Multimedia Systems (2018) 24:327–340
https://doi.org/10.1007/s00530-017-0554-9

REGULAR PAPER

On the trajectory of video quality transition in HTTP adaptive video streaming

Yusuf Sani[1] · **Andreas Mauthe**[1] · **Christopher Edwards**[1]

Received: 4 June 2016 / Accepted: 29 May 2017 / Published online: 17 June 2017

Abstract HTTP adaptive video streaming matches video quality to the capacity of a changing context. A variety of schemes that rely on buffer state dynamics for video rate selection have been proposed. However, these schemes are predominantly based on heuristics, and appropriate models describing the relationship between video rate and buffer levels have not received sufficient attention. In this paper, we present a QoE-aware video rate evolution model based on buffer state changes. The scheme is evaluated within a real-world Internet environment. The results of an extensive evaluation show an improvement in the stability, average video rate and system utilisation, while at the same time a reduction in the start-up delay and convergence time is achieved by the modified players.

Keywords HTTP adaptive streaming · Adaptive bitrate selection

1 Introduction

Nowadays, a typical video streaming service is expected to serve a variety of platforms, e.g. smartphones, web browsers, TVs. Each of these platforms has specific requirements with respect to transmission and video quality. Moreover, the environment within which most of the video streaming clients operate is both unreliable and varies over time. However, regardless of the access device, users want the best viewing experience possible. HTTP adaptive streaming (HAS) is the most successful technology so far that allows content providers to cater for the requirements of the multitude of devices and contexts. The process through which a HAS client chooses a video rate is called adaptive bitrate selection (ABR). The first generation of ABRs relied on throughput estimation and selected the highest video rate lower than the measured throughput [14]. This is based on the work of Wang et al. [22] that showed if the available TCP throughput is twice the bitrate of the video plus a few seconds of start-up delay TCP can ensure an acceptable video streaming experience. It later became clear that throughput estimation alone is not a sufficient parameter for designing efficient ABR since an accurate bandwidth estimation above the HTTP layer is difficult to achieve [6]. Consequently, any video rate selection algorithm that solely depends on such a relatively inaccurate estimate results in unnecessary rebuffering events [7], an undesirable variability of video rate [6] and sub-optimal video quality [6].

Various attempts have been made to improve some of the identified issues of throughput-based ABRs by supplementing throughput measurements with information about the playback buffer [1, 10, 21]. Using buffer occupancy as a factor in video rate selection has developed from regarding buffer state changes as a complementary factor in making a rate selection decision [10, 21] to employing it as the sole metric [7, 8]. Though, whatever factor an ABR primarily relies on, it is difficult to build an ABR that maximises Quality of Experience (QoE) without taking buffer state changes into consideration.

Communicated by P. Shenoy.

✉ Yusuf Sani
y.sani@lancaster.ac.uk

Andreas Mauthe
a.mauthe@lancaster.ac.uk

Christopher Edwards
c.edwards@lancaster.ac.uk

[1] School of Computing and Communications InfoLab21, Lancaster University, Lancaster LA1 4WA, UK

It has been shown in [7] that by basing ABR solely on playback buffer occupancy a client can choose the highest quality level without the fear of an increase in rebuffering. However, a linear increment of video rate, as used in [7], may not always enhance the QoE. For example, when the video quality is relatively high an increase in the current video rate does not necessarily translate into an improvement in the user-perceived quality, since as shown in [3, 17] a stage is always reached when users do not find any further increase beneficial. Furthermore, the buffer management model employed in [7, 8] artificially separates the ramping-up period from the steady state, which results in significant loss of quality.

To ensure that the video rate evolves in a way that optimises QoE, there is a need for a rate evolution map that captures the desirable pattern of video quality transition. This paper will concentrate on the following research question: *If we have a QoE-aware model of the relationship between the playback buffer state changes and the available video rates, how much improvement in user-perceived video quality can be achieved?*

In order to answer this question, the paper first identifies the patterns of quality changes that are known to affect QoE, and then develops a QoE-aware model of the rate map that combines all stages of video rate evolution, while incorporating an optimal number of patterns that improve user-perceived video quality. In this paper, we restrict ourselves to the set of objective QoE metrics that are known to have an impact on the user experience in adaptive video streaming, which are video freeze, average video rate, start-up delay, video quality convergence, and system utilisation. Note the presented model is descriptive. In other words, it is just a declarative representation of the relationship between the video rate and the buffer state changes and not a standalone algorithm. Hence, the paper demonstrates how the proposed model can be used in practical systems by modifying selected throughput-based and buffer-based ABRs. The modified algorithms are then integrated into the implemented players. Extensive experimentation over the Internet using both wired and wireless connections shows the performance of the scheme.

The rest of the paper is structured as follows: Sect. 2 presents background and the related work; Section 3 discusses the QoE-aware evolution trajectory and the system model; Section 4 details the methodology and experimental set-up used; Section 5 covered result presentation and finally the paper is wrapped with a conclusion in Sect. 6.

2 Background and related work

HAS services usually divide a video file into a number of chunks of equal temporal size with each chunk encoded in multiple bitrates. A client progressively requests a relevant chunk. The bitrate of the requested chunk is based on the client's measurement of the available resources. Throughput-based ABRs select a chunk with the highest video rate lower than the measured throughput [20]. When the throughput changes, the buffer level may be used as an indicator of whether to increase, decrease or stay with the current video rate [10, 16]. However, those ABR schemes that solely rely on buffer occupancy are called buffer-based ABRs.

Throughput-based algorithms assume that the throughput of a recently downloaded chunk is a rough estimate of the current network condition. But due to short-term throughput fluctuations, as a result of the TCP congestion control mechanism and the difficulty in accurately estimating throughput above the HTTP layer, throughput-based algorithms use a weighted average to smooth out the estimated network capacity [1]. However, using historical data is known to reduce the responsiveness of an algorithm [1]. A number of measurement studies have shown that throughput-based algorithms are unstable [13], unnecessarily rebuffer [6], request sub-optimal video rates [1], and are unfair [6].

A significant amount of research is focused on how to improve the accuracy of the TCP throughput measurement of a typical ABR scheme. The authors of [13] propose a probe and adapt technique. The algorithm mimics the congestion control of TCP but at the application layer. It uses TCP throughput as an input only when it is an accurate indicator of the fair share of bandwidth. In the same vein, the authors of [21] use machine learning techniques to predict the achievable throughput by using network state information.

In order to improve some of the downsides of throughput-based services, various researchers use the buffer level as a feedback signal to complement throughput estimation [16]. Tian and Lui [21] went further by using the playback buffer state change as the key feedback signal. Huang et al. [7, 8] propose an algorithm that completely relies on buffer occupancy for the video rate selection decisions. They are motivated by the fact that the end-to-end capacity can be indirectly derived from buffer state changes. However, the model employed separates the buffering from the steady-state phase, which obviously creates a disconnected flow. Furthermore, at the start-up period (called reservoir), only the lowest available video rate is downloaded. Hence, there is a substantial loss in video quality at the beginning of the streaming session. During the ramping-up period, the video rate is linearly incremented. However, in [23] it has been shown that the probability of buffer starvation decreases exponentially with respect to the initial buffer level. Therefore, a linear evolution of the video rate, when ramping-up, will unnecessarily prolong the convergence time. Furthermore, it is also worth noting that a constant and continuous increment of the video rate may not always enhance QoE. In [3] it was demonstrated that when the video quality is high an increase in the current rate does not necessarily translate into an improvement in the user-perceived quality. Nevertheless, the paper made an important observation: when the buffer is used as the main factor of an ABR the trade-off between video quality and the amount of rebuffering is unnecessary.

Earlier, Mok et al. [17] have studied the effect of video rate transition on QoE. They found that a sudden drop in video rate has a negative impact on user experience. To improve QoE, they opted to switch down the video rate to an intermediate level even when the target video level is lower. The problem with this design is that the user will be downloading higher bitrate than the download rate, hence increasing the risk of buffer starvation, especially since both the intermediate level and the maximum buffer size are heuristically determined. While this work narrowed its investigation to a pattern, in our previous work [18], we presented a bio-inspired model that pays attention to the whole sequence of the trajectory through the space of all possible system states.

