QUETRA: A Queuing Theory Approach to DASH Rate Adaptation

Praveen Kumar Yadav
School of Computing
National University of Singapore
praveen@comp.nus.edu.sg

Arash Shafiei*
Grenoble Rhône-Alpes
INRIA
arash.shafiei@inria.fr

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

ABSTRACT

DASH, or Dynamic Adaptive Streaming over HTTP, relies on a rate adaptation component to decide on which representation to download for each video segment. A plethora of rate adaptation algorithms has been proposed in recent years. The decisions of which bitrate to download made by these algorithms largely depend on several factors: estimated network throughput, buffer occupancy, and buffer capacity. Yet, these algorithms are not informed by a fundamental relationship between these factors and the chosen bitrate, and as a result, we found that they do not perform consistently in all scenarios, and require parameter tuning to work well under different buffer capacity. In this paper, we model a DASH client as an $M/D/1/K$ queue, which allows us to calculate the expected buffer occupancy given a bitrate choice, network throughput, and buffer capacity. Using this model, we propose QUETRA, a simple rate adaptation algorithm. We evaluated QUETRA under a diverse set of scenarios and found that, despite its simplicity, it leads to better quality of experience (7% - 140%) than existing algorithms.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

DASH; rate adaptation; queuing model; HTTP streaming

1 INTRODUCTION

DASH, or Dynamic Adaptive Streaming over HTTP, has emerged as a standard for delivering IP video due to its flexibility, usability, and ease of content management. DASH encodes a video content into different *representations*, each having a different quality and encoded into a different bitrate. Each representation is further segmented into segments of equal duration, usually in the order of seconds. The available segments and representations are provided to the client through a manifest file, using which the client can control which representation to download (and when) for each segment, using a *rate adaptation* algorithm. Rate adaptation is non-trivial: On one hand, the client should download a higher bitrate and better quality representation to improve the viewing quality. On the other hand, downloading a segment with bitrate higher than the current network throughput would drain the buffer and eventually lead to stall. Moreover, frequent switching between different representations leads to fluctuating video quality and impacts the viewing quality. In network settings where the throughput varies significantly over time (such as wireless network), predicting the network throughput and reducing the frequency of changing representation can be particularly challenging.

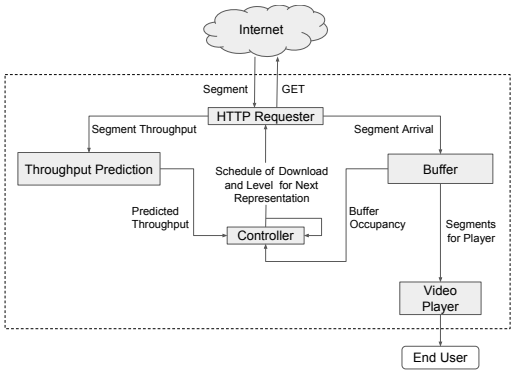

Figure 1: Generic DASH rate adaptation process.

Figure 1 shows a generic DASH rate adaptation process. The controller decides the download schedule and the representation of the next segment, depending either on the predicted throughput (estimated using segment download history), buffer occupancy, the current representation's bitrate, or their combinations. Many rate adaptation algorithms have been proposed in the literature, considering different factors in their rate adaptation decision. Section 2 surveys the related work in more details.

The decision factors for rate adaptation above, however, are not independent. The buffer occupancy in a DASH player depends on the network throughput, the bitrate of the representations chosen, and the buffer capacity. The inter-dependent relationship, however, is not explored and exploited in the existing works on rate adaptation. Furthermore, buffer capacity has a non-negligible influence on the performance of the rate adaptation algorithm that has not been studied in any existing work. To explore this relationship, we model the player buffer as an $M/D/1/K$ queue and derive the expected buffer occupancy under ideal condition (when video bitrate equals throughput). Based on this new understanding, we then designed and implemented QUETRA, a DASH rate adaptation algorithm that attempts to converge the buffer occupancy towards the ideal one. The strengths of QUETRA is that it is extremely simple and does not require the user to configure any weight or threshold, beyond a low reservoir threshold and decision period. We compared QUETRA with some of the state-of-the-art rate adaptation approaches,

*This work was done when the author is at National University of Singapore.

MM'17, , October 23–27, 2017, Mountain View, CA, USA.

ACM ISBN ISBN 978-1-4503-4906-2/17/10…$15.00
https://doi.org/http://dx.doi.org/10.1145/XXXXXX.XXXXXX

Method	Th	BO	BR
Dash.js ABR [?]	×		
BBA [?]		×	
ELASTIC [?]	×	×	
FESTIVE [?], PANDA [?], LOLLYPOP [?]	×		×
BOLA [?], SARA [?]		×	×
Tian and Liu [?], QDASH [?], Yin et al. [?], Zhang et al.[?]	×	×	×

Table 1: Rate adaptation factors (Th: estimated throughput, BO: buffer occupancy, BR: bitrate) for different methods.

including BOLA [?], ELASTIC [?] and Buffer-based Approach (BBA) [?], as well as the default rate adaptation algorithm, ABR, implemented in `Dash.js` [?]. QUETRA is able to attain 7%-140% higher QoE (using the model proposed by [?]) than the compared methods for different buffer capacities. More importantly, QUETRA performs consistently well across different buffer capacities and different scenarios without parameter tuning.

We further evaluated QUETRA with different throughput prediction algorithms. We found that simpler prediction methods that follow the throughput more closely (less smoothing) work better than methods that are more conservative in our experiments.

