NEWCAST: Joint Resource Management and QoE-driven Optimization for Mobile Video Streaming

Imen Triki, Rachid El-Azouzi and Majed Haddad
LIA/CERI, University of Avignon,
Agroparc, BP 1228, 84911, Avignon, France

***Abstract*—Predicting future throughput in mobile networks becomes more and more possible today thanks to the rich contextual information provided by mobile applications and smartphone sensors. It is even likely that such contextual information, which may include traffic, mobility and radio conditions will lead to a novel agile resource management not yet thought of. In this paper, we propose a framework (called NEWCAST) that anticipates the throughput variations to deliver video streaming content. NEWCAST takes advantage of the capacity prediction in order to better distribute the resources allocated by the scheduler among users over the prediction horizon. This has the advantage of leading towards better user engagement for video streaming users without harming other traffics present in the system. We develop an optimization problem that realizes a fundamental trade-off among critical metrics that impact the user's perceptual quality of experience (QoE) and the cost of system utilization. Both simulated and real-world throughput traces collected from [1], [2] were carried out to evaluate the performance of NEWCAST. It is shown from our numerical results that NEWCAST provides the efficiency that the new 5G architectures require in terms of computational complexity and robustness. We also implement a prototype system of NEWCAST and evaluate it in a real environment with a real player to show its efficiency and scalability in a multi-users scenario compared to baseline adaptive bitrate algorithms.**

***Index Terms*—Adaptive video streaming, quality of experience, resource allocation, mobile network, capacity prediction.**

I. Introduction

Due to the breakthrough evolution of smartphones and their large penetration in daily life, mobile networks have witnessed an unrivaled growth of their mobile traffic posing new challenges to their resource management. The evolution of multimedia services in the Internet and the increasing consumer demand for high definition (HD) contents have even led the operators and the industry to rethink the way networks are dimensioned. According to recent statistics carried out by Cisco [3], 82% of all Internet consumers' traffic will be http video streaming by 2021, which explains the huge amount of care being accorded to video streaming services.

In the literature, many studies were carried out to identify the critical metrics that may impact the user's perceptual QoE [4]–[8]. To estimate QoE, researchers have also developed model using data from extensive subjective quality assessment tests [5], [9], [10]. One of the key factors that may reflect the users' experience is user engagement. Authors in [11] quantified the user engagement and identified some critical metrics that may affect it such as the buffering ratio, the rate of buffering, the start-up delay, the rendering quality, and the average bitrate. It was revealed through [11] that the rebuffering events have a significant impact on the QoE in the sense that the time spent on rebuffering during a video session can significantly reduce user engagement. One other aspect that may impact user engagement is the temporal variations of the video quality. Indeed, authors in [12] claimed that temporal variability in quality can be considered as worse as a constant quality with a lower average bitrate. Additional empirical results in [13] showed that humans appear to be more forgiving on buffer stalls than they are on video quality variations. Long buffer freezing events are even not rated worse than short buffer freezing towards high video quality levels.

To improve user engagement in real time, DASH (Dynamic Adaptive Streaming over HTTP) appeared as an emerging standard for video content delivery [14]. Various commercial solutions adopting DASH have been proposed to improve the user's QoE such as Microsoft's smooth streaming, Adobe's HTTP dynamic streaming and Apple's live streaming. In DASH, each video file is divided into multiple small segments encoded at multiple quality levels, and it is up to the client to chose the most suitable quality level (bitrate) to stream the future segment. In the literature, adaptive bitrate algorithms are classified in three main classes: buffer-based [15], throughput-based [16] and buffer–throughput-based algorithms [17]. While the first class makes the decision based on the playback buffer occupancy state, the second class exploits the historical TCP throughput measurements [18] to estimate the current bandwidth and instantaneously adapt the quality. From tests conducted on millions of real users, authors in [15] were able to conclude that a buffer-based approach reduces the rebuffering rate by 20% while providing better video quality. To efficiently ensure network performance along with QoE, several researchers are exploring HTTP Adaptive Streaming (HAS) enabled architectures [8], [19], [20].

Related Work

Within these classes, many adaptive strategies were proposed to reduce the interruption of the playback buffer. In [17], authors proposed a predictive control algorithm that combines

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

throughput and buffer occupancy information. [21] developed a suite of techniques that guide the trade-offs between stability, fairness, and efficiency leading to a general framework for robust video adaptation. In [22], authors addressed the resource management issue in DASH QoE provisioning while considering user preferences on rebuffering and cost of video delivery. Several works have also explored cross-layer bandwidth allocation schemes to improve the QoE of adaptive video streams. Cross-layer allocation schemes that factor the channel quality, video quality requirements and encoding rate fluctuations of HAS video streams with the goal of minimizing the transmission delays experienced by users were proposed in [23], [24]. The authors in [25] propose a cross-layer scheme to optimize the total utility of all clients while maintaining stable video quality and supporting user and device-specific needs. AVIS presented in [26] is yet another cross-layer scheme that can separate resource management of adaptive video flows from regular video flows. In [27], authors investigated the buffer-based selection problem and formulated the problem as a stochastic optimization problem with an objective to maximize the QoE metrics. Their solution outperforms other alternative solutions such as FESTIVE [28], prediction method in [16], but they obtained low improvement compared to buffer based in [15].

We have compared NEWCAST with a recent approach developed in [15], [27]. Performance results have been obtained through real-world LTE traces from the University of Ghent. Our algorithm can be considered as a pure buffer-based algorithm as the capacity estimation is unnecessary in the steady state. Authors compared their approach with Netflix default algorithm, and they reduced the buffer rate by 10-20% while delivering a similar average video quality and a higher video quality in steady state. Now, if we ignore the cost of delivering, NEWCAST reduces the bitrate switching from 13 for throughput-based (TB) algorithms to 2, and from 27 for buffer-based (BB) algorithms in [27] to 2, which is very well appreciated for the users' perceptions. A more detailed analysis is presented in Section VI-E.

Although there is a rich literature on methods used for optimizing the QoE in video streaming services, very few papers exploited the knowledge of future throughput variations for quality adaptation. The main idea of this paper is inspired from [29], where authors designed a QoE-driven optimization framework that exploits the knowledge of future throughput variations to minimize the system utilization cost while avoiding rebuffering events. The main shortcoming of their approach is that it is only suited for classical video streaming as it ignores important visual quality metrics related to adaptive streaming.

In [30], [31], authors design a low-latency prediction based bitrate adaptation scheme over wireless access, which leverages TCP throughput predictions on multiple time scales (i.e., 1 to 10 seconds). They proposed several prediction methods in order to maximize the average video quality under some constraints on target latency, number of quality and number of playback interruptions. These techniques may result in some performance problems when multiple adaptive video streaming share a wireless link. These problems manifest as large number of switching rate and inefficient utilisation of the wireless link. Indeed, these frameworks ignored the system utilisation and how knowledge of future capacity variations could be used towards reducing system utilisation while maximizing the QoE [26]. NEWCAST exploits both future temporal and multi-user diversity to reduce the congestion in the wireless link and to maintain high QoE in terms of average video quality, number of quality and number of playback interruptions. The level of congestion is modeled here through the capacity that a user can get as function of the time. Small capacity reflects high congestion in the wireless link shared by several users.

Since video streaming is very bandwidth consuming, its delivery cost became too high for operators to support the increasing bandwidth demand with the arrival of ultra high definition (UHD) video quality, which requires 16 times more pixels than full HD. However, it is important to develop solutions taking into account the delivery cost as well as the QoE through different metrics like rebuffering, average quality and switching in quality levels. In this paper, we design a QoE-driven optimization framework that realizes the trade-off between bandwidth utilization cost and content resolution under constraints on rebuffering events. It extends the model developed in [29] by considering adaptive video streaming.

Summary of Contributions

We summarize our main contributions as follows:

- We provide a general optimization framework for stored video delivery that accounts for heterogeneous client preferences, QoE models and capacity variations,
- Under the constraint of no rebuffering events, we formally obtain an optimal solution where the transmission schedule is of a threshold type and the bitrate distribution is of an ascending order,
- We propose an efficient heuristic, which we call NEWCAST, that performs close to the optimal approach. NEWCAST performances are evaluated through simulations under the constraint of no rebuffering events[1],
- We study the characteristics of NEWCAST in terms of robustness (using real traces) and complexity. We then compare it to baseline adaptive bitrate algorithms,
- We conduct extensive simulations to evaluate the proposed solution. Experimental results indicate that NEWCAST achieves a good trade-off between the cost of the delivery and QoE metrics. Noticeably, we also observe that NEWCAST can be stabilized effectively and video bitrate switches occur rarely (at the maximum number of bit-rate levels). The freezing events can be almost completely avoided if the predictive capacity can download a video with lower quality. Our solution clearly outperforms the solution proposed in [27],
- We implement NEWCAST in a real environment and adapt it for real interactions with a real DASH player. We present a detailed design of NEWCAST at client side and how it interacts with the base station in a multi-users scenario. This allows NEWCAST to work with

[1] Due to the lack of space, we moved the results where we tolerate buffer stall during the video duration in [32].

any existing bases station scheduler, facilitating simpler deployments.

The rest of the paper is organized as follows: In Section II, we introduce the system model and formulate the optimization problem. In Section III, we discuss the properties of the optimal solution. In Section IV, we propose optimal approaches and heuristic algorithms for the problem resolution with the constraint of no rebuffering events. Then, in Section V, we consider the hypothesis to allow rebuffering events during the streaming session. Section VI is dedicated to both simulations and numerical results and Section VII is dedicated to experiments. We conclude the paper in Section VIII.

II. Problem formulation

We consider a video file stored in a video streaming server and divided into N segments of equal length in second. Each segment is composed of S frames and encoded at L different bitrates $\{b_1, \dots, b_L\}$, such that $b_i < b_j$ for $i < j$. To stream the video, the client requests the segments to the server one by one and indicates at each request the video quality (bitrate) needed for the streaming. Denote by $b(t)$ the video bitrate being streamed at time t, and by $\gamma(t)$ the quotient $\frac{b_L}{b(t)}$ where b_L is the highest video bitrate. We assume that, at the client side, the video frames are played at a rate of λ frames per second (fps), and that, before starting the video, a prefetching stage is introduced till having Q_0 frames in the playback buffer. Thus, Q_0 represents the number of accumulated frames that playout buffer should reach before the media player starts to play. To avoid buffer overflows, we assume that the playback buffer is very large.