3 System modelling and implementation

3.1 Quality evolution trajectory

A change in video quality level affects user experience. However, the degree of the impact and whether it is positive or negative depend on the pattern of the transition. In this section, a QoE-aware video quality evolution is derived.

At any given time t after the video streaming has started, the buffer may contain an array of chunks of different quality levels. However, chunks of different video rates generally have different sizes in bytes. We shall assume that all chunks contain an equal amount of video time V in seconds. Since there is no direct mapping between buffer size in bytes and video time, we calibrate buffer in time, i.e. by the second. This has also been assumed in [7, 21].

At the beginning of a streaming session ($t = 0$), a server presents to a client a set of different video rates $Q = \{q_0, q_1, q_2, \ldots, q_n\}$, with $|Q| = n + 1$. Let us suppose $q_0 < q_1 < \cdots < q_n$. Furthermore, let us assume that the quality of the video perceived by a user increases with video rate. Therefore, q_0 is the minimum quality level (referred here also as $q_{\min}$) and q_n is the maximum available quality level (called $q_{\max}$). Suppose B_t is the buffer occupancy at time t and $B_{\max}$ is the maximum buffer all measured in seconds. Let $\hat{c}_t$ denote the estimated throughput at time t with $C(t)$ being the system capacity (i.e. $\hat{c}_t \leq C_t$).

Usually, after the receipt of the media description file at t_0, the play-out buffer is empty ($B_0 = 0$), a client starts requesting a chunk with quality level $q_{\min}$ in order to minimise the start-up period. However, a prolonged download of $q_{\min}$ will negatively affect the user experience. Hence, using a video rate selection function $R(t)$, the client should immediately start a gradual improvement of the bitrate of the requested chunks as soon as it receives the initial chunk, such that the video rate of chunk $i + 1$ requested after successfully downloading chunk $i \geqslant 1$ with video rate q_k , where $\{k \in n : 0 \leq k \leq n\}$, is $q_{k+1} = \alpha q_k$, where $\{\alpha : 0 < \alpha \leq 1\}$.[1] Suppose that the download of chunk $i + 1$ with starts at time t^{s} and finishes at t^{e}. Let us also assume that the rate at which the client's requested video rate evolves with respect to time $\mathrm{d}R(t)/\mathrm{d}t$ is $g'(R)$. Assuming that $C(t) \geqslant R(t) = q_{\max}$. In other words, we have sufficient network capacity to cover for highest available video rate. With this assumption, a client can continuously increase it video until it reaches the highest available. Simply put, $g'(R)$ is positive at any time after the start of streaming except when $R(t) = q_{\max}$ and $B_t = 0$ in which case $g'(R) = 0$.

To ensure that the client gets its fair share of the available bandwidth, we rely on the recommendation of [7, 8], which states that the highest rate is selected only when a buffer is full or nearly full (i.e. $R(t) = q_{\max}$ when $B_t \to B_{\max}$). This will ensure that provided the highest video rate is not reached, the OFF period will not be activated (i.e. we use a back-to-back download). In other words, the system starts at minimum video rate when the buffer is empty, and continuously increment its video rate in such a manner that the video increment terminates when the buffer is full.

To avoid high amplitude variation (e.g. an abrupt drop of the video quality), which is known to be detrimental to QoE [17, 25] and to minimise the negative impact of recency effect [5, 9], transition decision to q_{k+1} should depend on q_k. Furthermore, since users are not known to be appreciative of an increase in the video quality when the video rate is relatively high [3] we recommend a non-linear $g'(R)$. In fact, Yamagishi and Hayashi [24] have shown that, given a set of linearly incremented video rates, the saturation begins to take effect at about halfway through the available video rates, when they studied the relationship between video rate and the subjective video quality. Therefore, we suggest that after reaching $q_{\max}/2$ a client should start reducing the rate at which it

[1] It is worth noting while other researchers like [4] and [13] have advocated for the use of a gradual increment of the video rate at the beginning of a streaming session, for the purpose probing the network capacity, our scheme differs in the sense that we use the increment to control buffer flow.

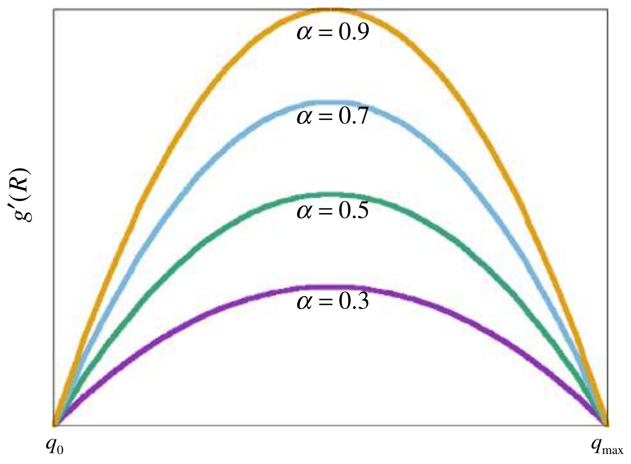


Fig. 1 Derived trajectory of video quality evolution

increases its video quality. Figure 1 summarises the trajectory of $g'(R)$ that we deduced from the foregoing discussion. The path is concave pinned at two points $q = 0$ and $q = q_{\max}$ with amplitude at $q_{\max}/2$. This pattern can easily be described by a quadratic function with $q = 0$ and $q = q_{\max}$ and a positive constant α. It should be noted that the value of the constant α determines the vertex of the parabola. In other words, it determines the maximum value the $g'(R)$ can attain (see Fig. 1 a number of trajectories using different values of α are plotted).

$$g'(R) = aq(q_{\max} - q) \tag{1}$$

3.2 Modelling

Given the desired video evolution path just derived, we next formulate a video rate prediction model.

3.2.1 Continuous rate

We first look at a case where $R(t)$ results in any value between $q_{\min}$ to $q_{\max}$. With this assumptions, we can model $R(t)$ as a continuous function.[2]

Clients usually infer $C(t)$ from $\hat{c}(t)$ for the purpose of rate selection. Suppose $c(t_i)$ is the estimated throughput when $t = t_i$ derived from the average of h number of chunks calculated thus:

$$c(t_i) = \frac{1}{t_i - t_{i-h}} \int_{i-h}^{h} \hat{c}(t)\mathrm{d}x.$$

Let us assume that a HAS client requests chunk i immediately after chunk $i - 1$ is completely downloaded except when the buffer is full. In which case it waits for V seconds (chunk size) before sending a request. Except during the off period, the playback buffer drains at the one buffer second every real time second and fills at $C(t)$ / $R(t)$, therefore the rate at which buffer changes is

$$\frac{\mathrm{d}B(t)}{\mathrm{d}t} = \frac{C(t)}{R(t)} - 1. \tag{2}$$

In most contexts, $C(t)$ is time-varying; therefore, if the client is to avoid buffer starvation, the output of $R(t)$ has to adapt to this changing environment with time.

$$\frac{\mathrm{d}R(t)}{\mathrm{d}t} = \frac{\mathrm{d}R(t)}{\mathrm{d}B} \cdot \frac{\mathrm{d}B}{\mathrm{d}t}. \tag{3}$$

We want $R(t)$ to closely match $C(t)$, in which case we do not expect the buffer to change often, that is, $\frac{\mathrm{d}B(t)}{\mathrm{d}t} \approx 0$. From Eqs. (1) and (3)

$$g'(R) = \frac{\mathrm{d}R(t)}{\mathrm{d}t} = \frac{\mathrm{d}R(t)}{\mathrm{d}B} = \alpha q(q_{\max} - q), \tag{4}$$

$$\frac{\mathrm{d}R(t)}{q(q_{\max} - q)} = \alpha \mathrm{d}B,$$

after simplification using partial fraction method and using $R(t) = q$ we have

$$\int \frac{1}{q} dq + \int \frac{1}{q_{\max} - q} dq = \int \alpha q_{\max} \mathrm{d}B. \tag{5}$$

by integrating equation (5) we have

$$\ln q - \ln |q_{\max} - q| = \alpha q_{\max} B + e. \tag{6}$$

The streaming starts with a minimum video rate; therefore, $q = q_{\min}$ and $B = B_0$. Using this information *e can be evaluated as thus:*

$$e = \ln \frac{q_{\min}}{q_{\max} - q_{\min}} - \alpha q_{\max} B_{t_0}. \tag{7}$$

