

Price-based Controller for Quality-Fair HTTP Adaptive Streaming (Extended Version)

Stefano D’Aronco¹, Laura Toni², and Pascal Frossard¹

¹LTS4, Ecole Polytechnique Fédérale de Lausanne (EPFL)

²Electrical and Electronic Department, University College London (UCL)

Abstract

HTTP adaptive streaming (HAS) has become the universal technology for video streaming over the Internet. Many HAS system designs aim at sharing the network bandwidth in a rate-fair manner. However, rate fairness is in general not equivalent to quality fairness as different video sequences might have different characteristics and resource requirements. In this work, we focus on this limitation and propose a novel controller for HAS clients that is able to reach quality fairness while preserving the main characteristics of HAS systems and with a limited support from the network devices. In particular, we adopt a price-based mechanism in order to build a controller that maximizes the aggregate video quality for a set of HAS clients that share a common bottleneck. When network resources are scarce, the clients with simple video sequences reduce the requested bitrate in favor of users that subscribe to more complex video sequences, leading to a more efficient network usage. The proposed controller has been implemented in a network simulator, and the simulation results demonstrate its ability to share the available bandwidth among the HAS users in a quality-fair manner.

1 Introduction

HTTP adaptive streaming (HAS) has become the universal client-driven streaming solution for video distribution over the Internet, an example of this paradigm is given by the Dynamic Adaptive Streaming over HTTP [1] (DASH) standard. In HAS, as it is shown in Fig. 1, the video content is available at the main server in different coded versions, namely representations, each one with a given bitrate and resolution. The representations are subdivided into chunks of few seconds typically, which are then downloaded by clients using HTTP requests over TCP. Each HAS client selects the best representation to download (i.e., the best encoding rate and resolution) independently from the other clients. Therefore HAS systems are able to respond to the heterogeneous demands of several HAS clients in a fully distributed and adaptive way. The bitrate to download is usually selected by taking into account both the download rate of the previous chunks and the status of the playout buffer, with the aim of maximizing the downloaded bitrate while minimizing the possibility of rebuffering events.

One of the most challenging aspects in HAS systems is the proper design of the adaptation logic (i.e., the selection of the bitrate to request) at the client side. An intense research

has focused on designing HAS client controllers that guarantee a stable and fair utilization of the network resources among multiple clients sharing the same bottleneck. However, most of this research aims at reaching rate fairness among clients rather than quality fairness. Ideally, video distribution solutions should share the bandwidth in such a way that the different users experience a similar video quality. Unfortunately, since video sequences generally have different characteristics, equal rate allocation among clients (rate fairness) does not necessarily translate into quality fairness. From this point of view, the complete freedom left to HAS clients that selfishly maximizes their own download bitrate reveals its drawback. To overcome this main limitation, the MPEG group is developing an extension of the DASH standard called Server and Network Assisted DASH (SAND) [2]. SAND is based on asynchronous client-to-network and network-to-network transmissions aimed at improving the Quality of Service (QoS) without interfering with the delivery of the media stream. In this spirit, we focus on the bitrate selection problem in order to increase the overall QoS of the clients and therefore improve the quality fairness.

Inspired by the well known Network Utility Maximization (NUM) framework in congestion controllers [3], we design a price-based distributed controller, that maximizes the overall delivered QoS and improves the QoS fairness among users while respecting the guidelines of the SAND extension. More in details, we consider a multi-users HAS system where clients share a common bottleneck. We define an objective function to properly map the encoding rate of the downloaded representations to the QoS delivered to clients. Typically, different video characteristics lead to different objective functions. We then define the congestion level of the network as a function of the downloading times of the chunks, which value can easily be measured by the clients. We introduce a coordination node, which corresponds, for example, to a DASH-Assisting Network Element (DANE) in the SAND terminology. As shown in Fig. 2, this node does not have to lie on the media delivery path, which facilitates the deployability of the proposed solution. The coordination node gathers the downloading times of the chunks from the HAS clients and iteratively updates the price value accordingly, the updated price is then sent back to the clients. By following an appropriate price-based bitrate selection policy, users with simple video sequences, i.e., low bandwidth requirements, do not increase the bitrate of the requested chunks in congested periods in favor of users downloading more complex videos. This policy ultimately leads to a higher overall QoS of the HAS system and to a quality-fair resource allocation. We test the proposed solution in a network simulator (NS3) under different network conditions and we compare it with other rate-fair controllers proposed in the literature. The simulation results confirm that the achieved rate allocation leads to a better quality fairness among the users with respect to the baseline rate-fair HAS controllers. Moreover, we show the ability of our new algorithm to coexist with TCP cross-traffic and other HAS controllers.