In our problem modelling, we exploit the knowledge of the user's future available throughput (hereinafter called network capacity) to optimize the system usage cost and the QoE. Let $c(t)$ be the network future capacity at time t and $r(t)$ be the transmission bitrate of the user at that time, note that $0 \leq r(t) \leq c(t)$. Inspired by [29], we define the system utilization cost as

$$\sigma = \frac{1}{T}\int_0^T \frac{r(t)}{c(t)}dt, \tag{1}$$

where $\frac{r(t)}{c(t)}$ is the proportion of resources allocated to the user at time t (can be interpreted as the proportion of time the user is occupying the network if we use discretize the time), and T defines the video length in second. We compute the number of frames that will be streamed with quality level j during the streaming session as

$$\int_0^T \frac{\delta_{\{b(t)=b_j\}} r(t)\lambda}{b(t)}dt = \int_0^T \frac{\gamma_j(t) r(t)\lambda}{b_L}dt, \tag{2}$$

where

$$\gamma_j(t) = \begin{cases} \gamma(t) & \text{if } b(t) = b_j, \quad j \in [1 \dots L], \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Assume that the user's perception on the video quality levels can be expressed by the mean of weights $\{w_1, \dots, w_L\}$ such that w_j corresponds to quality level j and $w_i < w_j$ for $i < j$. Hence, we define the weighted average quality of the video as

$$\rho = \frac{\sum_{j=1}^{j=L} w_j \int_0^T \gamma_j(t) r(t)\lambda dt}{b_L \times (N \times S)} = \frac{\sum_{j=1}^{j=L} w_j \int_0^T \gamma_j(t) r(t) dt}{S_L}, \tag{4}$$

where S_L represents the video total size in bits when it is coded with the highest bitrate level b_L, i.e., $S_L = \frac{b_L \times N \times S}{\lambda}$.

Normally, a high video quality comes at a high cost. However, it may happen that a user wishes to reduce his cost in return of a low quality, or that an operator wishes to save the network resources for further usage. To cover such situations, we define a positive parameter π to make the tradeoff between system utilization cost and video quality. Therefore, we define our optimization cost function as

$$\mathcal{F} = \sigma - \pi \times \rho.$$

Remark 1. All results obtained in this paper are still available for other types of function $\mathcal{F}$ that satisfy the following assumptions: the objective function $\mathcal{F}$ is increasing in system utilisation and decreasing in average quality of the video.

Let $u(t)$ be the *cumulative* number of arrival frames at time t and $l(t)$ be the *cumulative* number of frames being already played at that time. Therefore, we define the buffer underflow constraint as $u(t) \geq l(t)\ \forall t \leq T$. Given the transmission bitrate $r(t)$ and the corresponding video bitrate $b(t)$, we express the network frame rate as $\frac{\lambda\ r(t)}{b(t)}$.

Denote by (r, γ) the video transmission strategy during the streaming session, where r defines the transmission schedule and γ characterizes the distribution of video bitrates. We start with the case where no rebuffering events will happen during the streaming session. Hence, we summarize our optimization problem, as follows[2]

$$\min_{(r,\gamma)} \mathcal{F}(r,\gamma) = \frac{1}{T}\int_0^T \frac{r(t)}{c(t)}dt - \pi \times \frac{\sum_{j=1}^{j=L} w_j \int_0^T \gamma_j(t) r(t) dt}{S_L}$$
$$s.t \begin{cases} \int_0^t \frac{\lambda\ c(t)\gamma_1}{b_L} \geq l(t), & \forall t \leq T \\ \int_0^t \sum_{j=1}^{j=L} \frac{\lambda\ r(t)\gamma_j(t)}{b_L} \geq l(t), & \forall t \leq T \\ \int_0^T \sum_{j=1}^{j=L} \frac{\lambda\ r(t)\gamma_j(t)}{b_L} = l(T), \end{cases} \tag{5}$$

where the first constraint ensures the existence of at least one solution which corresponds to a mono-quality streaming using the lowest video bitrate and the whole resources. At the end of Section VI-B, we study the case where several rebuffering events are tolerated during the streaming session.

III. Properties of optimal solution without rebuffering events

A. The threshold scheme for transmission schedule

Definition 1. *Giving the network capacity c, we define the threshold transmission schedule by*

$$r_{th}(t) = \begin{cases} c(t) & \textit{if } c(t) \geq \alpha \\ 0 & \textit{otherwise.} \end{cases} \tag{6}$$

Proposition 1. *Assume that there exists a feasible solution that satisfies the constraints in (5), then there exists an optimal strategy $(r_{th}, \gamma_{r_{th}})$ of optimization problem (5), where r_{th} is a threshold transmission schedule.*

[2] We emphasize that all results obtained hereafter can be extended to other functions such as logarithm function. The more important assumption needed for our theoretical results is to assume that the objective function F is increasing in system utilization and decreasing with the weighted average quality of the video.

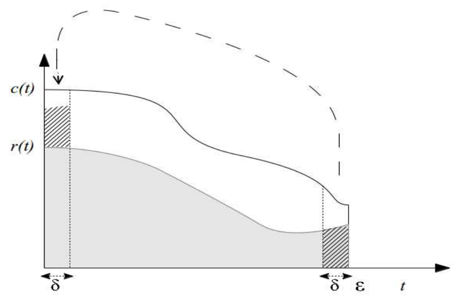

Fig. 1: Sketch of proof of the threshold strategy. Here, the hatched area on the right can be entirely shifted to the left, which gives a value of β equal to 1.

This propriety was actually inspired by [29]. Nevertheless, authors in [29] assumed a classical video streaming with only one bitrate level, whereas we consider adaptive video streaming with multiple bitrate levels, which makes our optimization problem more appealing as it fits current video streaming schemes.

Proof. Let c and r be the network capacity and the user transmission bitrate on a given interval of time $[0,\epsilon]$. Without loss of generality [3] and for the sake of illustration, we choose an interval of time where c is monotonically decreasing as shown in Fig. 1. As we have $r(t) \leq c(t)\ \forall\ t \in [0,\epsilon]$, then $\exists\ (\delta,\beta) \in [0,\frac{\epsilon}{2}] \times [0,1]$ such that $\forall\ t \in [0,\delta]$

$$c(t) \geq c(t+\epsilon-\delta) \tag{7}$$

and

$$\int_0^{\delta} \frac{r(t)+\beta r(t+\epsilon-\delta)}{c(t)}dt \leq \delta, \tag{8}$$

where Inequality (7) derives from the decreasing pace of c, and Inequality (8) derives from the fact that some data at the end can be transmitted beforehand. On the other hand, we have

$$\int_0^{\epsilon} \frac{r(t)}{c(t)}dt = \int_0^{\delta} \frac{r(t)+\beta r(t+\epsilon-\delta)}{c(t)}dt + \int_{\delta}^{\epsilon-\delta} \frac{r(t)}{c(t)}dt + \int_{\epsilon-\delta}^{\epsilon} \frac{r(t)}{c(t)}dt - \int_0^{\delta} \frac{\beta r(t+\epsilon-\delta)}{c(t)}dt. \tag{9}$$

Using Inequality (7), we obtain

$$\int_0^{\epsilon} \frac{r(t)}{c(t)}dt \geq \int_0^{\delta} \frac{r(t)+\beta r(t+\epsilon-\delta)}{c(t)}dt + \int_{\delta}^{\epsilon-\delta} \frac{r(t)}{c(t)}dt + \int_{\epsilon-\delta}^{\epsilon} \frac{r(t)}{c(t)}dt) - \int_{\epsilon-\delta}^{\delta} \frac{\beta r(t)}{c(t)}dt. \tag{10}$$

Obviously, if

$$\int_0^{\delta} \frac{r(t)+\beta r(t+\epsilon-\delta)}{c(t)}dt = \delta,$$

then all the given capacities in $[0,\delta]$ will be used, i.e., all the white surface in Fig. 1 will be filled. In that case, we define a new transmission schedule r' such that

$$r'(t) = \begin{cases} c(t) & \text{t} \in [0,\delta] \\ r(t) & \text{t} \in]\delta,\epsilon-\delta[\\ (1-\beta)r(t) & \text{t} \in [\epsilon-\delta,\epsilon] \end{cases} \tag{11}$$

which gives

$$\int_0^{\epsilon} \frac{r(t)}{c(t)}dt \geq \int_0^{\epsilon} \frac{r'(t)}{c(t)}dt$$

Otherwise, if

$$\int_0^{\delta} \frac{r(t)+\beta r(t+\epsilon-\delta)}{c(t)}dt < \delta, \tag{12}$$

then β will be equal to 1 since our objective is to shift as much data as possible from the times where the capacity is low to the times where the capacity is high. Therefore, to completely use the highest capacities, we must repeat the same shifting operation on $[0,\epsilon-\delta]$ considering a new transmission function r' verifying

$$\begin{cases} \displaystyle\int_0^{\delta} \frac{r'(t)}{c(t)}dt = \int_0^{\delta} \frac{r(t)+\beta r(t+\epsilon-\delta)}{c(t)}dt \\ r'(t) = r(t)\ \forall\ t \in [\delta,\epsilon-\delta]. \end{cases} \tag{13}$$

In both cases, Inequality (10) holds, which means that the highest capacities are less expensive than the lowest capacities in terms of network utilization cost if they were used for transmitting data. If we keep repeating the shifting operation on all the future horizon, we end up having all the highest capacities entirely used and all the lowest one unused, which is clearly a threshold transmission schedule as defined in Definition 1.