Contributions. First, we propose QUETRA, a simple rate adaptation algorithm based on an $M/D/1/K$ queuing model of DASH. The queuing model allows us to understand and better exploit the relationship between buffer capacity, buffer occupancy, network throughput, and the chosen bitrate. In particular, we are the first to analyze the effect of buffer capacity on the performance of the rate adaptation algorithms. These factors are used in the existing methods but often treated independently. Second, we evaluated QUETRA, BBA, ELASTIC, BOLA, and ABR extensively, using seven video samples, four network profiles, and three buffer settings. Existing works generally compare a smaller set of methods with limited video samples and network profiles, on one buffer setting. Our results reveal specific cases where the existing methods behave poorly and we provide an analysis of some of these cases.

Paper Organization. Section 2 provides a brief overview of important related work on DASH rate adaptation. Section 3 presents the $M/D/1/K$ queuing model for DASH streaming. We then present QUETRA rate adaptation algorithm in Section 4 and the evaluation of the algorithm in Section 5. We conclude the paper in Section 6.

2 RELATED WORK

We discuss the related work in DASH rate adaptation based on their approach. The rate adaptation methods primarily depend on throughput, buffer occupancy, and the representation bitrate. Table 1 summarizes the methods we discuss in this section, and the factors that they use.

2.1 Throughput-Based Method

A throughput-based method depends only on the throughput for deciding the bitrate of the next segment. The naive approach uses the throughput of the last segment as the future estimated value to select a bitrate less than or equal to it. Estimating future throughput is challenging as it may fluctuate due to network dynamics and causes a high number of changes in the chosen bitrate. Various algorithms use heuristics to avoid this situation.

DASH Industrial Forum's `Dash.js` player [?] uses the average of last three segment throughput values as the estimate of the future throughput to mitigate the fluctuation in estimation in the estimated value. The player stops downloading the next segment when the buffer is full and resumes it immediately when the buffer occupancy is lower than the capacity.

Because of the OFF period caused by buffer overflow, the other players cannot sense each other's presence in the network and predict the wrong throughput share. FESTIVE randomizes the time to start the download after the occurrence of overflow to overlap the on-period and sense the presence of other players [?]. The estimation of throughput is done by applying the harmonic mean on the last 20 segment arrivals. The player chooses to increase the bitrate more aggressively to reduce the chances of overflow. Fluctuation caused by frequent changes in the throughput is controlled by delaying the decision to change the bitrate. The delayed change may cause the underutilization of the bandwidth that is avoided by using a balancing factor between stability in the quality level and the bandwidth utilization efficiency.

PANDA [?] uses a probing method similar to TCP to estimate the bandwidth. It uses multiplicative increase and additive decrease using a fixed convergence and additive increase rate. It further applies filters to remove outliers. Reducing the number of changes in bitrate, however, is not explicit in PANDA.

2.2 Buffer-Based Approach

Buffer occupancy is much easier to observe and predict than network throughput. Huang et al. utilized this property to propose a simple approach that linearly maps buffer occupancy to bitrate for switching [?]. The method uses lower and upper reservoir of the buffer to avoid underflow and overflow respectively. Setting up an appropriate lower and upper reservoir is critical as small threshold leads to stalling whereas bigger threshold hampers the average playback bitrate.

SARA is another buffer-based method that considers both the actual segment size and the buffer occupancy for bitrate switching [?]. Although the method considers the harmonic mean of throughput to calculate the download time of the segment, the current buffer occupancy is the main factor for changing the bitrate. SARA also uses a lower reservoir, as well as other buffer thresholds to decide whether to increase the bitrate level conservatively or aggressively, or to delay the download of the next segment.

2.3 Buffer- and Throughput-Based Approach

Some methods apply control theory for rate adaptation, using both buffer occupancy and throughput as inputs. ELASTIC improves fairness by eliminating the ON-OFF period using a PID controller to converge the buffer occupancy to a given level [?]. The control depends on the ratio of throughput to bitrate and the difference between the target and current buffer occupancy.

Tian et al. [?] have proposed another PID controller based approach. One of the control variables is dependent on the ratio of

throughput and bitrate as well but the other one dependent exponentially on the gap between the target and current buffer occupancy. The controller tries to minimize the changes in quality using gradual increment and maximize the average bitrate.

Yin et al. use a model predictive controller to maximize QoE. The proposed QoE is primarily dependent on total bitrate played, the difference in bitrate during quality switch, stalling of buffer, and startup delay [**?**]. The method predicts some future throughput and tries to maximize the QoE-based on the factors. Maximization of QoE is treated as an optimization problem that is solved at each step using CPLEX solver. The QoE achieved is more than the `Dash.js` ABR [**?**], BBA [**?**], and FESTIVE [**?**].

2.4 QoE-Centric Approach

A sudden decrease of more than one quality level may hamper QoE of the user. Mok et al. [**?**] proposed QDASH, a rate adaptation method that gradually reduces the bitrate. QDASH avoids a sudden change in bitrate by downloading the segment at an intermediate level when the measured throughput changes significantly, even if the bitrate of the intermediate level representation is higher than the measured throughput. The method predicts throughput using RTT and calculates the number of intermediate-level segments that can be downloaded at one level higher than the supportable bitrate level without stalling.

In another QoE-based approach, LOLYPOP focuses on live DASH video streaming, where keeping small buffering delay is important [**?**]. As such, LOLYPOP uses playback deadline as an important parameter for rate adaptation. The algorithm computes the probability of exceeding the playback deadline for different representations and selects the one with the highest bitrate that falls within the deadline. The selected bitrate is only allowed to play if it is not much different in quality from the previously played segment.