In summary, the main contributions of the paper are the following: *i*) we propose a distributed HAS controller that targets quality fairness among several HAS clients sharing a common bottleneck; *ii*) we introduce a method for measuring the congestion level of a bottleneck link for HAS that relies exclusively on client measurements; *iii*) we design our controller such that it can be integrated in the SAND architecture; *iv*) we carry out performance simulations with a realistic network simulator that shows the benefits of the proposed solution.

The paper is structured as follows. In Section 2, we report some related works on the

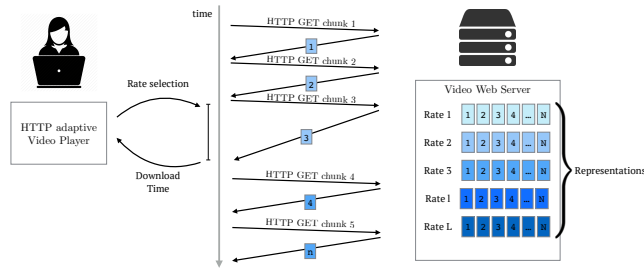


Figure 1: General HAS system architecture.

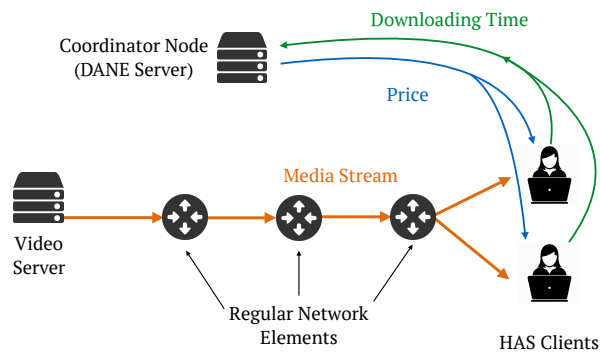


Figure 2: Proposed system architecture.

QoS enhancement in HAS systems. In Section 3, we provide a description of the considered framework. In Section 4, we derive the theoretical foundation of the proposed bitrate selection strategy using a simplified model. In Section 5, we describe in detail the practical implementation of the controller. We present in Section 6 the simulations results. Finally, conclusions are provided in Section 7.

2 Related Works

Since a complete description of the whole literature in adaptation algorithms for HAS would not possible due to space limitations, with the following we discuss the works that focus on quality-fairness in HAS.

In [4], the authors optimize the bitrate selection in order to maximize the Quality of Experience (QoE) among a set of HAS users on a wireless link. In this case the base station carries out the optimization according to the different video characteristics. Though this system is able to effectively allocate the available bandwidth it has some drawbacks in terms of deployability, it requires to modify a network element that lies on the delivery path, and scalability, the base station has to collect all the information about the users' videos and solve the optimization problem. In our proposed system the coordinator is not responsible

for solving the optimization problem and it does not need to hold any per user information, thus preserving system scalability.

Several works [5–7] have proposed solutions for improving DASH QoS based on Software Defined Networking (SDN). The common feature of these solution is the presence of a central network controller that controls the video flows that are currently active in the network. While SDN is a promising technology to improve Internet performance, it is not currently deployed on a wide scale, therefore solutions based on this technology are not suited for many of the nowadays networks. In this work, we rather aim at improving the QoS in HAS with an algorithm that exclusively works at the application level and does not assume any particular technology about the inner network nodes.

In [8], the authors propose a Q-learning multi-agent system for HAS users sharing a common bottleneck in order to maximize a global QoS metric. The problem is formulated as a reinforcement learning problem where the HAS user represents the learning agents. Although this method ultimately achieves the optimal bitrate selection, it requires a very long training phase to learn the optimal solution, making the deployability of this system in realistic environments problematic. In our case we use a model-based formulation therefore we do not require any learning phase and we quickly converge to the optimal bitrate selection.

3 System Model

We describe in detail the framework studied in this paper. We consider a HAS system with N users, or clients, sharing a bottleneck link with an unknown available capacity C . This scenario though not general is quite common, think for example about the case where the N users share the same access link or the case where the server access link is the bottleneck. In the event that a heavy traffic load is detected on these links the group of users can ostensibly be gathered.

Each client downloads video chunks of time duration T_{ck} by sending HTTP requests to the server. The client then stores the received video data in the playout buffer, which has a maximum capacity of M chunks. After a chunk is downloaded the next one is requested immediately if a free slot is available in the buffer, otherwise the client waits until a chunk is played and a slot becomes free to request the next one. When the buffer is full, requests therefore are made every T_{ck} in stationary regime.

