatih Ulupinar, Michael Luby, and Yinan Mao. 2015. Deriving and Validating User Experience Model for DASH Video Streaming. *IEEE Transactions on Broadcasting* 61, 4 (December 2015).

[29] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proc. of ACM SIGCOMM*.

[30] Ricky KP Mok, Xiapu Luo, Edmond WW Chan, and Rocky KC Chang. 2012. QDASH: a QoE-aware DASH system. In *ACM MMSys*.

[31] Netflix. 2015. Per-Title Encode Optimization. https://goo.gl/1J5vBv. (2015).

[32] Netflix. 2016. VMAF score aggregation. https://goo.gl/v38JMB. (2016).

[33] Pengpeng Ni, Ragnhild Eg, Alexander Eichhorn, Carsten Griwodz, and Pål Halvorsen. 2011. Flicker effects in adaptive video streaming to handheld devices. In *Proc. of ACM Multimedia*.

[34] Jan Ozer. 2017. Finding the Just Noticeable Difference with Netflix VMAF. https://goo.gl/TGWCGV. (September 2017).

[35] The WebM Project. 2017. VP9 Video Codec. https://goo.gl/Xep8rr. (2017).

[36] Yanyuan Qin, Shuai Hao, Krishna R Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. 2018. ABR streaming of VBR-encoded videos: characterization, challenges, and solutions. In *CoNext*. ACM.

[37] Yanyuan Qin, Shuai Hao, Krishna R Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. 2019. Quality-aware strategies for optimizing ABR video streaming QoE and reducing data usage. In *MMSys*. ACM.

[38] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *MMSys*.

[39] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2016. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. In *INFOCOM*. IEEE.

[40] T-Mobile. 2018. T-Mobile Binge On. (2018). https://goo.gl/Q9fbw6

[41] Laura Toni, Ramon Aparicio, Telecom Bretagne, Karine Pires, Gwendal Simon, Alberto Blanc, and Pascal Frossard. 2015. Optimal selection of adaptive streaming representations. *ACM Trans. Multimedia Comput. Commun. Appl.* (2015).

[42] Shichang Xu, Z. Morley Mao, Subhabrata Sen, and Yunhan Jia. 2017. Dissecting VOD Services for Cellular: Performance, Root Causes and Best Practices. In *IMC*.

[43] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *SIGCOMM*. ACM.