Substituting Eq. (7) into (6) and simplifying it, we have:

$$\ln \frac{q}{q_{\max} - q} - \frac{q_{\min}}{q_{\max} - q_{\min}} = \alpha q_{\max}(B_t - B_{t_0}). \tag{8}$$

Finally, solving for q and $(B_t - B_{t_0} \approx B_t)$, since $\{B_0 : 0 < B_0 \leq V\}$

$$R(t) = \frac{q_{\max}}{1 + [\frac{q_{\max}}{q_0} - 1]\mathrm{e}^{-\alpha q_{\max} B_t}} \tag{9}$$

3.2.2 Discrete rate

By dropping our assumption about the continuous nature of video rates, the video quality has to be chosen from a finite discrete set. Furthermore, q can only move from one valid value to another. We assume the quality level change

[2] This is without loss of generality, in fact, in the next section we drop this assumption.

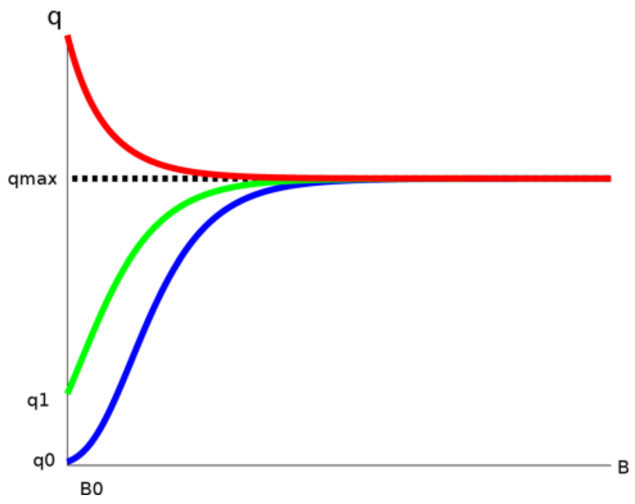


Fig. 2 Evolution of the $R(t)$ in both Continuous and Discrete mode.

is done only between adjacent video rates, that is, q_k can only move either to q_{k-1} or q_{k+1}.

The model is now modified to reflect this. To change a video rate a buffer must have grown or contracted by a certain buffer distance. Precisely, to change the quality level we need $\Delta B_k = R^{-1}(q_{k+1}) - R^{-1}(q_k)$. When ΔB_k is positive the quality level is going to be increased and when it is negative the quality level is reduced. When $R(B) = q_{\max}$ $\Delta B_k \leq 0$. Simply put at the maximum buffer level ABR algorithm can only reduce or stay with the current quality level.

3.3 Behaviour of the model

3.3.1 Convergence

Figure 2 presents various plots of Eq. (9) ($q_{\min} = 100$ kbps, 2000 kbps, 12000 kbps and $q_{\max} = 8000$ kbps). The most important observable characteristic of the curves is that the rate at which the video quality changes varies. It starts slowly and then becomes faster (the plots having steeper gradients) as the value of video quality level q_k increases. Again, on approaching $q_{\max}$ the rate begins to flatten. From this, it can be derived that $\lim_{B\to\infty} R(t) = q_{\max}$ (i.e. the limiting factor of $R(B)$ is $q_{\max}$). In other words, the maximum value of q assuming an infinite buffer size is $q_{\max}$. As can be seen $q_{\max}$ is asymptotically reached for all the plots independent of the initial value of the video rate (q_0). Furthermore, after reaching $q_{\max}$ any increase in the buffer size does not result in a rise in q. This buffer level (B^*), barring any other consideration by an algorithm designer, can be considered as $B_{\max}$.

Another relevant factor is the convergence time of the model. First, from the discussion of the desirable trajectory at Sect. 3.1 we can deduce that the evolution constant α determines the speed of the $g'(R)$. So, the higher the value of α the faster the rate of video quality change and hence the shorter the convergence time. Therefore, the choice of this parameter has an impact on the convergent time.

3.3.2 Stability

The equilibrium of the model is the point at which $C(t) = R(t)$, that is, when $\frac{\mathrm{d}R(t)}{\mathrm{d}B} = 0$. Equating the Eq. (4) to zero gives us two equilibrium points, $q^* = 0$ and $q^* = q_{\max}$. It is obvious that when a client has not started requesting any video it will stay in that state forever. However, it is interesting to investigate the behaviour of the model near $q^* = 0$. When streaming session is just starting, the buffer level is most likely going to be very low, in other words, close to $q^* = 0$ the buffer level is low. When q_k is very small, αq^2 is small compared to $\alpha q q_{\max}$. Therefore, Eq. (4) becomes $\frac{\mathrm{d}R(t)}{\mathrm{d}B} \approx \alpha q q_{\max}$. We can infer from this equation, provided $\alpha < 0$ any small perturbation in the system state will result in an exponential growth of the video rate away from the current rate resulting in an equilibrium that is unstable.

The second equilibrium point is $q^* = q_{\max}$. Again we are interested in what happens near this point. Let us assume that

$$\epsilon = q - q_{\max}.$$

When we substitute $q = q_{\max} + \epsilon$ into Eq. 4, we get

$$\frac{\mathrm{d}R(t)}{\mathrm{d}B} = -\alpha\epsilon q_{\max} - \epsilon^2.$$

However, if q is close to $q_{\max}$, for all $\alpha > 0$ the ϵ^2 will be very small; therefore, we have $\frac{\mathrm{d}R(t)}{\mathrm{d}B} \approx -\alpha\epsilon q_{\max}$. Thus small perturbation will decay exponentially, reverting

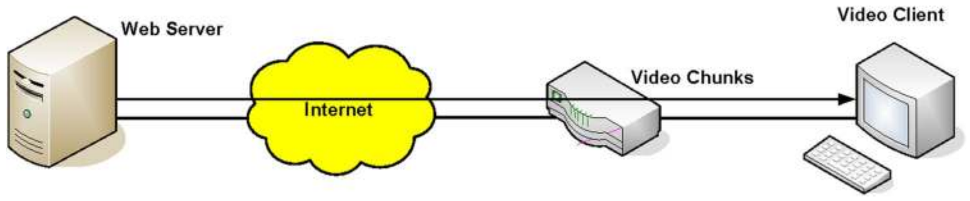


Fig. 3 Experimental set-up

back to the $q_{\max}$. Hence, the equilibrium $q^* = q_{\max}$ is asymptotically stable.

3.4 Implementation

The proposed model is applied within the two selected rate adaptation algorithms to demonstrate its applicability. First, the algorithm proposed by Huang et al. [7, 8], henceforth called the original buffer-based algorithm. And secondly, the one proposed by Miller et al. [16], which we called the original throughput-based algorithm.

When modifying the implementation of the buffer-based algorithm, the same algorithm is used as presented [7, 8] with one modification, that is, from the very beginning the proposed rate map is used. Simply put, the reservoir is not used. Hence, from the start, the algorithm relies on the proposed model. The summary of the algorithm is thus: the current video rate is increased to the next level only if the rate suggested by the proposed model exceeds the next higher available quality level. However, if the current video rate suggested by the model is below the next lower available video rate the quality level is switched down. Otherwise, the algorithm retains its video rate (for a detail discussion of the algorithm see [7, 8]).

To retrofit the proposed model into [16], the algorithm had to be slightly modified. It is worth noting that none of the changes affect the throughput related logic. In order to closely map the original buffer dynamics, the playback buffer is divided into three phases. The first phase is when the video rate change is slow, with a threshold at B_{qt_1}. The next phase is when the video rate grows exponentially, which ends at B_{qt_2}. The third is when the video quality level increase reaches saturation, which starts at $B_{\max}$. The threshold can be calculated thus:

$$B_{qt_x} = R^{-1}(q_{\min} + \beta(q_{\max} - q_{\min}))$$

For $x = 1$ the $\beta = 0.1$ and for $x = 2$ the $\beta = 0.73$. The modified version of the throughput-based algorithm is presented in the Algorithm 1.