Now, we assume that, knowing c, there exists a feasible solution (r,γ) that satisfies the constraints in (5). To perform the data shifting operation on the transmission schedule, three main conditions should be verified: (i) The shifted data must have the same video bitrate as the bitrate used in the shifted-to time, (ii) data shifting shall not interrupt a segment transmission schedule, (iii) data shifting shall not violate the stall constraints.

Actually, shifting the data transmission can be either done to the left (earlier) or to the right (later). As we assume a very large playback buffer, sending the video data at earlier times will not cause packets rejection and, thus, will not cause video stalls. In other words, any data shifting to earlier times of higher capacities will be performed without violating the stall constraints. However, when the higher capacity values come later, the data shifting must be checked whether it violates the stall constraints or not. As we only shift the data transmission without changing their corresponding video bitrates, we end up having a new bitrate level strategy $\gamma_{r_{th}}$ that gives the same weighted average quality as given by γ. Thereby, the resulting strategy $(r_{th},\gamma_{r_{th}})$ outperforms strategy (r,γ), which completes the proof. □

In practice, the setting of the transmission threshold α does not follow the data shifting process of the proof. We will thus design an approach to build a threshold strategy for the transmission schedule.

B. Ascending bitrate level strategy

In this section, we study the proprieties of the bitrate level strategy under a threshold based transmission schedule. More specifically, we analyze the impact of the video quality levels' order on the setting of α.

Definition 2. *We say a bitrate level strategy is* **ascending** *if the quality levels of the video segments increases during the session, i.e., for all $0 \leq t \leq t' \leq T$*

$$b(t) \leq b(t'),\ \textit{i.e.,}\ \gamma(t) \geq \gamma(t').$$

Proposition 2. *Assume that there exists a threshold-based solution (r_{th},γ) that satisfies the constraints in (5), then*

[3] The proof still holds for a monotonically increasing c.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

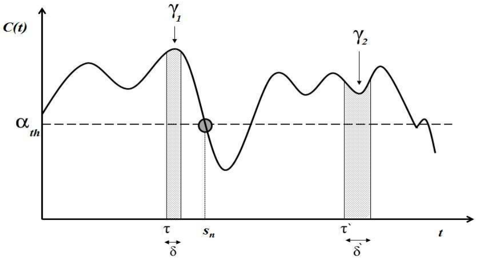

Fig. 2: Ascending bitrate level strategy.

there exists a threshold-based ascending bitrate level solution (r'_{th}, γ') *that optimizes problem in (5).*

Proof. Pick a suite of N segments with a non ascending order quality levels, in a way that they can be streamed without video stalls over the future horizon. Then, according to this quality levels' order, set a threshold-based solution (r_{th}, γ) with threshold α such that, beyond this threshold, the first constraint violation will occur at time $t = s_n$. Suppose that, under this solution, two bitrate levels b_1 and b_2 will be respectively streamed over $[\tau, \tau{+}\delta]$ and $[\tau', \tau'{+}\delta']$ as depicted in Fig. 2, such that

$$\tau + \delta < s_n, \quad \tau' > s_n, \quad b_1 > b_2,$$

and

$$\int_{\tau}^{\tau+\delta} r_{th}(t)dt = \int_{\tau'}^{\tau'+\delta'} r_{th}(t)dt.$$

Let $fr_{th}(t)$ be the network frame rate at time t. As we have $b_1 > b_2$, then the number of frames that will be streamed during $[\tau', \tau' + \delta']$ is greater than the number of frames that will be streamed during $[\tau, \tau + \delta]$. Therefore, $\exists\ \beta > 0$ such that

$$\int_{\tau'}^{\tau'+\delta'} fr_{th}(t)dt = \int_{\tau}^{\tau+\delta} fr_{th}(t)dt + \beta. \tag{14}$$

Suppose that we switch between b_1 and b_2 over these two intervals of time. Then, the number of cumulative received frames at s_n will be increased by β. Let u and u' be the cumulative number of arrival frames functions before and after switching the bitrates. therefore, we have

$$u'(s_n) = u(s_n) + \beta. \tag{15}$$

Actually, if $u'(s_n)$ is large enough and allows increasing the threshold beyond α without violating the stall constraint at $t = s_n$ and later, then the cost function will be reduced. Otherwise, the threshold remains the same without changing the system performance. In fact, as explained in the previous section, streaming the data beforehand will only add more flexibility toward the stall constraints since the buffer is assumed to be very large. We show by the sequel that, even if we switch between the two bitrate levels the streaming will remain without video stalls under the same threshold since $u' \geq u(t)\ \forall t \in [0, T]$ (see Fig. 3). Let fr'_{th} be the network frame rate function after switching. Then, we have

$$fr'_{th}(t) \ > fr_{th}(t)\ \forall t \in [\tau, \tau + \delta[, \tag{16}$$

$$fr'_{th}(t) \ < fr_{th}(t)\ \forall t \in [\tau', \tau' + \delta'[, \tag{17}$$

$$\int_{\tau}^{\tau+\delta} fr'_{th}(t) - fr_{th}(t)\ dt = \int_{\tau'}^{\tau'+\delta'} fr_{th}(t) - fr'_{th}(t)\ dt = \beta. \tag{18}$$

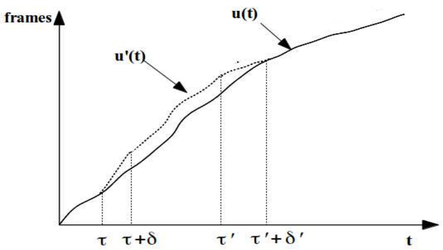

Fig. 3: Impact of bitrates switching on the cumulative number of arrival frames u.

We further define u' as

$$u'(t) = \begin{cases} u(t) & \text{t} < \tau \\ u(\tau) + \displaystyle\int_{\tau}^{t} fr'_{th}(s)\ ds & \text{t} \in [\tau, \tau + \delta[\\ u(t) + \beta & \text{t} \in [\tau + \delta, \tau'[\\ u(\tau') + \beta + \displaystyle\int_{\tau'}^{t} fr'_{th}(s)\ ds & \text{t} \in [\tau', \tau' + \delta'[\\ u(t) & \text{t} \geq \tau + \delta'. \end{cases} \tag{19}$$

Actually, the cumulative watched frames function l will remain the same as the playback frame rate λ remains the same for all bitrate levels. Now, we see clearly that $\forall\ t \notin [\tau', \tau' + \delta'[\ ,\ u'(t) \geq u(t)$. However, for $t \in [\tau', \tau' + \delta'[$, we have

$$u'(t) - u(t) = \beta - \int_{\tau'}^{t} fr_{th}(s) - fr'_{th}(s)\ ds, \tag{20}$$

which is positive according to (17) and (18). To conclude, putting the segments in an ascending bitrates' order may allow a higher transmission threshold which further reduces the cost function without degrading the average quality of the video. □

IV. Algorithmic approaches under no rebuffering events constraint

In this section we solve optimization problem (5) through algorithmic approaches based on the properties of the optimal solution characterised in the previous section. We provide an approach that compute the optimal threshold-based solution but this algorithm is faced with a high computational complexity necessary to obtain the optimal solution. Due to this shortcoming, we propose an alternative heuristic approaches to obtain nearly optimal solutions under the assumption of no rebuffering events during the session. Afterwards, we extend the study to the case where the number of video playback interruptions (stalls) can be tolerated to a certain level.

A. Optimal threshold-based solution

We summarize here our global optimal approach in three main steps : (i) first, we look for all the possible values of $\alpha \in [\alpha_{min}, \alpha_{max}]$ that satisfy the constraints in (5) and associate to each one the birates level strategy that gives the highest possible weighted average quality, (ii) for each threshold and its corresponding video quality, we compute the resulting cost function $\mathcal{F}$, (iii) the optimal solution corresponds to the one that minimizes $\mathcal{F}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

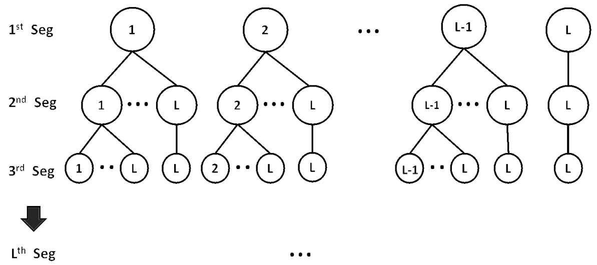

Fig. 4: Tree of choice for optimal ascending bitrate.

1) Optimal transmission schedule α: To find the optimal threshold α with the lowest complexity, we propose to sort the future capacity in an ascending way, then try its ascendent values as thresholds till reaching the one that causes video stalls. This approach will determine all the possible thresholds $[\alpha_{min}, \alpha_{max}]$. Fig. 8 illustrates the example used for the simulation section.

2) Optimal bitrate level strategy: Our approach for generating an optimal ascending bitrate level strategy consists of using a tree of choice of N levels as depicted in Fig. 4, where each level corresponds to a video segment. The nodes of a tree level i correspond to all possible quality levels that can be assigned to segment i. The parent of a node (if it exists) has either a worse or equal quality. The children (if they exist) have either a better or equal quality. We construct the tree level by level to form the path that gives the optimal sequence of bitrates. At each level, we remove the nodes whose paths cause a constraint violation in order to minimize the number of nodes at the bottom of the tree. At each level, we compute the partial weighted average quality till reaching the end of the tree. The optimal sequence of bitrates corresponds to the path that maximizes the total weighted average quality. The complexity of this algorithm may reach up to $\mathcal{O}((L+1)^N)$, which makes it non suited for online streaming services.

B. NEWCAST design

NEWCAST (aNticipating qoE With threshold sCheme And aScending biTrate levels) follows the same principle as the optimal global approach, but it uses two heuristics INVEST and AWARE for respectively computing the thresholds and generating the sequence of bitrates. Let γ_α and $\mathcal{F}_\alpha$ be the ascending bitrate levels strategy and the cost function under r_α-based transmission schedule. The main steps of this heuristic are described in Algorithm 3.