Let r_i be the bitrate of the last chunk downloaded by user i , and $\mathbf{r} = [r_1, r_2, \dots, r_N]$ be the vector corresponding to the bitrates of all the recent clients requests. We denote by $\tau_i(\mathbf{r})$ the downloading time for client i , defined as the time necessary for user i to download a chunk encoded at rate r_i . Note that $\tau_i(\mathbf{r})$ depend s on the entire vector \mathbf{r} , since the bottleneck is shared by all users. We denote the rate vector \mathbf{r} as *sustainable* if $\tau_i(\mathbf{r}) \leq T_{ck}, \forall i$. A sustainable rate vector implies that users download their chunks in an amount of time that is sufficient to avoid buffer underflow.

Note that the downloading time $\tau_i(\mathbf{r})$ is an extremely complex function in reality, and represents the network response to the client requests. It depends on the capacity C of the bottleneck link, on the starting time of the downloads, as well as on the random fluctuations of the TCP rate due to packet losses. For the sake of simplicity, we first assume an ideal TCP behavior, which means that: *i*) the bandwidth is always equally shared among the active

connections, *ii*) the channel is fully utilized when at least one connection is active. Note that these are the ideal characteristic of every rate-fair congestion control algorithms. We then use a realistic TCP connection to evaluate our controller in the conducted experiments.

We define $U_i(r_i)$ to be a strictly increasing concave utility function that represents the quality experienced by user i when the video is downloaded at bitrate r_i . Utility functions of different users have different shapes to model the different bandwidth requirements for different video sequences. We finally define the overall QoS of the system as the sum of the single utility functions experienced by each user, more formally, $\mathcal{U}(\mathbf{r}) = \sum_{i=1}^N U_i(r_i)$.

4 Quality-Fair HAS Congestion Controller

In this section we derive the theoretical foundation of the proposed controller. We focus on the bitrate selection of the users at regime, which means that users need to experience a stationary average downloading time smaller than or equal to T_{ck} , in order to avoid buffer underruns, and they request one video chunk every T_{ck} . Rebuffering phases and proper buffer management policies are considered later in the practical implementation of the controller, which is described in the next section.

We formulate a utility maximization problem for the multi-user system at regime. The goal is to find a rate vector \mathbf{r} that is sustainable and that maximizes the aggregate utility. This can be achieved by solving the classical NUM problem:

$$\begin{aligned} & \underset{\mathbf{r}}{\text{maximize}} && \sum_{i=1}^N U_i(r_i) \\ & \text{subject to} && \sum_{i=1}^N r_i \leq C. \end{aligned} \tag{1}$$

The problem consists in maximizing a concave objective function of utilities subject to a linear inequality constraint on the cumulative bitrate.

The optimization problem in (1) can be solved using a dual algorithm, see [3, 9]. The Lagrangian of the problem in (1) corresponds to:

$$L(\mathbf{r}, \mu) = \sum_{i=1}^N U_i(r_i) + \lambda \left(\sum_{i=1}^N r_i - C \right), \tag{2}$$

where λ is the dual variable, or price, associated to the bottleneck capacity constraint. The optimal solution of the problem can be determined by solving iteratively the following system of discrete dynamic equations:

$$r_i^{k+1} = \left[U'_i(\lambda^k) \right]^{-1} \quad i = 1 \dots N \tag{3a}$$

$$\lambda^{k+1} = \left(\lambda^k + \beta \left(\sum_{i=1}^N r_i^{k+1} - C \right) \right)_+ \tag{3b}$$

where $[U'_i(\cdot)]^{-1}$ represents the inverse of the derivative of the utility function of user i , $(\cdot)_+$ denotes the projection onto the positive orthant and β is a simple parameter to set the

speed of change of the dual variable. Note that users can compute the first step, Eq. (3a), independently, if they know the value of the dual variable λ . For evaluating the second step, Eq. (3b), the value of the capacity C needs to be known. However this quantity cannot be determined handily since its value depends on protocols overheads and potential cross traffic (which cannot be known in advance).