Algorithm1: Modifiedthroughput-basedABR

```
input  : $c(t_k)_{k=1,\dots t_e}$
         $B_k\ \{B_k : 0 < B_k \le B_{max}\}$
output: $q_{k+1}$:Nextvideorate
         $B_{delay}$
static runningFastStart:=true;
$B_{delay=0}$
$q_{k+1} = q_k$
if $runningFastStart$
  $\wedge\ q_k \ne q_{max}$
  $\wedge\ q_k \le \alpha_1.c(t_k)$ then
    if $B(t) < B_{qt_1}$ then
        if $q_{k+1} \le \alpha_2.c(t_k) \vee R(t) \ge q_{k+1}$ then
            $q_k := q_{k+1}$
        end
    elseif $B(t) < B_{qt_2}$ then
        if $q_{k+1} \le \alpha_3.c(t_k) \vee R(t) \ge q_{k+1}$ then
            $q_k := q_{k+1}$
        end
    else
        if $q_{k+1} \le \alpha_4.c(t_k) \vee R(t) \ge q_{k+1}$ then
            $q_k := q_{k+1}$
        end
        if $B(t) \ge B_{max}$ then
            $q_k := q_{max} \wedge delay := V$
        end
    end
else
    runningFastStart:=false;
    if $B(t) < B_{qt_1}$ then
        if $R(t) \le q_{k-1}$ then
            $q_k := q_{k-1}$
        end
    elseif $B(t) < B_{qt_2}$ then
        if $q_{k-1} \ge \hat{c}(t) \vee R(t) \le q_{k-1}$ then
            $q_k := q_{k-1}$
        end
    else
        if $q_k \ge \alpha_5.c(t_k) \vee R(t) == q_{max}$ then
            $delay := V$
        else
            $q_k := q_k$
        end
    end
end
```

4 Performance evaluation

This section presents the experimental set-up and the performance evaluation metrics

4.1 Experimental set-up

The test-bed set-up is shown in Fig. 3. The client is connected to the Internet either via an Ethernet switch or EE's 3G networks. The web server is located at the Alpen-Adria-Universität Klagenfurt, which hosts the Big Buck Bunny dataset [12].

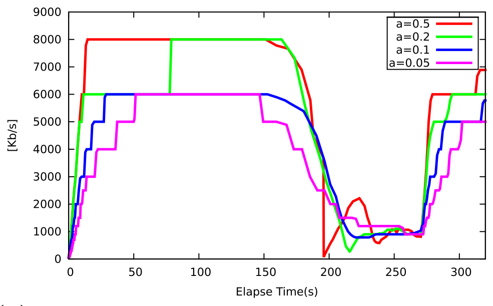
All the players used are implemented in Python, and run on top of Ubuntu 12.04.2 LTS. The host that runs the players also hosts: Dummynet, tcpdump, lsof, and Wget.

Throughout the wireline experimentation, the maximum downstream available bandwidth was limited to 6 mbps, while for the wireless, a "blue-sky" test was conducted. For all the buffer-based players, $B_{\max} = 240$ s. And for the player running the original buffer-based algorithm, the reservoir is set to 40s. We found through experiments the values between 0.05 and 0.1 are appropriate for the growth constant $q_{\max}\alpha$ of the proposed model; therefore, we use $q_{\max}\alpha = 0.05$ throughout, while for both throughput-based algorithms (original and modified), we retain the same configurations as used in [16]. Each experiment was conducted 10 times and the average result is used. When more than one player is used, or when a player and background traffic worked at the same time, all are run on the same machine.

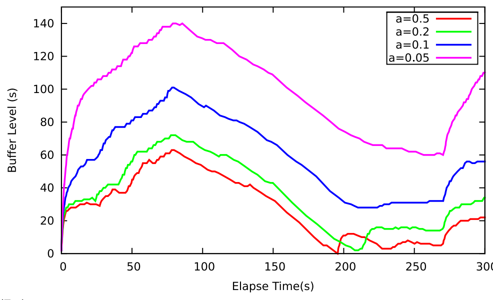
4.2 Evaluation metrics

The following objective QoE metrics are known to affect user experience [11, 15, 19] and are used in evaluating the impact of the proposed model on the two modified algorithms.

- Rebuffers: is the total number of video freeze events per streaming session [19].
- Average video rate: is the average of video quality weighted by the duration each video was played. This is calculated as $\frac{t_1 q_1 + t_2 q_2 \ldots t_n q_n}{t_n - t_1}$ and measured in kb/s [19].
- Instability: is the fraction of successive chunk requests by a player in which the requested video rate changes [2], measured at the steady state.
- Utilisation of available network resource: is calculated by dividing the average video rate by the average network capacity [21].
- Convergence time: is the time taken to settle at the sustainable video rate.
- Start-up Delay: is defined as the amount of time it takes a player to download a predefined number of chunks before the playback starts [11].



(a) Effect of different values of α on video rates.



(b) Effect of different values of α on buffer evolution.

Fig. 4 Impact of different evolution rate constant α

5 Results

This section discusses the result of the various test-bed experiments conducted in both wired and wireless environments. The purpose of these experiments is to evaluate how much improvement in QoE-related metric, if any, is gained by the use of the proposed model.

5.1 Client with wireline access

5.1.1 Evolution rate

The evolution rate of the video quality is determined by the constant α, which takes a value from $\{0 < \alpha\}$. To assess the effect of different values of α on the performance of an adaptive video player we present, in Fig. 4, the result of some experiments conducted using the following carefully selected values: (0.5, 0.2, 0.1, 0.05). As shown in Fig. 4a for the values of $q_{\max}\alpha = 0.5$ and $q_{\max}\alpha = 0.2$ the system is very aggressive. The player downloads video rates that are more than what the available system capacity can safely handle. Additionally, when the bandwidth drops the player is not able to reduce the video rate to the sustainable quality level in time to avoid rebuffering (see Fig. 4b). The video stall is unnecessary since the capacity of the system could

Table 1 Effect of different values of evolution constant(α)

Evolution constant	Buffer undershoot	Convergence time	$B_{\max}$(s) (%)
0.5	3	10	28
0.2	1	12	32
0.1	0	28	63
0.05	0	51	126

have sustained a lower quality level without any rebuffering. Furthermore, when the available capacity is low the player recurrently undershoots its buffer, which not only increases the chances of rebufferings but the player's instability. However, when the evolution rate constant is reduced to $\alpha = 0.1$ and $\alpha = 0.05$ the player is not only able to prevent video freeze but also the video rate is able to converge at maximum available bandwidth.

Table 1 summarises the effect of different values of evolution constant on the performance of the proposed model. It can be seen as the values increase both the buffer requirement and convergence time drop. However, this comes at an increasing risk of rebuffering events. In summary, it was found when $q_{\max}\alpha$ is above 0.1 the player is extremely aggressive, while the player is very stable between 0.1 to 0.05.

5.1.2 Variable bandwidth

These experiments are aimed at demonstrating the elasticity of the proposed model, i.e. how it adapts to a rapidly changing bandwidth. As can be seen from Fig. 5, the streaming started with a maximum available bandwidth of 6 mb/s, after 80s the bandwidth dropped to 2 mb/s at 150s it dropped again to 900 kb/s, finally at 270 s it rose back to 6 mb/s and stayed until the end.

The first thing to note is that the video rate of chunks downloaded by the player employing the proposed model (Fig. 5b, d), are able to converge at a higher video rate, in fact, the modified buffer-based player converges at exactly the system capacity (see Fig. 5b). Table 2 shows that the modified players achieve a maximum video rate of 6 and 5mb/s against the 4mb/s for the original players. This translates to 100 and 85% throughput utilisation, which is an improvement of 33 and 18% utilisation compared to the original buffer-based and throughput-based players, respectively. As can be observed, both throughput-based players suffer lower utilisation than the corresponding buffer-based players, a look at Fig. 5c, d will show that in both instances once the buffer level reaches the threshold value, both players activate the periodic download scheduling to stabilise the buffer. This can be seen in the green plot, results in high throughput variation, consequently affecting the utilisation of both. Furthermore, the modified buffer-based player recorded an improvement of 854 kb/s in the average video rate, while modified throughput-base player improved the average video rate by 433 kb/s.