BOLA is the other recent QoE-based approach [**?**]. It models DASH rate adaptation as an optimization problem for an objective function whose prime objective is to increase playback utility and avoid stalling. BOLA also tries to reduce the number of changes in quality by delaying it while playing at a lower bitrate. BOLA reported a higher utility and fewer stalls than PANDA and ELASTIC.

2.5 Non-Normative Approaches

Georgopoulos et al. proposed an SDN-based method for maintaining fairness among the clients in the same network. The framework uses an OpenFlow-enabled switch [**?**] to check the bitrate available for each client using their MPD file. The switch divides the total throughput among the clients to have a comparable level of QoE [**?**]. Although using an SDN to monitor all clients in the network deviates the framework from the DASH standard [**?**].

In another approach, Akhshabi et al. also deviated from the standard by using server-based approach to reduce quality fluctuation [**?**]. The server observes the number of changes in a short interval of time and chooses a lower bitrate to control fluctuation.

De Cicco et al. [**?**] modeled an architecture where both the video player and the streaming server maintain a buffer and the system controls both buffer levels using the estimated throughput, thus requiring changes to the DASH streaming server.

Method	Configuration Paramters Used
Dash.js ABR [**?**]	Rich Buffer = 20 Sec
FESTIVE [**?**]	Buffer Tolerance = 0.8 × Bandwidth Stability and Efficiency Trade-off = 20
PANDA [**?**]	Buffer Convergence Rate β = 0.2
BBA [**?**]	Lower Reservoir = 90 sec Upper Reservoir = 24 sec
SARA [**?**]	Lower Reservoir = 2 segments Additive Increase = 5 segments Aggressive Switch = 10 segments Delayed Download = 12 segments
ELASTIC [**?**]	Proportional Coefficient = 0.01 Integral Coefficient = 0.001
Zhang et al. [**?**]	Reservoir = 4 Segments
Tian and Liu [**?**]	Trade-off Factor ∈ { 1, 5, 15, 20 }
Yin et al. [**?**]	Weighting Parameters λ =1
LOLLYPOP [**?**]	Transport Latency Threshold = 3 sec
BOLA [**?**]	Trade-off Factor (between utility and buffer occupancy) = 0.93

Table 2: Configuration parameters in different methods.

Zou et al. proposed a method for cellular networks where the client gets an accurate prediction of throughput through APIs. The method claims to improve the QoE significantly but accepts the fact that accurate prediction is an open challenge [**?**].

Zhang et al. propose a bitrate adaptation algorithm considering the variations in segment size at the same representation level to reduce the number of stalls and startup delay, assuming the size of each segment (in bytes) is known to the player [**?**]. This method requires transmission of additional segment size information in the MPD file and introduces additional overheads.

2.6 Configuration Parameters

Most of the methods in the literature require proper settings of configuration parameters, in the form of weights and thresholds. Table 2 lists the configuration parameters in the existing methods that required tuning. In many cases, the chosen values of the parameters are not well explained and seem arbitrary.

3 QUEUING MODEL

We now present how we model the DASH client as a queuing system. Using our model, we can estimate the buffer occupancy that the DASH video buffer would converge to given a segment bitrate and the network throughput. Our rate adaptation then selects the representation of the next segment to download so that the buffer occupancy converges to the ideal value.

A DASH client can be seen as an $M/D/1/K$ queuing system where video segments are arriving in a finite capacity queue, and the queue length (i.e., buffer occupancy) is measured as the video duration in seconds. The segment arrival is assumed to follow a Poisson distribution and segments are serviced with a deterministic service rate, one segment at a time, during video playback. Our proposed work does not schedule the download of segments. The client requests the next segment as soon as a previously requested

segment arrives in the buffer. In case the buffer is full, the DASH client stops downloading; if the buffer is empty, the playback stalls.

The $M/D/1/K$ model of DASH client contains only three parameters: (i) the segment arrival rate λ, which can be derived from the network throughput and segment size, (ii) the service rate μ, which depends on the playback speed and segment duration, and (iii) the capacity of the queue K, which depends on the buffer size configured in the player.

We now present the model in detail. Note that we use average value modeling [**?**] in this model where each variable models the average value of a random variable.

Let d be the duration (in seconds) of a video segment, r be the bitrate of the segment representation being downloaded, and b be the network throughput. The size (in bytes) of a video segment is rd, and the arrival rate of the segment λ is $b/(rd)$. Cao et al. [**?**] have shown that packet arrivals in a congested network follows Poisson distribution. Based on the split property and the addition property of Poisson distribution, the segment arrivals can be modeled with a Poisson distribution as well.

Let p be the playback speed of the DASH player, denoting how many seconds of video is consumed and played in one second. The service rate of the segments μ is therefore p/d. The queuing server utilization $\rho = \lambda/\mu$ is therefore b/rp. For the rest of this paper, for simplicity, we let $p = 1$ and $\rho = b/r$, which models the most common case where the user plays back the video at normal speed.

Using results from Brun et al. [**?**], we model the average queue length $X_{K,r,b}$ for an $M/D/1/K$ queue as $X_{K,r,b} = K - P_{K,r,b}$ where

$$P_{K,r,b} = \frac{\sum_{i=0}^{K-1} x_i}{1 + \frac{b}{r} x_{K-1}} \tag{1}$$

$$x_i = \sum_{j=0}^{i} \frac{(-1)^j}{j!} (i-j)^j \left(\frac{b}{r}\right)^j e^{(i-j)\frac{b}{r}} \tag{2}$$

We call $P_{K,r,b}$, which denotes the difference between the buffer capacity K and the average buffer occupancy $X_{K,r,b}$, as *buffer slack*. We use the estimated buffer slack, given K, r, b, in our proposed rate adaptation algorithm, which we outline in the next section.