1) INVEST : INcrease with VariablE foot STep : This heuristic also follows the same principle as the optimal approach. However, instead of trying all the sorted capacity values as thresholds till violating the contraints, it defines a variable foot step to increase the threshold initially set to c_{min}. The values taken by this foot step will depend on the dynamic of the network capacity, Let $\{\alpha_1, \ldots, \alpha_M\} \subset [\alpha_{min}, \alpha_{max}]$ such that $\alpha_{i+1} > \alpha_i, i \in \{1, \ldots, M-1\}$. To compute α_{i+1} knowing α_i, we set the number of bits that we want to abandon through increasing the threshold (denoted by Q). Then, we find the capacity value (threshold) that allows doing that as described in Fig. 5. $\alpha_{i+1} - \alpha_i$ will define the i^{th} foot step (See Algorithm 4).

Algorithm 1: INVEST: INcrease with VariablE foot STep

```
Data: $c$, $i$, $Q$
SortedC=sort($c$),
CumSortedC=CumulativeSum(SortedC),
ind = $\max$(find (CumSortedC $\leq i \times Q$)),
return SortedC(ind)
```

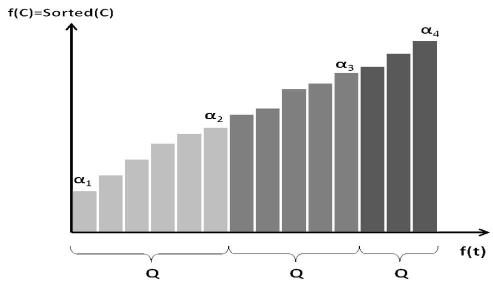

Fig. 5: INVEST: INcrease with VariablE foot STep.

2) AWARE : Anticipating qoe With Ascending bitRate lEvels: This heuristic has a polynomial complexity and is quite faster than the optimal approach. Our simulation results show that its outcoming solution approaches the optimal solution at almost 98% in terms of the video average quality. We summarize its steps in the few following points:
At the beginning, we assign the lowest bitrate to all video segments. Then, starting from the end of the video (latest segment) back to the beginning, we increase the bitrate of each segment by one level as long as the stall constraints are satisfied. We repeat this step many times till reaching the highest available bitrate (See Fig. 6). By following this approach, the number of times the bitrate will be increased is at most equal to $L-1$ (see Algorithm 2). To reduce the startup delay, which is a prominent key QoE factor (but not included in our optimization problem), we set the startup-segments to the lowest bitrate and stream them using a greedy[4] transmission rather than a threshold-based transmission. As shown in Fig. 7, an inherent advantage of this algorithm is that it ensures a progressive increase of the bitrate instead of an aggressive increase as given by the optimal approach, which is quite more appreciated by the users.

V. Algorithmic approaches under rebuffering events

So far, we have assumed no rebuffering events during the streaming session, i.e., the future capacity has been assumed quite sufficient to allow streaming the hole video at the lowest bitrate. In extreme cases, the capacity may not be sufficient and may cause the player having video stalls even with the lowest quality level. End-user may prefer to tolerate few stalls

[4] A greedy transmission uses all the available network capacities.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

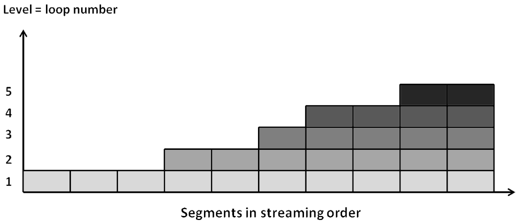

Fig. 6: Illustrative example of the bitrate increasing steps used in AWARE.

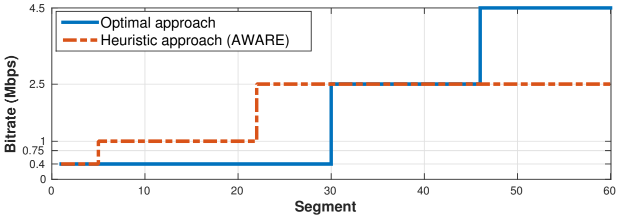

Fig. 7: Comparative example between optimal approach and AWARE.

in order to have a better quality. To go further with the analysis, we adapt our approach to a similar case where q stalls can be tolerated during a session. The optimization problem in (5) becomes

$$\min_{(r,\gamma)} \mathcal{F}(r,\gamma) = \frac{1}{T}\int_0^T \frac{r(t)}{c(t)}dt - \pi \times \frac{\sum_{j=1}^{j=L} w_j \int_0^T \gamma_j(t) r(t) dt}{S_L}, \tag{21}$$
$$s.t \begin{cases} \int_0^T \sum_{j=1}^{j=L} \frac{\lambda\ r(t)\gamma_j(t)}{b_L} = l(T), \\ \mathcal{F}_{(r,\gamma)}(T) \leq q, \end{cases}$$

where $\mathcal{F}_{(r,\gamma)}(T)$ is the number of stalls during the streaming session under strategy (r, γ).

Lemma 1. *Any optimal strategy will experience exactly q stalls.*

Proof. Assume that there exists an optimal solution (r, γ) that has experienced q' stalls such that $q' < q$. Suppose that under (r, γ), x_0 frames have been downloaded over $[\tau, \tau+\bar{x}_0]$ where $\bar{x}_0$ is the time needed to download x_0 frames. By imposing an additional starvation at time τ, the number of cumulative frames at playout buffer will be increased by x_0. This allows to give more opportunity for the transmission schedule to reduce the cost of transmission without changing their corresponding video bitrate γ and without violating the stall constraints. Thus, the strategy (r, γ) may decrease the cost function $\mathcal{F}$ by forcing an additional stall, which completes the proof. □

The following result extends the proprieties of the optimal solution by including the possibility of rebuffering. By Lemma 1, the following corollary holds.

Corollary 1. *Assume that there exists a feasible solution that satisfies the constraints in (21), then there exists an optimal strategy (r_{th}, γ_{th}) of optimization problem (21), where r_{th} is a threshold transmission schedule and γ_{th} is a threshold-based ascending bitrate level solution.*

Algorithm 2: AWARE: Anticipating QoE With Ascending bitRate lEvels.

```
Data: $c, \alpha$, videoProperties, $b_1 \dots b_L$
$s = 1$, SegmentsBitrates[1 : $N$]=$b_s$,
while $s < L$ do
    $s = s + 1$,
    Start=FirstSegmentOfBitrate($b_{s-1}$),
    End=N,
    middle = (End-Start) div2 +1,
    while middle $\geq$ 1 and End $\geq$ Start and middle $\leq$ End ) do
        init=SegmentsBitrates,
        SegmentsBitrates[middle:End]=$b_s$ ,
        SegmentsBitrates[1:StartupSegments]=$b_1$ ,
        Test = ExistViolation(SegmentsBitrates,$c$, $\alpha$,videoProperties),
        if Test then
            SegmentsBitrates[middle:End] = init[middle:End],
            middle=middle+(End-middle) div2 +1 ,
        else
            End=middle-1,
            middle=Start+(End-Start) div2 +1,
        end
    end
end
[$r_\alpha$,$\gamma_\alpha$]=TransmitVideo($c$, $\alpha$, VideoProperties, SegmentsBitrates),
Test = ExistViolation(SegmentsBitrates, $c$, $\alpha$, VideoProperties),
return ($\overline{Test}$, $r_\alpha$, $\gamma_\alpha$)
```

Algorithm 3: NEWSCAST: aNticipating qoE With threshold sCheme And aScending biTrate levels.

```
Data: $c$, VideoProperties, $L, w, Q$
$\alpha$=$c_{min}$, $i = 1$,
[PossibleTransmission, $r_\alpha$, $\gamma_\alpha$]=AWARE($c$, $\alpha$, videoProperties, $L$),
while PossibleTransmission do
    $\mathcal{F}_\alpha$=computeObjFunction ($c, r_\alpha, \gamma_\alpha, w$),
    i=i+1,
    $\alpha$ = INVEST($c, i, Q$),
    [PossibleTransmission, $r_\alpha$, $\gamma_\alpha$]=AWARE($c$, $\alpha$, videoProperties, $L$),
end
$\mathcal{F}^*_{\alpha^*}$=min{$\mathcal{F}_\alpha$},
$\alpha_{th}$=$\alpha^*$,
return ($\alpha_{th}$,$\gamma_{\alpha_{th}}$)
```

With the above results, the algorithmic approaches under no rebuffering events still hold for the general case where the number of video playback stalls can be tolerated.

In Algorithm 4, we present the modified NEWCAST algorithm where we allow video playback stalls to happen. The major modification concerns only AWARE algorithm to compute the optimal bitrate level strategy since INVEST algorithm remains unchanged under rebuffering events.

VI. Simulations and Numerical Results

A. Simulation tools and setup

We performed all our simulations using Matlab server R2015b on a Dell PowerEdge T420 Intel Xeon running Ubuntu 14.04. The streaming session was configured according to some DASH and Youtube parameters [33], [34]. To the best of our knowledge, no explicit way does really exist to compute the weights that can be accorded to the video bitrates. In [35], authors were exploring a QoE estimation model in which they

Algorithm 4: AWARE-MS$_q$: AWARE with at Maximum q Stalls

```
Data: $c$, $\alpha$, videoProperties, $b_1 \dots b_L$, maxStalls=$q$,
$s = 1$, SegmentsBitrates[1:N]=$b_s$,
while $s < L$ do
    $s = s + 1$,
    Start=FirstSegmentOfBitrate($b_{s-1}$),
    End=N,
    middle = (End-Start) div2 +1,
    while middle $\geq$ 1 and End $\geq$ Start and middle $\leq$ End ) do
        init=SegmentsBitrates,
        SegmentsBitrates[middle:End]=$b_s$ ,
        SegmentsBitrates[1:StartupSegments]=$b_1$ ,
        nbrStalls = ComputeViolations(SegmentsBitrates, $c$, $\alpha$, videoProperties),
        if nbrStalls $>$ maxStalls then
            SegmentsBitrates[middle:End] = init[middle:End],
            middle=middle+(End-middle) div2 +1 ,
        else
            End=middle-1,
            middle=Start+(End-Start) div2 +1,
        end
    end
end
[$r_\alpha$,$\gamma_\alpha$]=TransmitVideo(c,$\alpha$, VideoProperties, SegmentsBitrates),
nbrStalls = ComputeViolations(SegmentsBitrates,c,$\alpha$,VideoProperties),
Test= nbrStalls $\leq$ maxStalls,
return (Test,$r_\alpha$,$\gamma_\alpha$)
```

Window Size	3 min 10 s
Average throughput	2 Mbps
Capacity Time Slot	1 s
Video Length	3 min
Segment Length	1s
Video frame rate	30 fps
Playback cache	4 s
Video bitrates (Mbps)	[0.4 0.75 1 2.5 4.5]

TABLE I: Parameters of Matlab simulations.

were assigning to each video segment a QoE metric with a logarithmic variation as function of the bitrate and the motion factor. In [36], however, authors used a MOS (Mean Opinion Score) factor in order to reflect the user's satisfaction toward each quality level. In this paper, we assign the weights to the bitrates in a proportional way as follows : $w_i = b_i / \sum_{i=1}^{L} b_i$, where b_i is the i^{th} bitrate level and w_i is its corresponding weight. All the parameters are listed in Table I. For the sake of accuracy, we explore the values of the threshold α using the optimal approach. Our heuristic (INVEST) will be discussed later in Section VI-D.