5.1.3 Video rate convergence

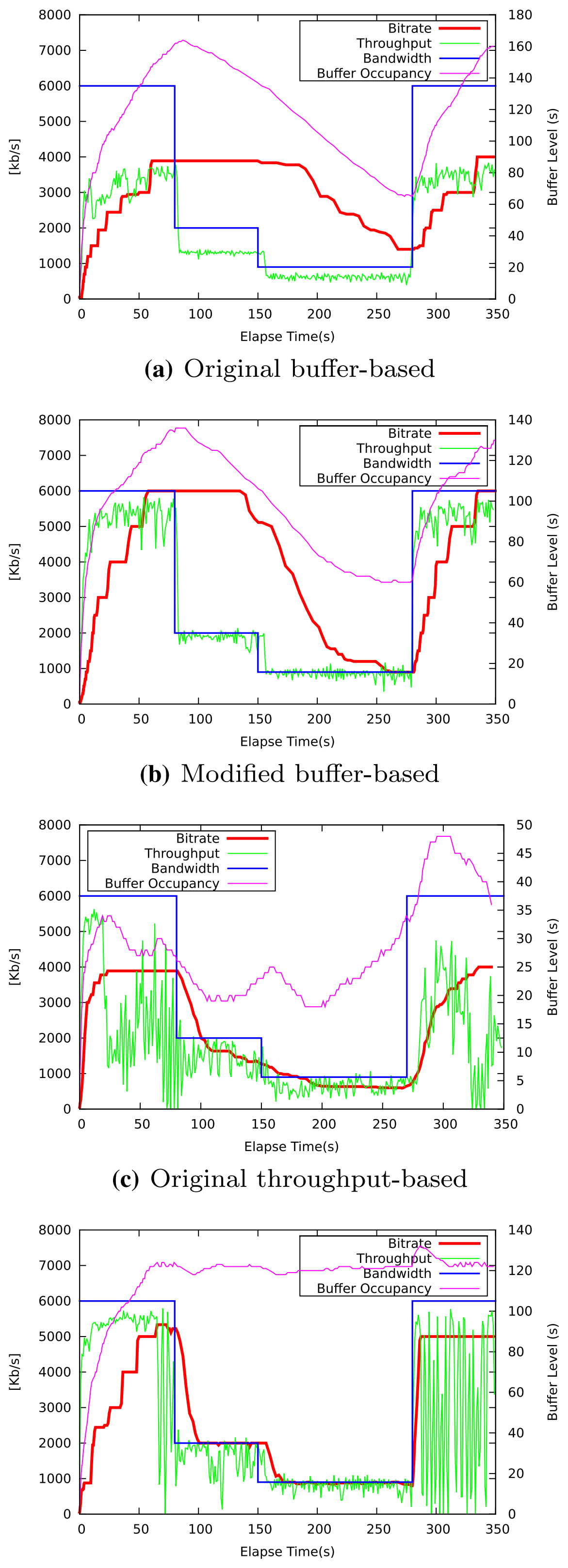
Next, we compare the convergence time of the modified players against the original players. Fig. 6a shows when the bandwidth suddenly increases at 120 s both players running the buffer-based algorithm are able to converge at the right quality level, albeit at different time. While it only took the modified buffer-based player 65 s to reach the convergence state, it took the original buffer-based player three times longer (i.e. 165 s). Furthermore, from Fig. 6b it can be seen that by using the proposed model the convergence time can be reduced by up to 80 s in comparison with the original throughput-based player.

However, a player needs not to always converge to a high video quality level. It can as well converge to a lower quality level. Fig. 7 presents such a scenario. When the bandwidth suddenly drops, it takes the player using the original buffer-based logic longer to converge even though it is coming from a quality level that is a lot lower (see Fig. 7a). That is 102s against 146s for the unmodified buffer-based algorithm.

Furthermore, Fig. 7b shows when the bandwidth suddenly drops, the player running the unmodified throughput-based algorithm was so aggressive in its reduction of the video rate that the player had to reach the lowest available video quality before it later stabilises at a sub-optimal rate. Such a large amplitude in video quality change is detrimental to QoE. However, the player running the proposed model was much more conservative in its reduction and was able to converge at the appropriate quality level.

5.2 Start-up delay

In all the experimentation conducted, the players are set to start playing after fifteen (15) chunks are downloaded, which translates to $B_t = 30$. Figure 8 shows the delay incurred by each of the players. As can be seen, both buffer-based players (modified and original) are able to start playing at less than 2 s, that is, about 1.7 s after the first chunk is requested. Though they achieve similar performance, the original buffer-based player set about 40 s of a reservoir (the initial period when only the lowest video quality is requested), effectively downloading further twenty-three (23) chunks at the lowest video rate after the elapse of the start-up period, when this needs not to be the case. However, the modified buffer-based player starts a gradual increase



(a) Original buffer-based

(b) Modified buffer-based

(c) Original throughput-based

(d) Modified throughput-based

Fig. 5 Video quality change, for both the original and the modified algorithms, operated in an environment with changing bandwidth

in its video quality after the first nineteen (19) chunks are downloaded, i.e. it requests only four further chunks at the lowest video rate after the start of the playback. For the original throughput-based player the start-up delay is quite high, 5.4 s. But interestingly, the modified version reduces the start-up delay to about 2 s. While the former tries to strictly match video rate to the available bandwidth, hence downloading relatively high video rates at the start-up period the latter is more conservative, replenishing its buffer faster.

5.3 Fairness

Another desirable property of a streaming service is fairness towards other players and background traffic in the network. In this section, we investigate how fair the players running the proposed model are to both other players and background long-live TCP traffic.

5.3.1 Multiple players

For the experiments, four players running the same implementation of either the original players or the modified player are used, as the case may be. In each case, the maximum bandwidth of the bottleneck link is set to 6 mb/s. In the case where the players are fair, they should equally share the available bandwidth since all the players are connected to the same network and are also running on a similar device. As can be seen from Fig. 9a, b none of the buffer-based players (both the modified and original) utilised more/less than 1.5mb/s, which is the fair share. Furthermore, they achieved this with a high level of stability. However, the case of the two throughput-based players there is a slight difference, both players were able to reach the fair bandwidth, but its comes at the cost of decrease in stability (see Fig. 9c, d).

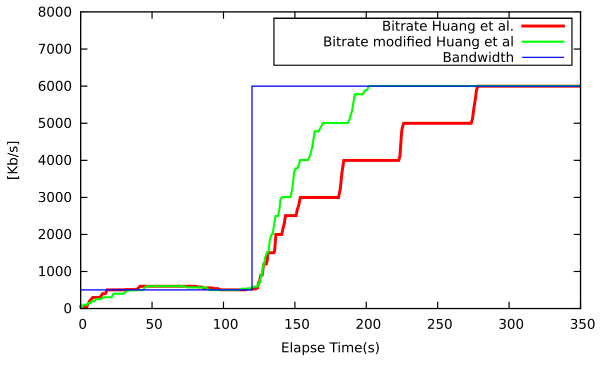
5.3.2 A player and background traffic

In this section, the impact of background TCP traffic on all the players is investigated. To do this the case where a player and background traffic (file download from the same server) compete is investigated. The download starts 30s after the start of streaming.