4 QUETRA RATE ADAPTATION ALGORITHM

We now present our proposed algorithm, QUETRA (QUEuing Theory-based Rate Adaptation).

Let a given video be segmented into m segments and encoded into n representations, with bitrate values $R = \{r_1, r_2, ..., r_n\}$, with $r_i < r_j$ if $i < j$. QUETRA periodically estimates the network throughput and decides which bitrate (i.e., which representation encoded with that bitrate) to download for the next segment. We denote r_{i+1} as the next bitrate selected and b_{i+1} as the next estimated throughput. Further, let B_t be the buffer occupancy level at time t. The QUETRA algorithm to select r_{i+1} at time t is simple: *it chooses the bitrate such that the expected buffer slack $P_{K,r,b_{i+1}}$ is closest to B_t*, i.e.,

$$r_{i+1} = \arg\min_{r \in R} |P_{K,r,b_{i+1}} - B_t|,$$

breaking ties by favoring the higher bitrate.

The intuition behind the QUETRA is as follows. Consider the ideal case where segment bitrate r equals to the throughput b, which gives 100% utilization ($\rho = 1$). When $r = b$, one can show that x_i can be approximated with $2i + 2/3$ for large i [**?**], and so $P_{K,r,b_{i+1}}$ can be approximated with $K/2$. By choosing the bitrate such that the buffer occupancy is as close to the buffer slack as possible, QUETRA, in the hypothetical case where the bitrate value is continuous, would converge to the buffer occupancy to $K/2$, which is the ideal scenario.

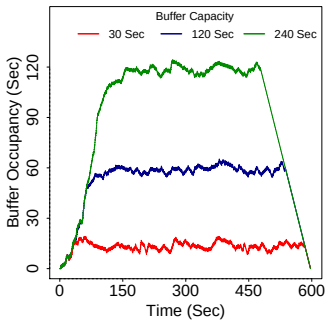

Figure 2: Example of a case where buffer occupancy in QUETRA converges to $K/2$.

In practice, the set of available bitrate values R is discrete, and the estimated throughput b fluctuates. By trying to equalize the buffer slack to buffer occupancy, QUETRA moves the buffer occupancy away from two undesirable cases: (i) an empty buffer that stalls the playback, and (ii) a full buffer that pauses the download, resulting in an OFF period in the traffic.

To illustrate, Figure 2 plots the buffer occupancy for QUETRA for different values of K when we run QUETRA on video $V1$ and network profile $P2$ (see Section 5 for details on $V1$ and $P2$).

An alternative algorithm is to pick a bitrate such that the $X_{K,r,b}$ is closest to $K/2$, without considering B_t. This naive algorithm, however, would not work as well. By minimizing $|P_{K,r,b_{i+1}} - B_t|$, we implicitly make QUETRA more aggressive when the buffer occupancy is further away from $K/2$. In the case of an empty buffer, such aggressiveness will choose a low bitrate to move the buffer occupancy level away from emptiness aggressively.

We will show, in Section 5, that despite being simple, the algorithm works well, across different scenarios, compared to other more rate adaptation algorithms in the literature.

4.1 Throughput Prediction and Smoothing

QUETRA uses both the current buffer occupancy and estimated throughput to decide which representation to download next. This decision does not explicitly try to reduce the changes in the representation chosen, the fluctuation of which can negatively affect the QoE. For QUETRA, the key to maintaining smooth changes in the representation is to maintain a smooth change in the estimated throughput. To achieve this, QUETRA estimates the throughput and runs the rate adaptation for every s segment (we use $s = 5$ in the evaluation) instead of every segment.

Some existing throughput-based rate adaptation methods (e.g., `Dash.js` [**?**], FESTIVE [**?**], PANDA [**?**]) use different throughput prediction and smoothing methods. In addition to comparing QUETRA to other methods without any prediction or smoothing (simply

using the last measured throughput), we also evaluated the QUETRA using different throughput smoothing and prediction methods to understand the effect these methods on QUETRA. We evaluated the following six throughput estimation methods in combination with QUETRA.

Last Throughput. The simplest approach is to assume that the throughput does not change, and set b_{i+1} to the measured throughput of the most recently downloaded segment. This is the default throughput estimation method in QUETRA.

Average Throughput. This approach, adapted by the `Dash.js` player [**?**], uses the mean of last three throughput samples as b_{i+1}.

Exponential Moving Average (EMA). Here, the estimated throughput is computed using a weighted sum of most recent throughput measured $\hat{b}_i$ and the last predicted value, using a constant smoothing gradient α:

$$b_{i+1} = (1-\alpha)b_i + \alpha\hat{b}_i.$$

Lowry et al. [**?**] reported that $\alpha = 0.1$ is efficient for detecting a small shift in the average vector and has a longer average run length. We use $\alpha = 0.1$ in our evaluation.