B. Framework performance

As a first step, we generate the network capacity randomly around a constant average value. This capacity will serve us to show the main characteristics of the proposed algorithm. Then, we resort to use real-world capacities for robustness and performance comparison with baseline rate adaptation policies. Fig. 8 illustrates the dynamic of the capacity along with its correspondent threshold values α. Note that, when α exceeds its maximum value, a stall constraint will be violated. By the sequel, we define our benchmark as the case where all the future capacity is used and the highest possible video quality is delivered, i.e., $\alpha = c_{min}$. The execution of NEWCAST using the above parameters shows a variation in the system performance for π ranging from 4.50 to 4.70. Beyond the limits of this interval, the system performance remains the same. In the following analysis, we will only focus on three values of π: *low*, *medium* and *high*. Let us denote by α_π the outcoming threshold after running NEWCAST using the a chosen value of π.

Fig. 9 illustrates the variation of α_π as function of π. A small value of π results in a high α_π as it prioritizes the system utilization cost. A big value of π, however, results in a low threshold as it gives more importance to the average quality. As a matter of fact, a medium π leads to an in-between threshold that balances QoE and system cost.

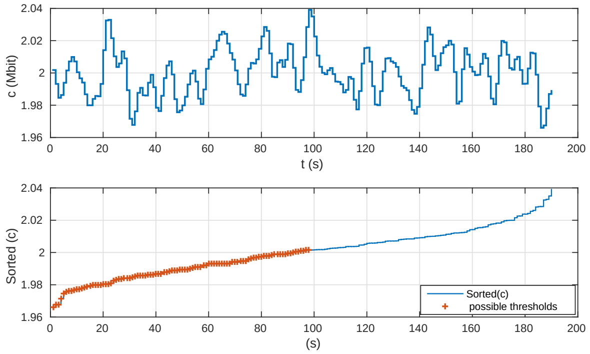

Fig. 8: Network capacity and its corresponding threshold α.

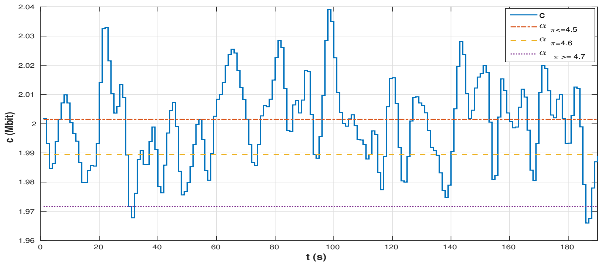

Fig. 9: Variation of α_π as function of π.

In Fig. 10, we plot the playback buffer state evolution over time and its correspondent sequence of bitrates for the three aforementioned values of π. When π is small, many silent times are noticed and the buffer state evolves with high slopes (mainly at the beginning and at the middle of the video). This is actually due to the low quality of the segments being streamed. Note that the player streams as much frames as the bitrate is low. For the medium value of π, more flexibility is noticed with shorter silent times and better quality. As for the big value of π, no silent times are noticed since almost all the network resources are used. The buffer state evolves gradually with low slopes, given the fact that segments of high-order quality are being streamed.

Now, we explore the idea of enforcing a stall during the streaming session. Let $\mathcal{F}_{or}$ be the original cost function before enforcing a stall, and $\mathcal{F}_{st}$ be the resulting cost function after enforcing a stall. In Fig. 11, we plot again the playback buffer

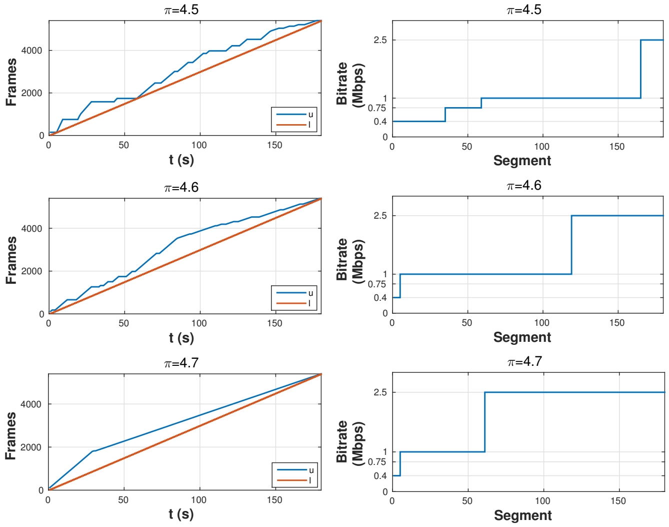

Fig. 10: Playback buffer state evolution and corresponding sequence bitrates for different π.

state evolution over time for the three values of π, and plot below the variation of $\mathcal{F}_{st}$ as function of the stall emplacement (1^{st} segment, 2^{nd} segment, etc.). As depicted in the figure, for $\pi = 4.5$, $\mathcal{F}_{st}$ experiences high fluctuations around $\mathcal{F}_{or}$ mainly when the stalls are enforced at the beginning of the video. The lowest values of $\mathcal{F}_{st}$ are noticed when the stalls are enforced at the moments where the original buffer state is critical, i.e., a low quality with no much flexibility toward the stall constraint. Note that, the critical states of the buffer at these moments prevent NEWCAST from setting a higher threshold. When a stall is enforced there, the video is divided into two independent parts and the streaming strategy is optimized before and after the stall, leading to two different thresholds that reduce the overall system utilization cost. Now, by increasing π, we observe a quasi-constant decrease in $\mathcal{F}_{st}$. A stall enforcement certainly enhances the quality at the beginning part of the video, but it condemns the flexibility and the average quality for the rest of the video. The degradation in the global quality induces a reduction in the global system cost that outweighs the resulting $\mathcal{F}_{st}$. To sum it up, a stall enforcement may be only interesting when the value of π is low since it may reduce the system cost. A judicious choice of its emplacement would be at the moments where the original buffer state is critical.

C. Robustness under prediction errors

One key limitation of the proposed idea is that there is still no explicit approach that accurately predicts the network capacity over more than ten seconds to the future. In order to evaluate the robustness of NEWCAST, we used the real throughput traces of the HSDPA dataset [1]. This dataset consists of 30 minutes of continuous throughput measurements of a moving device in Telenor's 3G/HSDPA wireless mobile network. We used the traces of the Ljabru-Jernbanetorget trajectory as it has the least variance in the throughput spatial variation (see Fig. 12 and Fig. 13). A temporal mapping of the throughput variation was performed by supposing the user moving at a speed of 50Kmph. Using the same parameters of Table I, we computed the performance $\mathcal{P}_{av}$ of NEWCAST by averaging all the throughput realizations, then, we computed its performance $\mathcal{P}_{real}$ by using each throughput realization apart. The robustness of the framework was evaluated through the performance averaged error rate

$$\mathcal{P}_{error} = \left| \frac{\mathcal{P}_{real} - \mathcal{P}_{av}}{\mathcal{P}_{av}} \right|.$$

Results shown by Fig. 14 depict an averaged error rate less than 15% for both the system cost and the average quality. They even depict a lower sensitivity of the system cost to prediction errors when π is smaller, and a lower sensitivity of the average quality to prediction errors when π is higher. In general, we can claim that our scheme performs pretty well even with the presence of real prediction errors.

D. Complexity

1) Framework performance under bigger time slots.: In Fig. 15, we compute the mean execution time of NEWCAST (using optimal thresholds) by averaging results on 100 (randomly generated) capacities and using different time slots (from 1s to 5s). It takes almost 4s to compute the final strategy with a time slot equal to 1s. As expected, using bigger time slots takes much shorter time. However, this comes at the expand of the final result accuracy depending on the value of π. In the same figure, we show the system response (through $\mathcal{F}$) for each time slot by averaging results over the 100 capacities. We compute an accuracy rate factor (≤ 1) by comparing the obtained results with the result of 1s time slot. In our model, we assume that in a time slot only one bitrate level can be streamed, which explains why using bigger time slots may add constraints to the QoE. For high values of π, very slight degradation is noticed since the system tends to use all the network resources. However, for low values of π, the constraints have bigger impact since the system tends to use less network resources, which explains the higher degradation in the QoE, and, by the sequel, the higher reduction in the system cost.

2) Framework performance under different values of Q: Here, we set the time slot to 1s and run NEWCAST using different values of Q (between 1Mbit and 5Mbits) by averaging results on the same 100 capacities. Results in Fig. 16 show that setting Q to the average throughput (2Mbps) leads to a high accuracy rate (≈ 1) with an execution time of 4s (as for optimal thresholds). Setting lower values of Q, increases the execution time and keeps almost the same accuracy on $\mathcal{F}$. For higher Q, the complexity is notably reduced, but slight degradations are noticed on the accuracy rate (less than 16%). A judicious choice of Q should then be made depending on the operator's preferences: a high Q gives a high QoE and a very low complexity, whereas, a low Q gives a low system cost and a higher complexity.