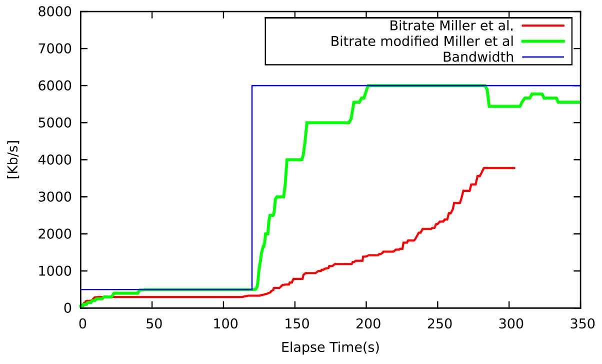
As can be seen from Fig. 10 in all the investigated scenarios the players are able to use, almost, the entire available bandwidth (see the achieved throughput in green). However, as soon as the background traffic is started the achieved throughput starts to gradually drop until an equilibrium is reached. The buffer-based players (both the baseline and modified versions) fairly share the available bandwidth, that is, each uses about 3000 kbps. Furthermore, it is worth noting that the drop in video rate does not affect the stability of the players. In other words, both buffer-based

Table 2 Adaptation for variable bandwidth

Players	Maximum video rate (kb/s)	Average video rate (kb/s)	Throughput utilisation (%)
Original buffer-based	4000	2982	67
Modified buffer-based	6000	3827	100
Original throughput-based	4000	2212	67
Modified throughput-based	5000	2645	85

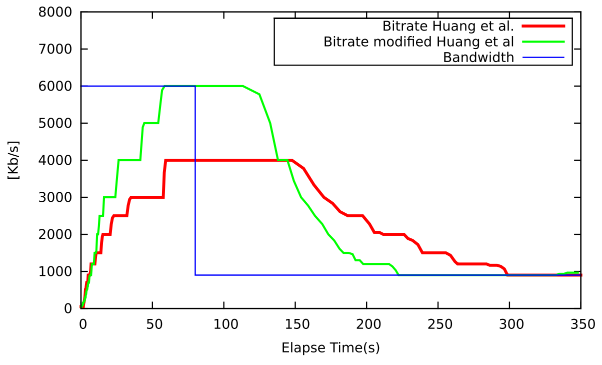


(a) Original vs modified buffer-based player

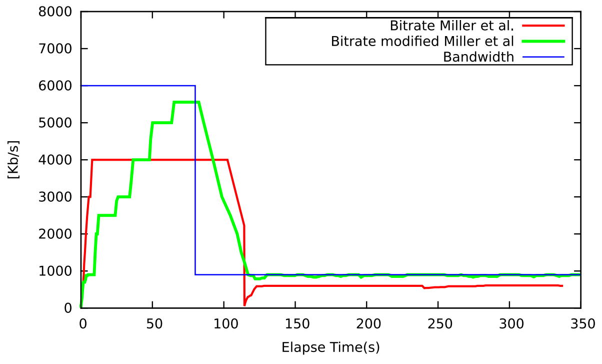


(b) Original vs modified throughput-based player

Fig. 6 Video quality convergence, for both the original and the modified algorithms, when bandwidth increases



(a) Original vs modified buffer-based player



(b) Original vs modified throughput-based player

Fig. 7 Video quality convergence, for both the original and the modified algorithms, when bandwidth suddenly drops

players are able to maintain a near absence of video rate oscillation in their download even though the TCP throughput fluctuates (see Fig. 10a, b). However, the original throughput-based player requests video chunks lower than its fair share (1500 kbps) under-utilising its fair share of the available bandwidth by about 50%. The modified version improves the video rate to 2000 kbps, but still not a fair situation.

5.4 Client with wireless access

The next set of experiments were conducted in a wireless environment. We used a MacBook Pro to run all the

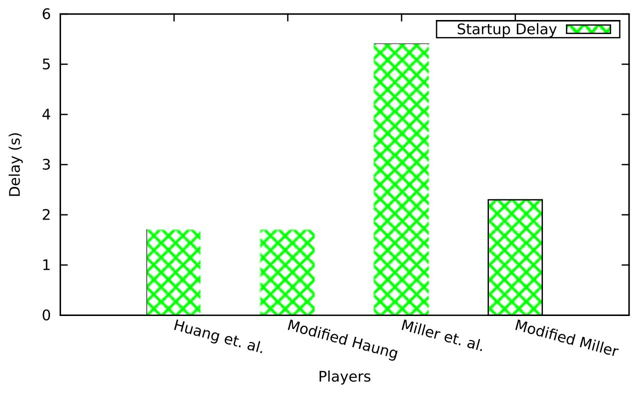
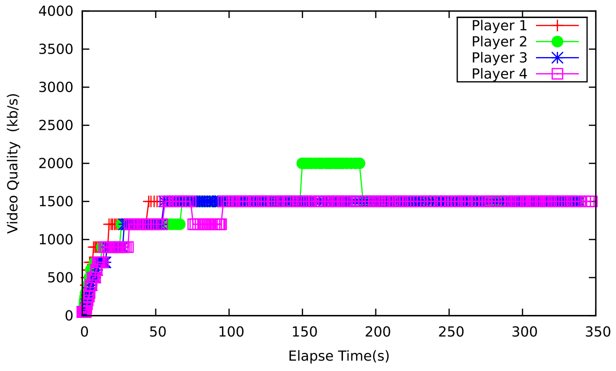
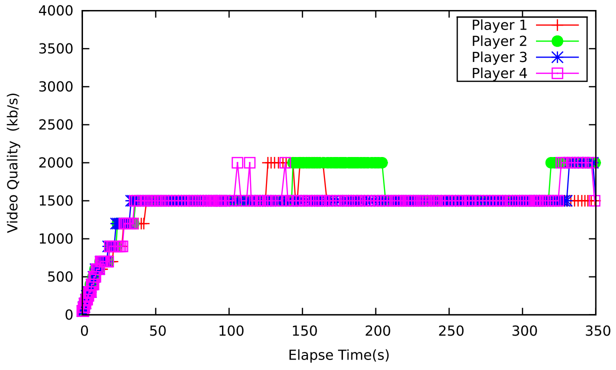