Gradient Adaptive EMA. This approach extends the EMA with a dynamic smoothing gradient (α_i) [**?**] that varies according to:

$$\alpha_i = \alpha_{i-1}^{M_{norm}(i)/M_{inst}(i)},$$

where $M_{inst}(i)$ is the slope of change in the last two throughput measured and $M_{norm}(i)$ is the mean gradient of throughput over time. These two variables are calculated as

$$M_{inst}(i) = \frac{|\hat{b}_i - \hat{b}_{i-1}|}{T_i - T_{i-1}},\ M_{norm}(i) = \frac{\bar{b}}{T_t - T_0}, \tag{3}$$

T_i and $\hat{b}_i$ are the playback time and measured throughput for the i-th segment, and $\bar{b}$ is the mean measured throughput of all segments. As in the EMA approach, we initialized $\alpha_0 = 0.1$ in our evaluation.

Low Pass EMA. This approach is a variation of the Gradient Adaptive EMA [**?**] where the gradient changes according to:

$$\alpha_i = \frac{1}{1 + M_{inst}(i)/M_{norm}(i)}$$

As above, we set $\alpha_0 = 0.1$ in our evaluation.

Kaufman's Adaptive Moving Average (KAMA). KAMA [**?**] uses a smoothing constant in a moving window of last 10 values. To estimate the throughput of the segment $i+1$, it first calculates the efficiency ratio e_i as the ratio of *change* over *volatility*:

$$e_i = \frac{|b_i - b_{i-9}|}{\sum_{x=i-9}^{i} |b_{x+1} - b_x|}$$

Volatility captures the total noise in the path to attain that change in throughput. The smoothing constant (sc) is then calculated

$$sc_i = (e_i(fsc - ssc) + ssc)^2$$

with fastest smoothing constant (fsc), set to 2/3, and slowest smoothing constant (ssc), set to 2/31. The window size is reduced to i, if $i < 10$. The final b_{i+1} is calculated with:

$$b_{i+1} = b_i + sc_i(\hat{b}_i - b_i). \tag{4}$$

There are many other algorithms for smoothing throughput estimation, including Moving Average Convergence Divergence (MACD) [**?**], which considers the subtraction of two parametrized EMAs, and Sliding Percentile (SP) [**? ?**], which is similar to EMA but better handles outliers.

5 EVALUATION

To evaluate QUETRA, we compare it against several representative methods for rate adaptation. For buffer-based approach, we compare against the method proposed by Huang et al. [**?**], which we denote as BBA in the rest of this paper. For throughput-based approach, we compare QUETRA against the default algorithm in `Dash.js`, which we denote as ABR [**?**]. We also compare against ELASTIC [**?**], which, like QUETRA, combines buffer-based and throughput-based approaches and tries to converge the buffer occupancy to $K/2$, and BOLA [**?**], which is a QoE-based approach.

We evaluated the schemes above using three buffer occupancy settings. The first, 30/60, is the default in `Dash.js` where K is 30s if the video is shorter than 10 minutes and K is 60s otherwise. The second setting has a buffer occupancy of 240s, as used by Huang et al. [**?**] to evaluate BBA. Finally, we use $K = 120$s as the intermediate buffer capacity between 60s and 240s.

5.1 Implementation

We implemented QUETRA, BBA, and ELASTIC as alternative rate adaptation algorithms in `Dash.js` [**?**] v2.1.1. ABR and BOLA are already implemented in `Dash.js`.

QUETRA. In QUETRA implementation, for efficiency reason, we precomputed the buffer slack (Equation 1) for a given ρ, which we quantized at the interval of 0.01. We further approximate the buffer slack to be 0 for $\rho < 0.5$ and to be K for $\rho > 1.2$.

BBA. The original BBA algorithm uses a buffer capacity of 240s, with a lower reservoir of 90s and upper reservoir of 24s. We keep the same ratio of the reservoirs, i.e., $0.375K$ and $0.1K$ when evaluating BBA using different values of K. For fair comparison, we introduce a lower reservoir of $0.375K$ to QUETRA as well.

ELASTIC. We implemented PID using the proportional coefficient of 0.01 (same as [**?**]) but adjusted the integral coefficient to 0.0001 to avoid obtaining negative bitrate value from the controller.

5.2 Video Samples

We use seven video samples, denoted $V1$ to $V7$, of different characteristics for evaluation. Table 3 summarizes the video samples. $V1$ to $V4$ are taken from [**?**]; $V5$ to $V7$ are taken from [**?**] [**?**] [**?**] respectively. The video samples have been encoded into either 3, 5, or 10 bitrate levels and different segment durations. The segment duration affects the frequency of throughput measurement and rate adaptation decision. Keeping different values for segment duration tests the robustness of QUETRA. $V1$ and $V2$ both have approximately equal gap between bitrates. $V3$ and $V4$ both have two bitrates that are closer than the other. $V5$ has a shorter duration. $V6$ having 10 levels along with $V7$ have a relatively closer level of bitrates, which may cause more changes in bitrate level.

5.3 Network Profiles

We use four network profiles in the evaluation, denoted $P1$ to $P4$. $P1$ and $P2$ are taken from DASH Industry Forum Guidelines [**?**]. $P3$ and $P4$ are taken from HSDPA dataset [**?**]. There are five levels of throughput in $P1$ and $P2$ (5000 kbps, 4000kbps, 3000 kbps, 2000

Video Sample	Representation bitrates (kbps)	Duration (Sec)	Segment Size (Sec)
$V1$	1200, 2200, 4100	594	2
$V2$	1622, 2313, 4077	654	5
$V3$	1198, 1974, 4103	654	2
$V4$	1143, 1795, 4140	594	5
$V5$	350, 600, 1000,2000, 3000	244	4
$V6$	230, 331, 477, 688, 991, 1427,2056, 2962, 5027, 6000	634	3
$V7$	300, 700, 1500, 2500, 3500	653	2

Table 3: Characteristics of Video Samples.