E. Comparison with baseline adaptive bitrate (ABR) algorithms

In this section, we compare NEWCAST to two baseline ABR algorithms: one is throughput-based (TB-ABR) in [16], the other is buffer-based (BB-ABR) in [27]. We develop each algorithm on Matlab and simulate its behaviour on different

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

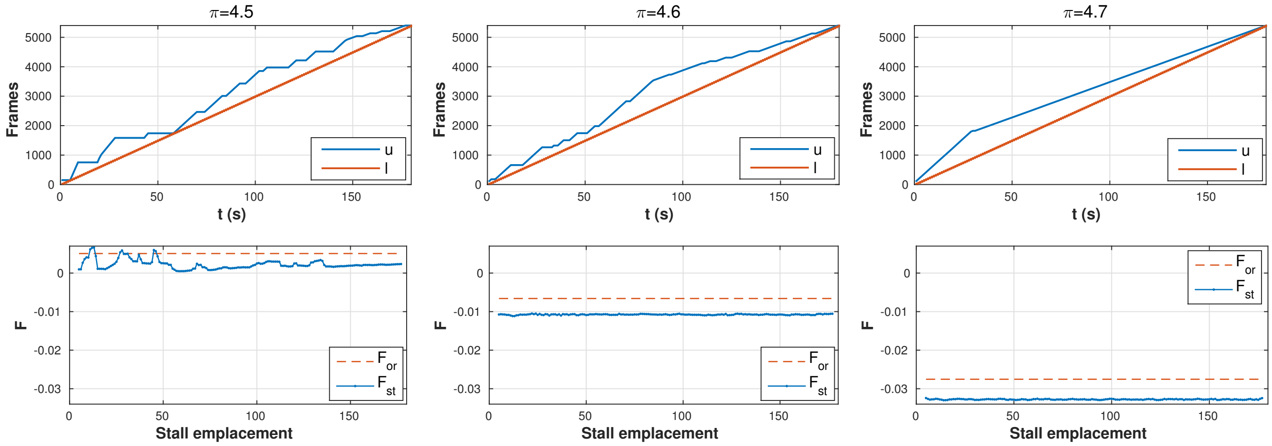

Fig. 11: System performance with buffer stall enforcement.

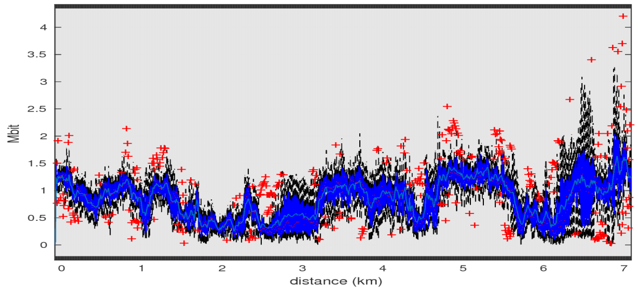

Fig. 12: Experimental spatial variations of the capacity for the tramway Ljabru-Jernbanetorget trajectory.

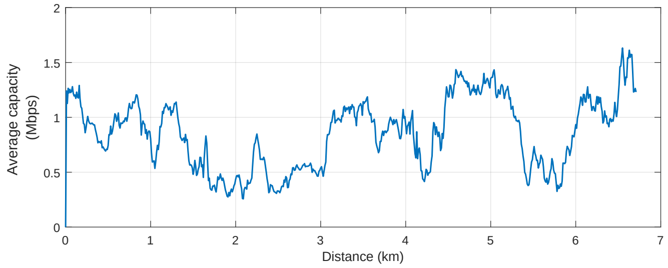

Fig. 13: Average spatial variations of the capacity for the tramway Ljabru-Jernbanetorget trajectory.

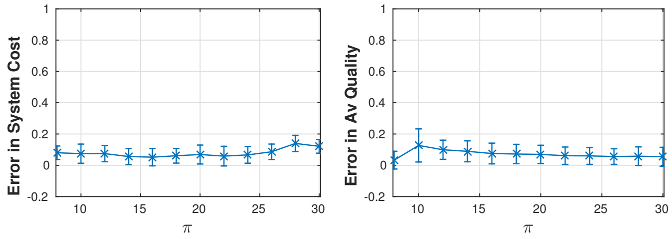

Fig. 14: Average error rate on the system performance under real throughput prediction errors.

video streaming sessions. We keep all the parameters setting of Table I.

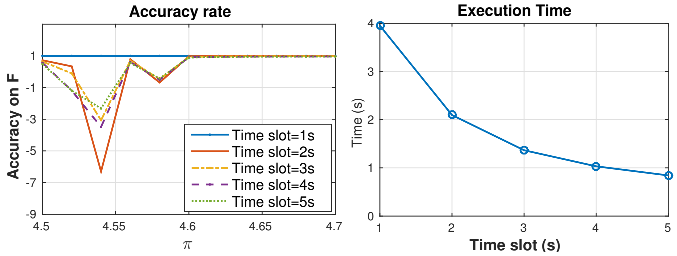

Fig. 15: Accuracy and complexity variations with different time slots.

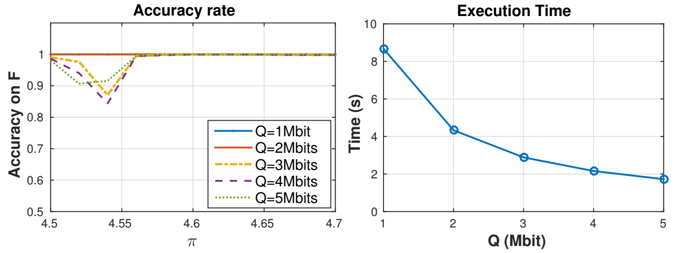

Fig. 16: Accuracy and complexity variations with different Q.

1) ABR algorithms: The key difference between current ABR algorithms is the logic they use for bitrate selection. As found in the literature, ABR logics can be categorized in two main classes: throughput-based class [21], [37], [38] and buffer-based class [15], [27], [39]. While the first class relies only on the next throughput prediction to decide on the current bitrate selection, the second class relies only on the current playback buffer occupancy. Few algorithms, however, were proposed as a mixture of throughput-based and buffer-based algorithms [17].

2) TB-ABR and BB-ABR configuration: criteria of choice for comparison with NEWCAST: The main characteristic of NEWCAST is that it increases the quality of segments *progressively* to avoid bothering the user with sudden quality jumping. To do so, we configure the TB-ABR and the BB-ABR algorithms to be both *conservative*. For TB-ABR, we use *the smoothed throughput* estimation such that

$$\hat{T}(i+1) = \sum_{k=i-3}^{i} p_k T(k), \tag{22}$$

with $p_1 = 0.5, p_2 = 0.3, p_2 = 0.15$ and $p_2 = 0.05$. $T(i)$ designs the throughput measured after downloading segment i, and $\hat{T}(i+1)$ is the throughput estimate of segment $i+1$. As for the bitrate selection, we use a method close to that defined in ”Microsoft Smooth Streaming”.

For BB-ABR, we use the algorithm in [27]. The bit rate selection is determined by a mapping function that characterises the relation between the bitrate of the next segment and the current buffer size. The algorithm defines two thresholds B_{min} and B_{max} and design a buffer-based controller that take into account several metrics such as playback freezing, bitrate switch and video quality.

3) Capacities of test: In this section, we use the real throughput traces of the online 4G/LTE dataset [2] collected for the car trajectory. Both NEWCAST and the ABR algorithms are evaluated. Our analysis is driven by the three metrics that mostly characterize NEWCAST: the system cost, the per segment average video quality and the average number of quality switching. In Fig.17, we plot each of these metrics as function of π for π ranging from 1 to 7.

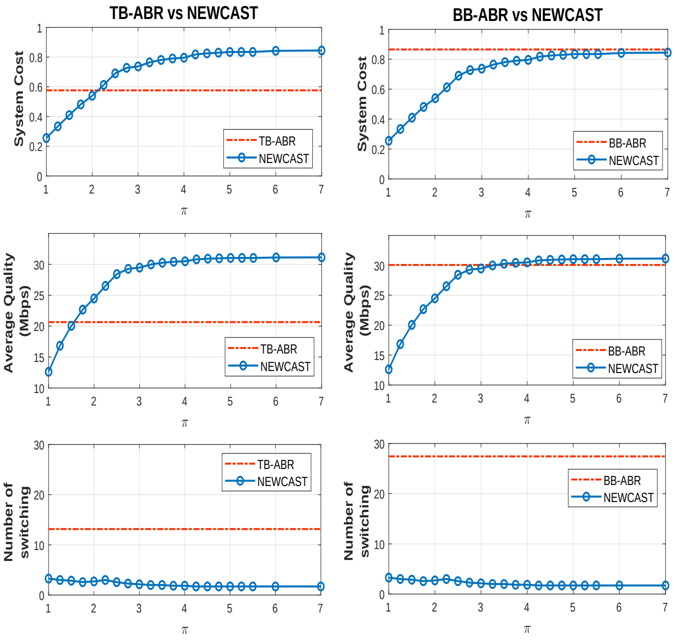

Fig. 17: TB-ABR vs. NEWCAST and BB-ABR vs. NEWCAST (without stalls).