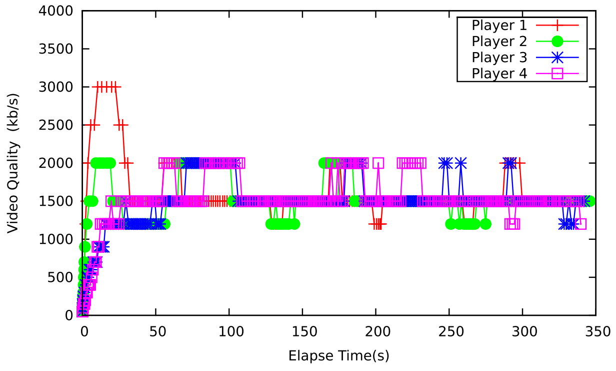
Fig. 8 Start-up delay when $B_t = 30$ s



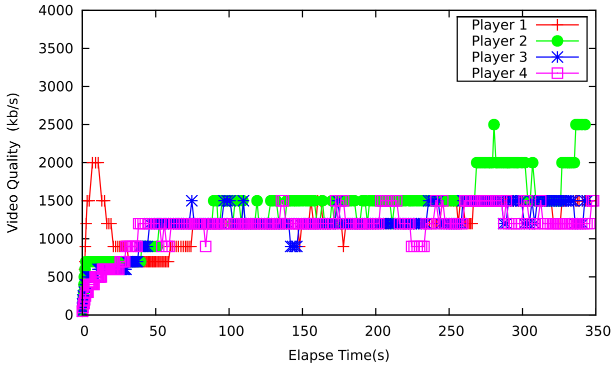
(a) Original buffer-based player



(b) Modified buffer-based player

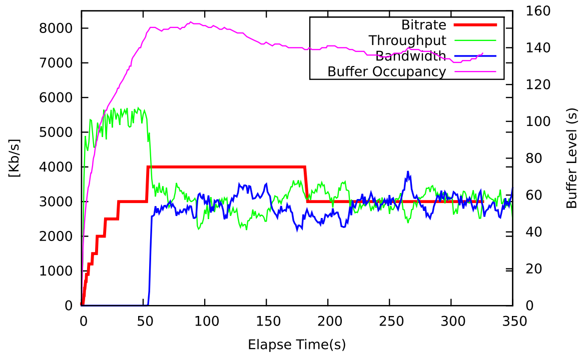


(c) Original throughput-based player

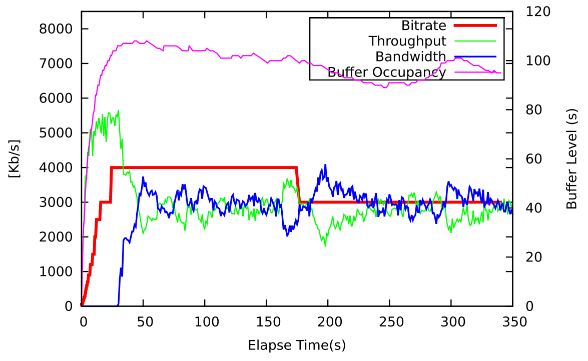


(d) Modified throughput-based player

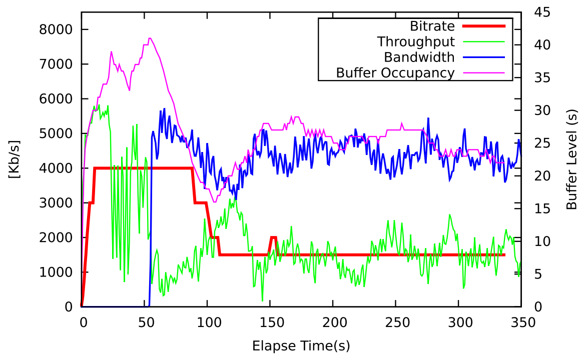
Fig. 9 Four players streaming at the same time



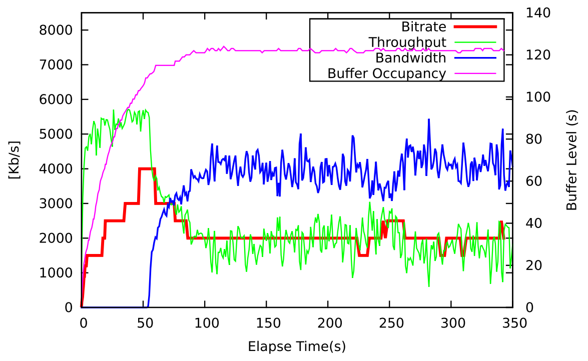
(a) Original buffer-based player



(b) Modified buffer-based player



(c) Original throughput-based player



(d) Modified throughput-based player

Fig. 10 Video quality change as a player compete with background TCP traffic

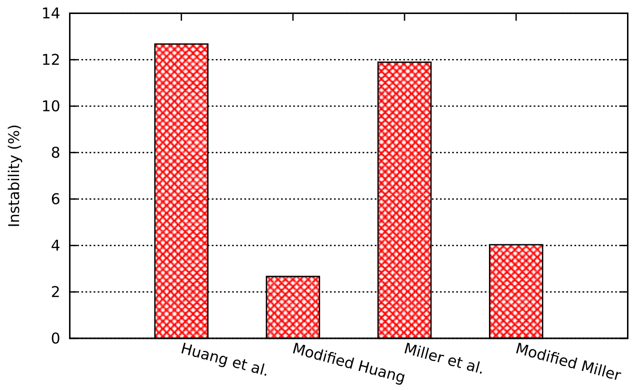


Fig. 11 Video quality stability

Table 3 Adaptation in wireless environment

Players	Maximum video rate (kb/s)	Average video rate (kb/s)
Original buffer-based	600	567
Modified buffer-based	1500	1247
Original throughput-based	700	536
Modified throughput-based	1500	1239

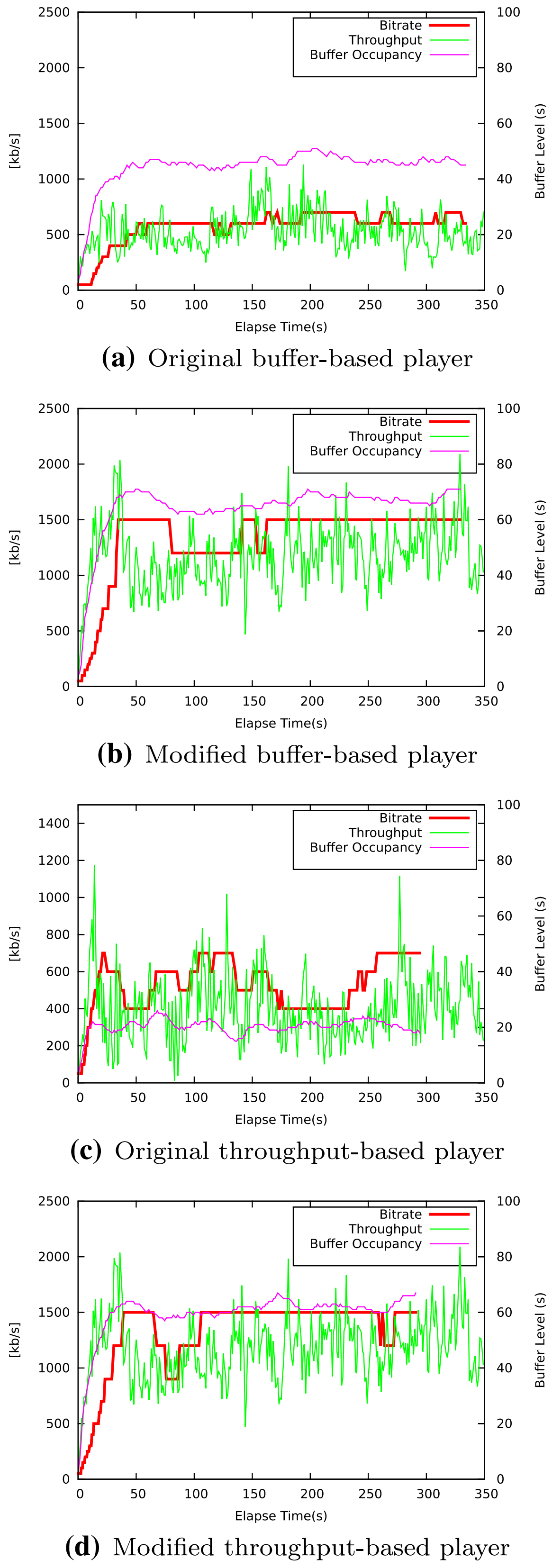
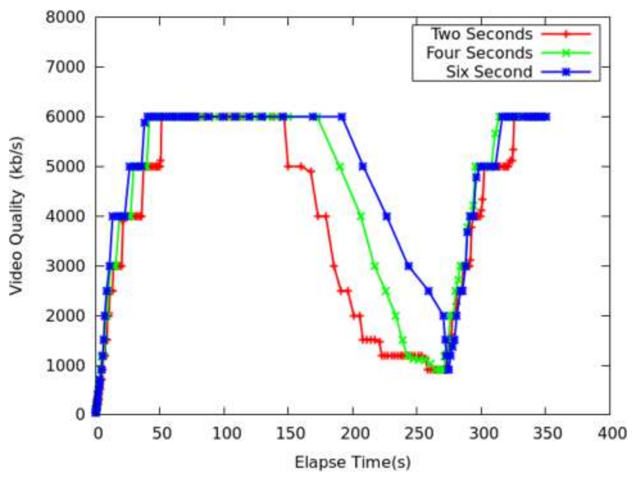


Fig. 12 Video quality change, for both the original and the modified algorithms, operated in a wireless environment

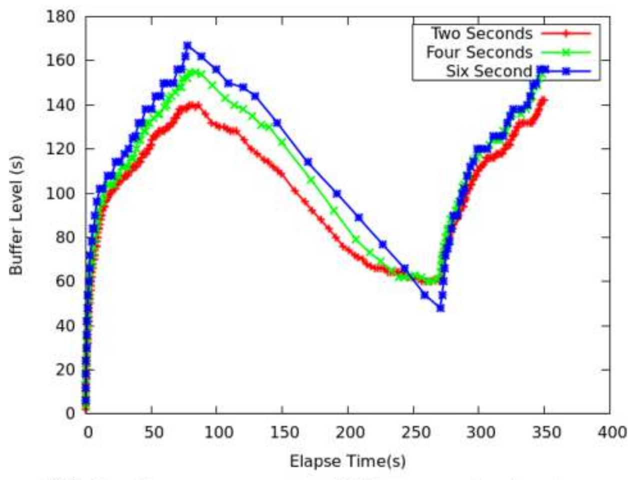
players as and when required. The laptop was connected to the EE 3G network at Lancaster City Centre. The same server as in the case of wired environment is used. All experiments were conducted within two days. The channel capacity, in all of the presented results, has not been restricted.