Network profile	Inter-variation duration (sec)	Min (kbps)	Max (kbps)	Type
$P1$	30	1500	5000	High-Low-High
$P2$	30	1500	5000	Low-High-Low
$P3$	1	241	5876	Trace
$P4$	1	3	3344	Trace

Table 4: Characteristics of Network Profiles.

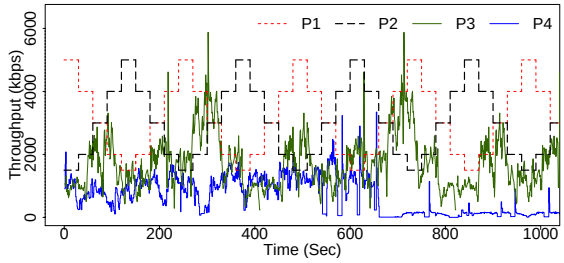

Figure 3: Different network profiles used in the experiment.

kbps, and 1500 kbps), varying at the interval of the 30s. $P1$ follows a high-low-high pattern; $P2$ follows a low-high-low pattern. $P1$ favors those algorithms that need buffer occupancy build up to raise the bitrate as throughput is high in the beginning. $P2$ favors those algorithms that are independent of buffer occupancy. Throughput in $P3$ and $P4$ are collected from moving train and car, sampled every second. An algorithm that increases download bitrate too aggressively would have playback stalls in $P3$ and $P4$. In addition, $P4$ has a period of significantly low throughput (minimum of 3 kbps) and thus, even a conservative algorithm cannot avoid stalls. We repeated the network profile if its duration is shorter than the video playback duration. Table 4 shows the details of the profiles and Figure 3 shows their variations over time.

5.4 Experimental Setup

We use a Web server running Apache to serve the video segments to a Web client, running `Dash.js` with one of the above rate adaptation algorithm. The server and client run on two different hosts, connected by a router. We run the network emulator `netem` [**?**] to control the network throughput according to one of the profiles.

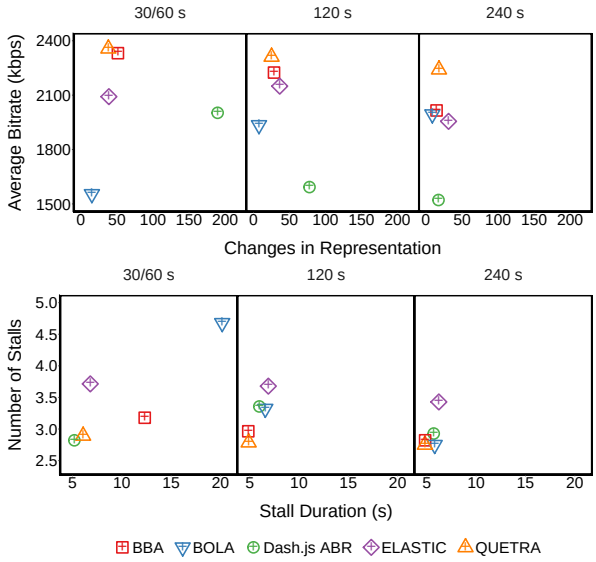

Figure 4: XY plot of bitrate versus changes in representation, and number of stalls versus stall duration for different algorithms.

5.5 Performance Metrics

We evaluated the rate adaptation methods using the following metrics. To measure the overall video quality, we compute the average bitrate of representations downloaded over all segments. We also count the number of changes in quality (i.e., changes in representation), and the magnitude of changes in quality (as difference in bitrate values). Frequent changes in video quality is shown to reduce the quality of experience. In addition, we measure the number of stalls (buffer empty), and the duration of each stall.

We also apply the QoE model proposed by Yin et al. [**?**], which combines the various performance metrics into one QoE value:

$$QoE = \sum_{n=1}^{N} q(R_n) - \lambda \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)| - \mu T_{stall} - \mu_s T_s. \quad (5)$$

Here, N is the total number of segments; q maps a bitrate to a quality value; T_{stall} is the total stall duration during the playback and T_s is the startup delay. As in [**?**], $q(\cdot)$ is the identity function, $\lambda = 1$, μ and μ_s are set to the maximum bitrate of the video sample.

5.6 Results: QUETRA vs. Rest

We now present the experimental results comparing QUETRA to ABR, BBA, ELASTIC, and BOLA. We first show the results for each method and buffer capacity, averaging the performance metrics over all 28 combinations of video samples and network profiles.

Average bitrate and changes in representation. Figure 4 (top) shows the XY plot for the average bitrate vs.number of changes in representation for different buffer capacities. Points closer to the upper left yield better performance in both dimensions.

ABR has 190, 77, and 17 changes in representation for 30/60s, 120s, and 240s buffer capacity respectively, which are higher than others. Since ABR is throughput-based and it estimates the throughput after downloading every segment, it suffers from fluctuating

quality levels, especially in network profiles $P3$ and $P4$. BOLA has the least number of changes in representation (between 7 to 15) in all buffer capacity settings. BOLA is conservative and tends to keep downloading low bitrate representation, BOLA leads to lower video quality, especially for buffer capacity of 30/60s (1554 kbps, vs. 2356 kbps for QUETRA). QUETRA has better average bitrate compared to other methods, achieving up to 11% improvement over BBA on average at BBA's default buffer capacity of 240s, and comparable changes in representation to BBA and ELASTIC.