4) Main comparison points:

a. TB-ABR vs. NEWCAST: According to Fig. 17, the main advantage of NEWCAST is that it can achieve the same quality as TB-ABR with a system utilization cost reduced by at least 21%, and that it can achieve the same system cost with an average quality enhanced by up to 28%. This is mainly due to the smart threshold-based-strategy of NEWCAST that uses the less expensive resources depending on the value of π. It is then up to the operator to make the tradeoff and to wisely calibrate the value of π to outperform the TB-ABR algorithm. A further important observation lies in the very reduced number of quality switching achieved by NEWCAST (at most 2) compared to that achieved by TB-ABR (around 13).

b. BB-ABR vs. NEWCAST: We notice from Fig. 17 that BB-ABR is very greedy toward the resource usage compared to TB-ABR, which makes it give near performance to NEWCAST when applied with high values of π. Actually, for some values of π, NEWCAST outperforms BB-ABR, but this outperformance is marginal. In fact, the same average quality can be achieved with a system cost reduced by 11.68%, and the same system cost can be achieved resulting in an average quality increased by 3.49%. The greedy character of BB-ABR can be either emphasized or de-emphasized depending on the mapping function, which chooses the bitrate of the next chunk based on the current buffer state. So, it may happen that BB-ABR uses all the resources and gives a higher average quality than NEWCAST, but this outperformance will not exceed 2% since the heuristic used by NEWCAST approximates the optimal quality arrangement by 98%. Overall, the most noteworthy advantage of NEWCAST, is that it gives a far less number of quality switching (at most 2 against 27 with BB-ABR), which is very well appreciated for the users’ perceptions. Moreover, NEWCAST limits the risk to have a starvation compared to BB-ABR. Recall that NEWCAST avoid a starvation when the predictive capacity allows to download the video with low quality.

In conclusion, when the knowledge of the future throughput is perfect[5], NEWCAST can perform better than the baseline TB-ABR and BB-ABR algorithms. By mean of a wise calibration of the value of π, the tradeoff between system utilization cost and QoE can be steered to either save more resources or increase the average quality. In all cases, the number of quality switching remains the most suitable for the end user’s perception.

VII. Framework design and implementation

NEWCAST is designed to manage the video streaming process in order to maximise the subjective video quality with minimum radio resources. In order to achieve this target, we implement NEWCAST at the client side to ensure that each adaptive video stream can adapt its strategy as function of the time window in which the prediction is accurate. Such a design has several benefits (i) NEWCAST can be combined with any existing scheduling policy implemented at the base station such as proportional fair (PF) scheduler. Notice that PF schedulers strive to achieve fairness of resources allocated across the users. (ii) NEWCAST does not require modification of existing base station schedulers, facilitating quicker deployments and (iii) all types of traffic can be handled with the existing scheduler at the base station.

As the the prediction of future capacity is done by the operator based on collected data, it is natural to consider that this information on the future capacity is available at the base station side. However, the future capacity is sent to each adaptive video flow and each user computes the optimal threshold strategy α_{th} and the optimal bitrate levels strategy using NEWCAST. The optimal threshold strategy α_{th}

[5] The case where no accurate throughput prediction is made available is left for another work [40].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

is signaled to the base station in order adjust its priorities among users. The scheduler can use a filtering approach to allocate resource to mobiles based on values of α_{th} of all video streaming users in the cell. If the capacity of a user is accurate, then the scheduler knows at each TTI whether a mobile user needs to be served or not. A simple way to do this consists in adding a filter behind the scheduler to restrict the set of UEs to be served at each TTI, which corresponds to 1 ms in LTE. If the predicted capacity of a given user is below the threshold α_{th}, this user will be excluded from being served by the scheduler during the time slot TTI. This allows the scheduler to allocate the resource not used by adaptive traffic to other types of traffic.

A. *NEWCAST interactions with real video streaming entities*

In real environments, NEWCAST shall be implemented at the client side as an independent framework. It shall be able to communicate the threshold α_{th} to the network scheduler and the set of video bitrates γ_{th} to the media player as described in Fig. 18. The transmission threshold α_{th} as a kind of a cross layer that also allows to apply the threshold-based transmission scheme. The set of video bitrates γ_{th}, however, can be directly sent to the player at the beginning of the streaming session. These bitrates will then be consecutively requested by the player to the streaming server. Note that, in our analytical model, the variable γ_{th} was set to describe the variation of the video bitrate *in function of time*, in real implementation, the player will not use it that way, it will rather use the bitrate variation *in function of the segments' orders*, which can be directly returned by NEWCAST. In Fig. 19, we show the sequence diagram of the video streaming process using NEWCAST.

The prediction of the future capacity is available at base station, and each video streaming user receives its future capacity by the base station. However, each user computes the optimal threshold strategy α_{th} and the optimal bitrate levels strategy γ_{th}. The base station receives the value of α_{th} from each user and incorporates it into the scheduler. The scheduler uses a filtering approach to allocate resource to mobiles based on values of α_{th} of all mobile users in the cell. If the prediction of the future capacity of a user is accurate, then the scheduler knows at each time slot if a mobile user needs to be served or not. A simple way to do this consists in adding a filter behind the scheduler to restrict the set of users to be served at each transmission time interval (TTI), which corresponds to 1 ms in LTE. If the predicted capacity of a user is below the threshold α_{th}, this user will be excluded from being served by the scheduler during the time slot TTI. Unused resource could be affected to other types of traffic or users that their future capacities are not available.

B. *Implementation tools and environment*

We use a Linux environment with two virtual machines: one is used as a DASH server and the other is used as a DASH client. In the DASH server, we install Apache and put inside the Dashjs framework [41] with the Envivio video segments encoded at different quality levels [42]. In the DASH client, we only install Google chrome browser. We configure the two virtual machines to be able to communicate through their Ethernet interfaces. To emulate the network schedule and make the bandwidth between the two machines follow a predefined variation (considered as the predicted capacity), we use the Linux tc-tool for traffic shaping as shown in Fig. 20. To develop NEWCAST and make it interact with the Dashjs player, we use Javascript and other basic web languages. NEWCAST is put with the player call function in a same *.php* file that the DASH client requests to start the video streaming session. A video demo of NEWCAST is put available online in [43]. In Table II, we put more details on the hardware/software tools used for the implementation.

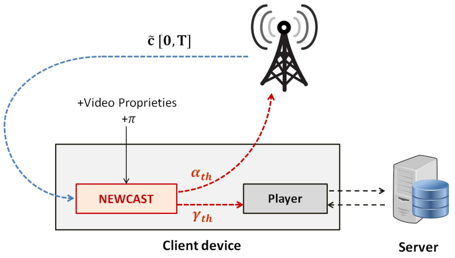

Fig. 18: Illustration of NEWCAST interactions with the network scheduler and the media player.

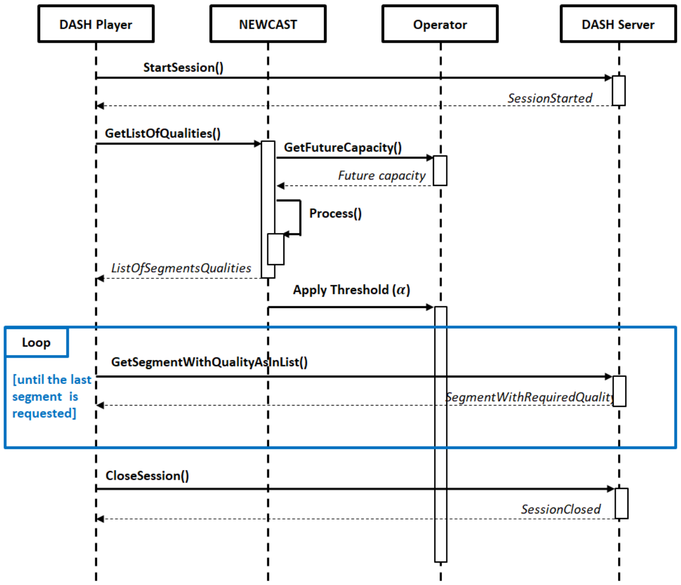

Fig. 19: Sequence diagram of a video streaming session using NEWCAST.

Host machine	Optiplex 7010 Intel Core i7-3770 CPU 3.40Ghz
Distribution	Ubuntu 14.04.5 LTS
Virtual machines	Linux Container Lxc 1.0.9
Apache	2.4.7
Dashjs	2.4.0
Google Chrome	55.0.2883.87

TABLE II: Details on the software/hardware tools used for real implementation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

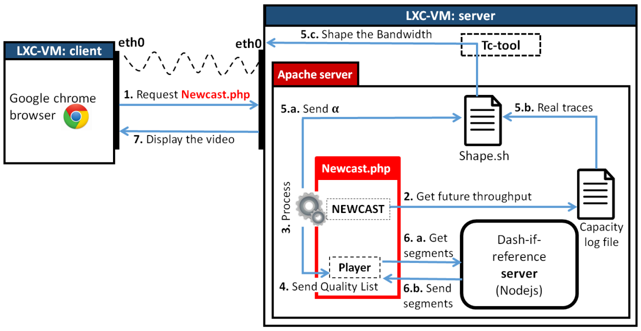

Fig. 20: Architecture of the system used for experiments.

C. Requirements for real implementation

1) Changes inside the Dashjs framework: To make NEWCAST interact with the Dashjs framework, we made some changes inside the media player: (i) A new event was added to the player class to detect the moments where a segment of type *"video"* is completely loaded to the client. (ii) The restrictions on the playback buffer size defined at the *"MediaPlayerModel.js"* file were changed to fit the infinite buffer size assumption, since, otherwise, the player will remove the earliest played segments and, in some cases, delay the requests of the coming segments. (iii) The threshold of prefetching *after a stall happens* was changed inside the *checkIfSufficientBuffer()* function to fit the prefetching threshold used by NEWCAST.

2) Required player APIs: Two essential APIs are actually responsible for the interaction between NEWCAST and the Dashjs framework: The *setAutoSwitchQuality()* API to disable the quality auto-switch mode of the player and the *setQualityFor()* API to enforce the quality of the coming segments.

3) Traffic configuration: To make the *real* throughput compliant to the throughput r modelled by NEWCAST, we processed as follows: (i) We deleted the *"audio traffic"* description from the *.mpd* file since, in our study, we are only interested in *video traffic*. (ii) We added Apache to the Linux sudoer list to allow it use the tc-tool functions and shape the bandwidth in parallel to the streaming. (iii) As we found that the average duration of a real *segment-request* is equal to 0.06s, which is not insignificant as was assumed in our theoretical model, we considered, for the implementation, each *segment-request* as a virtual file of size 0.06 multiplied by the predicted throughput at the considered second. (iv) A long stall duration caused by a high threshold α_{th} may lead the session to be closed. To avoid such situations we were disabling the threshold transmission schedule during prefetching when a stall happens.