As can be seen from Fig. 12 the throughput of the wireless network is highly fluctuating, which makes it difficult to ascertain the actual capacity of the link. Therefore, no utilisation is presented. From Fig. 12a, c the maximum quality level attained by the original players are 600 and 700 kb/s respectively. However, the modified versions of the players are able to achieve 1500 kb/s each (see Fig. 12b, d). Importantly, this helps the modified players to achieve 45% increase of the average video quality. The summary results are presented in Table 3.

Finally, a stability test for the players is conducted (see Fig. 11). Both the original throughput-based and buffer-based players suffer a high degree of instability, at the steady-state the players are respectively 12.6 and 11.8% unstable. However, the instability is significantly reduced, when the proposed model is used, to 2.6% for the modified buffer-based player and 4.0% for the modified throughput player.



(a) Video quality change for different chunk sizes



(b) Buffer occupancy of different chunk sizes

Fig. 13 Impact of different chunk sizes

Table 4 Effect of different chunk sizes

Chunk size (s)	Average video rate (kb/s)
2	4000
4	4524
6	4891

5.5 Effect of different chunk sizes

Up until now, we have used chunks with the size of 2s, that is $V = 2$, in this section, we present the test of the sensitivity of the proposed model to different chunk sizes. For this, we run the same experiment presented at Sect. 5.1.2 with chunks of four and six seconds video playtime each. Further, only the result of the modified buffer-based player is shown.

From Fig. 13b, it can be seen the pattern of the buffer evolution is identical regardless of the chunk size used. From this, we can conclude that the model is not sensitive to the change in chunk size. However, the proposed model has seen an increase in average video rate as the chunk size increases (see Table 4). This is confirmation of a result reported in earlier in the literature, which found the larger chunk size the higher the chances of TCP reaching the steady state, which in turn allows the play to perceive higher throughput.

6 Conclusion

The purpose of an adaptive bitrate selection module, in HTTP adaptive streaming services, is primarily to match video rate to the available system capacity. The selection of the resource representing the system capacity, on which the video rate adaptation is based on, is context dependent. However, regardless of the resource use as a situational indicator, It will be difficult to build an ABR scheme that maximises QoE without taking into consideration the dynamics of buffer state. In this paper, a QoE-aware model of the relationship between video rate and buffer state changes is presented. The model defines the optimal buffer requirements for any given set of video quality levels while providing an optimum QoE. The proposed scheme is evaluated within a real-world Internet environment. Our results show that incorporating a system model that captures the relationship between buffer state changes and video rate reduces both the start-up delay and convergence time, while at the same time increases the average video rate, network utilisation, and stability. Importantly, this is achieved without introducing additional rebuffering.

References

1. Akhshabi, S., Anantakrishnan, L., Begen, A.C., Dovrolis, C.: What happens when http adaptive streaming players compete for bandwidth? In: Proc. of the NOSSDAV, pp. 9–14 (2012)
2. Akhshabi, S., Anantakrishnan, L., Dovrolis, C., Begen, A.C.: Server-based traffic shaping for stabilizing oscillating adaptive streaming players. In: NOSSDAV, pp. 19–24 (2013)
3. Cranley, N., Perry, P., Murphy, L.: User perception of adapting video quality. Int. J. Hum.-Comput. Stud. **64**(8), 637–647 (2006)
4. De Cicco, L., Mascolo, S.: An adaptive video streaming control system: modeling, validation, and performance evaluation. IEEE/ACM Trans. Netw. **22**(2), 526–539 (2014)

5. Hossfeld, T., Seufert, M., Sieber, C., Zinner, T.: Assessing effect sizes of influence factors towards a QoE model for http adaptive streaming. In: Proc. of the QoMEX, pp. 111–116 (2014)
6. Huang, T.Y., Handigol, N., Heller, B., McKeown, N., Johari, R.: Confused, timid, and unstable: picking a video streaming rate is hard. In: Proc of the ACM IMC, pp. 225–238 (2012)
7. Huang, T.Y., Johari, R., McKeown, N.: Downton abbey without the hiccups: buffer-based rate adaptation for http video streaming. In: Proc. of the FhMN, pp. 9–14 (2013)
8. Huang, T.Y., Johari, R., McKeown, N., Trunnell, M., Watson, M.: A buffer-based approach to rate adaptation: evidence from a large video streaming service. In: Proc. of the 2014 ACM conference on SIGCOMM, pp. 187–198 (2014)
9. J.341, I.T.: Objective perceptual multimedia video quality measurement of HDTV for digital cable television in the presence of a full reference. https://www.itu.int/rec/T-REC-J.341-201603-I/en
10. Jiang, J., Sekar, V., Zhang, H.: 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, ACM, pp. 97–108 (2012)
11. Krishnan, S., Sitaraman, R.: Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. IEEE/ACM Trans. Netw. **21**(6), 2001–2014 (2013)
12. Lederer, S., Muller, C., Timmerer, C.: Dynamic adaptive streaming over http dataset. In: Proc. of the MMsys, pp. 89–94 (2012)
13. Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A., Oran, D.: Probe and adapt: Rate adaptation for http video streaming at scale. IEEE J. Sel. Areas Commun. **32**, 719–733 (2014)
14. Liu, C., Bouazizi, I., Gabbouj, M.: Rate adaptation for adaptive http streaming. In: Proceedings of the second annual ACM conference on Multimedia systems, pp. 169–174 (2011)
15. Liu, Y., Dey, S., Gillies, D., Ulupinar, F., Luby, M.: User experience modeling for dash video. In: 2013 20th International Packet Video Workshop, pp. 1–8 (2013)
16. Miller, K., Quacchio, E., Gennari, G., Wolisz, A.: Adaptation algorithm for adaptive streaming over http. In: Proc. of the PV, pp. 173–178 (2012)
17. Mok, R.K., Luo, X., Chan, E.W., Chang, R.K.: Qdash: a QoE-aware dash system. In: Proc. of the MMsys, pp. 11–22 (2012)
18. Sani, Y., Mauthe, A., Edwards, C., Mu, M.: A bio-inspired HTTP-based adaptive streaming player. In: 2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW), pp. 1–4 (2016)
19. Seufert, M., Egger, S., Slanina, M., Zinner, T., Hobfeld, T., Tran-Gia, P.: A survey on quality of experience of http adaptive streaming. Commun. Surv. Tutor. IEEE **17**(1), 469–492 (2015)
20. Thang, T.C., Ho, Q.D., Kang, J.W., Pham, A.T.: Adaptive streaming of audiovisual content using mpeg dash. IEEE Trans. Consum. Electron. **58**(1), 78–85 (2012)
21. Tian, G., Liu, Y.: Towards agile and smooth video adaptation in dynamic http streaming. In: Proc. of the CoNEXT, pp. 109–120 (2012)
22. Wang, B., Kurose, J., Shenoy, P., Towsley, D.: Multimedia streaming via TCP: an analytic performance study. In: Proceedings of the 12th annual ACM international conference on Multimedia, pp. 908–915 (2004)
23. Xu, Y., Zhou, Y., Chiu, D.M.: Analytical QoE models for bit-rate switching in dynamic adaptive streaming systems. IEEE Trans. Mob. Comput. **13**(12), 2734–2748 (2014). doi:10.1109/TMC.2014.2307323
24. Yamagishi, K., Hayashi, T.: Parametric packet-layer model for monitoring video quality of iptv services. In: 2008 IEEE International Conference on Communications, pp. 110–114 (2008)
25. Zink, M., Schmitt, J., Steinmetz, R.: Retransmission scheduling in layered video caches. In: IEEE International Conference on Communications, 2002. ICC 2002, vol. 4, pp. 2474–2478 (2002)