To see why, recall that BBA maps buffer occupancy linearly to the bitrate of the available representations. As buffer capacity increases, it takes longer for BBA to fill up the buffer to reach the level to download a higher bitrate. As such, the average bitrate declines with the increase in the buffer capacity. As a result, BBA has a lower average bitrate than QUETRA (2016 kbps compared to 2241 kbps with 240s buffer) and fewer changes in representation than QUETRA (14 compared to 18). In other words, BBA sacrifices video quality for fewer changes in representation.

Number and duration of stalls. Figure 4 (bottom) shows the XY plot of number of stalls vs. the total stall duration for each method for each buffer capacity, averaged across all 28 combinations of network profiles and video samples. Data points closer to the lower left corner perform better on both dimensions.

ELASTIC has 3.4 to 3.7 stalls on average, which is higher than other methods in general. ELASTIC does not use a reservoir that prevents stall. ELASTIC, however, perform consistently over different buffer capacity, as it tries to converge the buffer occupancy to $K/2$, similar to QUETRA. BOLA has the highest stalls duration of 20s at buffer capacity of 30/60s, as it restrains from changing the quality even when downloading at the highest bitrate, leading to buffer starvation. BBA has a longer stall (over 12s) at buffer capacity of 30/60s compared to 120s and 240s. QUETRA, on the other hand, has the fewest stalls and shortest stall duration on average, over the three buffer capacity settings. QUETRA's strategy to try and converge the buffer occupancy level towards $K/2$ works well.

QoE. Figure 5 compares the QoE for all five methods of rate adaptation over three different buffer capacities, computed using Equation 5 and averaged over all 28 combinations of video samples and network profiles. Compare to other methods, QUETRA exhibits the highest QoE of 13% to 140%, 7% to 97%, and 9% to 51% for the 30/60s, 120s, and 240s buffer respectively.

OFF Period. The `Dash.js` player stops downloading when the buffer is full, resulting in an OFF period that could lead to incorrect estimation of throughput for other clients [**?**]. QUETRA, however, tries more aggressively to move the buffer occupancy away from K as the buffer occupancy increases. As such, we expected QUETRA to reach the buffer full condition less frequently.

Figure 8 shows the total buffer full duration for all the compared methods, again averaged from all video samples and network profiles. BOLA has the shortest duration for all buffer capacities, as it turns off downloading before the buffer becomes full, due to its rate adaptation policy. BBA has the next shortest durations overall. Since BBA is buffer based, it sometimes selects representation with a bitrate higher than throughput. This leads to lower buffer occupancy in general.QUETRA encounters fewer buffer full events than ELASTIC and ABR. Since the duration of buffer full for QUETRA is low, we are expecting fewer OFF period and thus a higher fairness

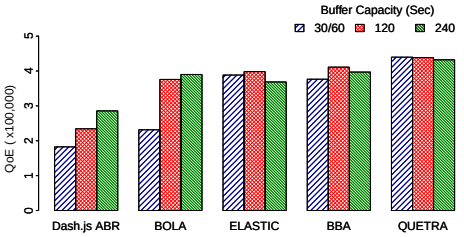

Figure 5: QoE for different methods.

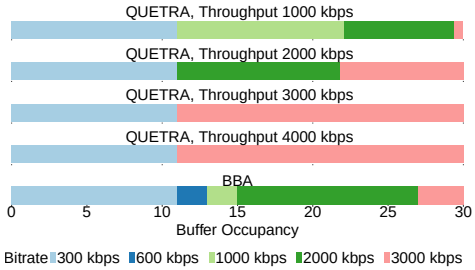

Figure 6: Change in bitrate with buffer occupancy for video V5 for BBA (irrespective of throughput) and QUETRA with different throughput having 30s buffer capacity.

in the average bitrate played by multiple clients. To verify this, we ran stream $V1$ to 10 clients with 10 times the bandwidth of network profile $P1$. The Jain's fairness index [**?**] for average bitrate played across different clients obtained is 0.998.

Specific Cases. The results so far examine the average performance of the rate adaptation algorithms, over all video samples and network profiles. For the rest of this section, we highlight two specific cases (i) video samples $V5$ [**?**], (ii) stall duration for BBA and BOLA in profile $P4$ and video sample $V2$.

Figure 7 is similar to Figure 4, but only the results for $V5$ (averaged over $P1 - P4$) are shown. QUETRA has almost consistent average bitrate as the buffer capacities increase (1971 kbps - 2190 kbps). BBA, however, performs worse as the buffer capacity increases from the 30s to 240s (drops 45% from 2117 kbps to 1200 kbps). $V5$ has a duration of 244s, not uncommon for videos available on social media, which is close to the buffer capacity 240s. Figure 6 visualizes the selected bitrate at different values of buffer occupancy for QUETRA (under different throughput) and BBA (irrespective of throughput) for $V5$ having 30s buffer capacity. As shown in the figure, QUETRA chooses a bitrate lower than or equal to the throughput at lower buffer occupancy but aggressively increases the bitrate if buffer occupancy is high, except for the lower reservoir where it chooses the minimum bitrate. BBA, on the other hand, linearly maps the bitrate of representation to the buffer occupancy. Comparing the figure when throughput is 3000 kbps, for instance, one can see that QUETRA downloads segments at much higher quality than BBA. For BBA, by the time buffer occupancy reaches the sufficient level to download the highest bitrate representation, a major portion of $V5$ has already been played. This behavior leads to lower average bitrate and lower utilization of throughput.