D. Validation through experiments

To supervise the system behavior in real time, we developed a graphical interface in which we plot the real time throughput variation, the real time buffer evolution and the real time video bitrate alongside with the strategies modeled by NEWCAST, as shown in our video demo [43]. We conduct the same experiment several times using one of the throughput logs available in [1] and different values of π. Results shown by Fig. 21 depict a high instability of the system behaviour when the value of π is small (between 0.1 and 0.7), i.e., when the threshold α_π is high. This instability actually induced high numbers of video stalls. More stability, however, is noticed when the value of π is high (between 0.8 and 1.6). A noteworthy observation here is in the fact that the real system reacts very closely to what was modeled by NEWCAST. The difference in the system utilization cost is very small (approximately equal to 5.2%) and the average number of video stalls too (almost equal to 0.53). Although we were conducting the same experiments for each value of π, the system behaviour was variable. We link this, mainly, to the casual errors of the bandwidth shaping and to the variation of the *segment-request* duration. Overall, these results offer hope that, under high values of π, the exploitation of NEWCAST in real environments becomes feasible, unless an accurate throughput prediction is available. Under low values of π, however, the quality of the streaming risks to be degraded since the system becomes sensitive to the tiniest prediction error.

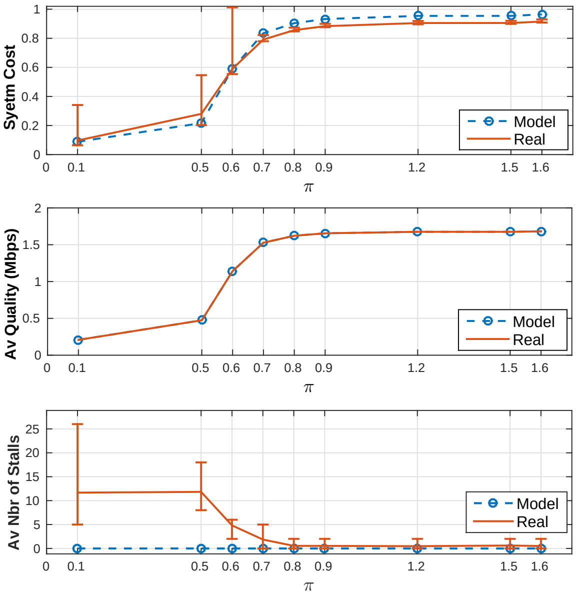

Fig. 21: NEWCAST performance in real environment.

VIII. Conclusion

In this paper, we have developed a new framework called NEWCAST, for optimizing the delivery of video streaming content under the knowledge of future capacity. This framework has been designed to balance the system utilization cost and some key QoE metrics such as average video quality and rebuffering events. From an implementation point of view, results have shown the possibility to use NEWCAST as an online algorithm (well suited for dynamic adaptive streaming over HTTP). Real experiments conducted with a real DASH player have shown that NEWCAST can be efficiently used in real-world streaming provided that the throughput is accurately estimated. Interesting future directions consist in incorporating errors in the throughput prediction to see how much this impacts on the robustness of the proposed approach.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2019.2952498, IEEE Transactions on Network and Service Management

References

[1] DATASET: HSDPA-bandwidth logs for mobile HTTP streaming scenarios. http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/.

[2] DATASET: 4G/LTE Bandwidth Logs. http://users.ugent.be/~jvdrhoof/dataset-4g.

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

[4] Y. Xu, E. Altman, R. El-Azouzi, M. Haddad, S. Elayoubi, and T. Jimenez, "Analysis of Buffer Starvation With Application to Objective QoE Optimization of Streaming Services," *Multimedia, IEEE Transactions on*, vol. 16, no. 3, pp. 813–827, April 2014.

[5] J. Song, F. Yang, Y. Zhou, S. Wan, and H. R. Wu, "QoE evaluation of multimedia services based on audiovisual quality and user interest," *IEEE Trans. Multimedia*.

[6] C. Zhou, C.-W. Lin, and Z. Guo, "mDASH: A Markov Decision-Based Rate Adaptation Approach for Dynamic HTTP Streaming," *IEEE Trans. Multimedia*, vol. 18, pp. 738–751, 2016.

[7] B. Rainer, S. Petscharnig, C. Timmerer, and H. Hellwagner, "Statistically indifferent quality variation: An approach for reducing multimedia distribution cost for adaptive video streaming services," *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 849–860, April 2017.

[8] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "SDNHAS: An SDN-Enabled Architecture to Optimize QoE in HTTP Adaptive Streaming," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2136–2151, Oct 2017.

[9] K. Yamagishi and T. Hayashi, "Parametric quality-estimation model for adaptive-bitrate-streaming services," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1545–1557, July 2017.

[10] S. Tasaka, "Bayesian hierarchical regression models for qoe estimation and prediction in audiovisual communications," *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1195–1208, June 2017.

[11] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *ACM SIGCOMM*, 2013.

[12] C. Yim and A. C. Bovik, "Evaluation of temporal variation of video quality in packet loss networks," *Signal Processing: Image Communication*, vol. 26, no. 1, pp. 24–38, 2011.

[13] A. K. Moorthy, L. K. Choi, A. C. Bovik, and G. de Veciana, "Video quality assessment on mobile devices: Subjective, behavioral and objective studies," *J. Sel. Topics Signal Processing*, pp. 652–671, 2012.

[14] A. Bentaleb, B. Taani, A. Begen, C. Timmerer, and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP," *IEEE Communications Surveys & Tutorials*, 08 2018.

[15] 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.

[16] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic http streaming," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, 2012.

[17] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," *SIGCOMM Comput. Commun. Rev.*, pp. 325–338, 2015.

[18] A. Jain and A. Terzis and N. Sprecher and P. Szilagyi and H. Flinck, "Mobile Throughput Guidance Signaling Protocol draft-flinck-mobile-throughput-guidance-00," *IETF*, April 2014.

[19] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward QoE-assured 4K video-on-demand delivery through mobile edge virtualization with adaptive prefetching," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2222–2237, Oct 2017.

[20] K. T. Bagci, K. E. Sahin, and A. M. Tekalp, "Compete or collaborate: Architectures for collaborative DASH video over future networks," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2152–2165, Oct 2017.

[21] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, 2012.

[22] V. Joseph and G. de Veciana, "NOVA: QoE-driven optimization of DASH-based video delivery in networks," in *INFOCOM, Proceedings IEEE*, April 2014.

[23] S. Colonnese, F. Cuomo, T. Melodia, and I. Rubin, "A cross-layer bandwidth allocation scheme for HTTP-based video streaming in LTE cellular networks," *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 386–389, Feb 2017.

[24] S. Colonnese, F. Cuomo, L. Chiaraviglio, V. Salvatore, T. Melodia, and I. Rubin, "CLEVER: a cooperative and cross-layer approach to video streaming in HetNets," *IEEE Trans. Mobile Comput.*, vol. PP, no. 99, pp. 1–1, 2017.

[25] Y. Im, J. Han, J. H. Lee, Y. Kwon, C. Joe-Wong, T. Kwon, and S. Ha, "FLARE: Coordinated rate adaptation for HTTP adaptive streaming in cellular networks," in *IEEE Int. Conf. Distributed Computing Systems (ICDCS)*, June 2017, pp. 298–307.

[26] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proc. 19th Annu. Int. Conf. Mobile Computing and Networking*, ser. MobiCom '13. New York, NY, USA: ACM, 2013, pp. 389–400.

[27] W. Huang, Y. Zhou, X. Xie, D. Wu, M. Chen, and E. Ngai, "Buffer state is enough: Simplifying the design of QoE-aware HTTP adaptive video streaming," *IEEE Trans. Broadcast.*, pp. 1–12, 2018.

[28] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb 2014.

[29] Z. Lu and G. de Veciana, "Optimizing stored video delivery for mobile networks: The value of knowing the future," in *INFOCOM*, 2013.

[30] K. Miller, A.-K. Al-Tamimi, and A. Wolisz, "Qoe-based low-delay live streaming using throughput predictions," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 1, pp. 4:1–4:24, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/2990505

[31] S. Colonnese, F. Cuomo, K. Miller, V. Sapio, and A. Wolisz, "Affordable delay based quality selection for http adaptive video streaming," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, June 2017, pp. 1–2.

[32] Imen Triki, Rachid El-Azouzi and Majed Haddad, "Newcast: Anticipating resource management and qoe provisioning for mobile video streaming," 2018. [Online]. Available: http://arxiv.org/abs/1512.05705

[33] *Live encoder settings, bitrates and resolutions*, https://support.google.com/youtube/answer/2853702?hl=en, consulted in 2015-07-18.

[34] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, 2012.

[35] Y. Shen, Y. Liu, Q. Liu, and D. Yang, "A method of QoE evaluation for adaptive streaming based on bitrate distribution," in *IEEE International Conference on Communications, Workshops Proceedings*, 2014.

[36] A. E. Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, and E. G. Steinbach, "Quality-of-experience driven adaptive HTTP media delivery." IEEE, 2013, pp. 2480–2485.

[37] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *ACM Multimedia Syst.*, 2011.

[38] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012.

[39] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming," *IEEE Journal on Selected Areas in Communications*, 2014.

[40] I. Triki, R. El-Azouzi, and M. Haddad, "Anticipating resource management and qoe for mobile video streaming under imperfect prediction," in *2016 IEEE International Symposium on Multimedia (ISM)*, Dec 2016.

[41] *Dash-Industry-Forum*, https://github.com/Dash-Industry-Forum/dash.js/.

[42] *Index of /129021/dash/envivio/Envivio-dash2*, http://dash.edgesuite.net/envivio/Envivio-dash2.

[43] Video demo of NEWCAST. https://drive.google.com/open?id=0B1gjdIZb5PPIcW1OLWY4d2xKS2s.