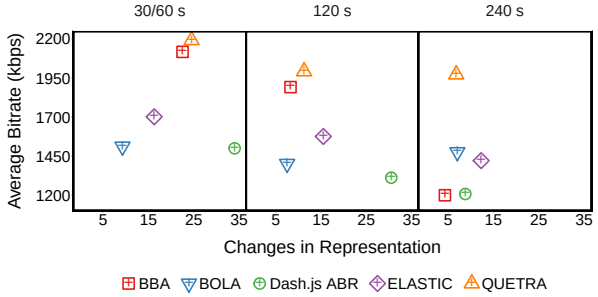

Figure 7: XY plot of bitrate versus changes in representation for video V5.

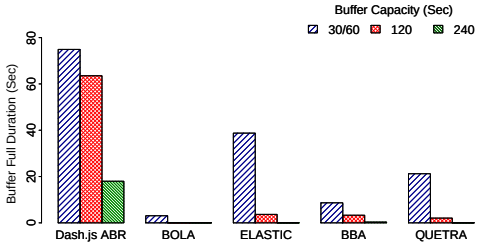

Figure 8: Duration of buffer full for different methods.

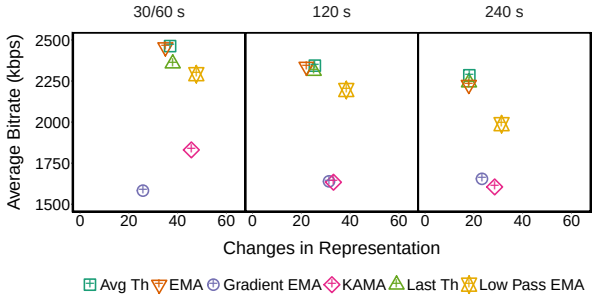

Figure 9: XY plot of bitrate versus changes in representation for different throughput prediction methods.

The second special case we highlight is for network profile $P4$ and video sample $V2$. $P4$ has the lowest average throughput among all four profile; $V2$ has the highest segment duration of 5s and higher gaps between the levels. As a result, $P4$ and $V2$ are a challenging combinations for all rate adaptation methods. In particular, this specific combination contributes to the significantly longer stalls (on average) that we see in BBA and BOLA (12.3s and 20.2s respectively). Upon closer inspection, we found that $P4$ has a downtime after 11 minutes, where the throughput falls to between 20-200 kbps. While most of the methods finished the download of all $V2$ segments before reaching that region, except for BBA and BOLA in 60s buffer capacity. When BBA's buffer occupancy is high, it switches to a higher bitrate, which delays the download if the throughput drops, which happens often in $P4$ due to the fluctuation in throughput; BOLA frequently stops the segment downloads to limits the fluctuation in the video quality. As a result, both BBA and BOLA are still downloading segments of $V2$ after 11 minutes in this specific case, causing long stalls. For a fair comparison, we recomputed the average stall duration excluding $V2$ and $P4$ for 30/60s buffer. The result shows that the stall time for BBA and BOLA reduced to 3.7s and 5.1s respectively, which is comparable to QUETRA (4.3s).

5.7 Results: Effects of Buffer Capacity

Buffer capacity is a critical parameter that would affect the performance of rate adaptation. In a system with finite buffer, buffer overflow occurs less often with larger buffer. A larger buffer, however, is unable to utilize the possible future increase in throughput. Our experiment shows non-negligible effect of buffer capacity on the performance of some methods. Figure 8 shows that the buffer-full duration for 30/60s buffer is 3-40 times more than 240s buffer for all methods. Figure 4 shows that average bitrate for `Dash.js` ABR and BBA at 240s buffer is 31% and 15% less than that at 30/60s buffer. Since BOLA aborts segment download to control the changes in representation, its bitrate increases with the increase in buffer capacity, from 1554 kbps to 1994 kbps when buffer increases from 30/60s to 240s. On the other hand, QUETRA and ELASTIC have only 5% variation in average bitrate as buffer capacity increases, because they try to maintain the buffer occupancy at $K/2$.

5.8 Results: Effects of Throughput Estimation

Figure 9 compares QUETRA using the default last throughput approach with other throughput prediction method. EMA and the average of the last three throughputs increase the average bitrate up to 4.5% and 4.2% respectively for different buffer settings, compared to the last throughput method. Gradient-based EMA, low pass EMA, and KAMA are conservative due to consideration of larger time frame for smoothing, and therefore predict a lower throughput than the actual value. A lower predicted value calls for the selection of a lower bitrate, fills up buffer faster, which in turn calls for the selection of higher bitrate. The contradictory evaluation of the situation leads to more changes in representation.

6 CONCLUSION

We presented QUETRA, a DASH rate adaptation algorithm for DASH that is informed by a fundamental relationship between selected representation bitrate, estimated throughput, buffer occupancy, and buffer capacity. By exploiting this relationship obtained through a queuing model, QUETRA avoids the complexity and the needs for parameter tuning that plagues many existing methods, yet performs better than the existing methods under different scenarios. Furthermore, we show that buffer capacities can have significant impact on the performance of existing rate adaptation algorithms, when viewing videos of different durations. QUETRA, on the other hand, performs consistently across different buffer capacities, and on average, its QoE is 140% better than the default in `Dash.js`, the reference player from DASH Industrial Forum, and is 7% better than the closest competitor, BBA, which is proposed by Netflix.

ACKNOWLEDGMENTS

The authors would like to thank Y. C. Tay for his valuable feedback as well as Hancong Kong and Bentaleb Abdelhak for their contributions during the brainstorming sessions. The research is supported

by Singapore's Ministry of Education Academic Research Fund (Grant T1 251RES1506).