We need therefore to modify the second step of the iterative algorithm in order to avoid the explicit the value of the capacity C . We thus propose to use the maximum downloading time $\tau_{MAX} = \max_{i=1\dots N} \tau_i$ in place of the rate sum. According to the ideal TCP behavior described in the previous section, when $\sum_{i=1}^N r_i \leq C$ the rate vector \mathbf{r} is sustainable since the total amount of data can be downloaded in less than T_{ck} , similarly when $\sum_{i=1}^N r_i > C$ the rate vector is not sustainable¹. We can therefore map the sum rate constraint into a downloading time constraint, leading to the following equivalent conditions:

$$\sum_{i=1}^N r_i \leq C \iff \tau_{MAX}(\mathbf{r}) \leq T_{ck} \quad (4)$$

By using the above equivalency we modify the dynamic system in (3) as follows:

$$r_i^{k+1} = \left[U'_i(\lambda^k) \right]^{-1} \quad i = 1\dots N \quad (5a)$$

$$\lambda^{k+1} = \left(\lambda^k + \beta(\tau_{MAX}(\mathbf{r}^{k+1}) - T_{ck}) \right)_+ \quad (5b)$$

The first step has not changed, but the second step of Eq. (5b) can now be easily computed since every user knows the downloading time of the requested chunks, and the maximum value can easily be extracted. The capacity value is not used explicitly anymore, however it is implicitly included in the downloading time measurement $\tau_{MAX}(\mathbf{r})$. Since the constraints in (4) are equivalent, (3) and (5) converge at equilibrium to the same rate vector \mathbf{r} .

We give now a brief discussion of how the iterative steps of system (5) can be computed in reality. The adaptation logic, i.e., the selection of the bitrate at the client side is represented by Eq. (5a), while the price update of the coordinator node is given by Eq. (5b). In more details, in the first step, Eq. (5a), all the users independently compute the optimal bitrate and request the chunks to download at the next iteration accordingly. After the download every user sends to the coordinator node the measured downloading time. The coordinator then performs a maximum pooling operation on the received downloading times and updates the dual variable λ using Eq. (5b). The value of λ is then sent to the users for the next bitrate selection. By performing these steps iteratively, the system converges to the optimal equilibrium point.

The iterative solution in (5) represents a modification of the solution of classical NUM problems for the case of HAS system. By using the downloading time of the chunks we can detect an overuse of the available bandwidth without requiring the knowledge of its actual value. Finally, note that the equivalency of the two conditions in Eq. 4 is true only if the ideal characteristic of the congestion control is verified. If this assumption does not hold, the equivalency is only an approximation whose accuracy depends on the actual behavior

¹This follows directly from the assumption that when a single TCP connection is active the channel is fully utilized, and that users request at least one chunk every T_{ck} to avoid buffer underruns.

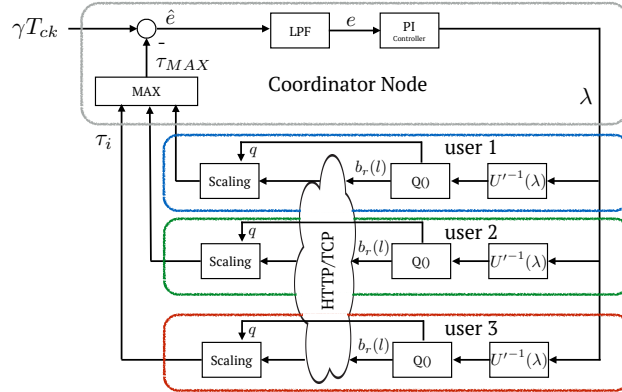


Figure 3: Simplified block diagram of the overall system.

of the congestion control. As a result in the real world we need to consider the usage of the downloading time condition instead of the original rate condition as an heuristic approximation suggested by ideal assumption on the congestion control used. Nevertheless the update rules in (5) are extremely important as to derive a cooperative adaptation strategy for HAS users.

5 Controller Implementation

From the theoretical study of the previous section, we now show how to adjust the iterative solution in (5) for it to be used in HAS system in practice. In particular we consider a discrete rather than continuous set of bitrates, as well as the actual playout buffer management. The overall HAS multi-user system, depicted in Fig. 3, can be seen as a control loop composed of two main entities: the coordinator node, which receives the downloading time measurements from the users and updates the price λ accordingly; and the users, which receive the price from the coordinator node and perform the chunk requests based on the video characteristic and the updated price.

5.1 Coordinator Node

In Algorithm 1, we present the operations that are executed by the coordinator node to update the price λ at every iteration step. The key point is to have a coordinator node that stays as simple as possible without any need for a per user state information, such that the scalability of the system is preserved.

Since users are not synchronized the coordinator node processes one user transmission per time. For each downloading time measurement τ_i , received from client i , the coordinator updates the current maximum downloading time (line 2) and returns the last updated value of the price to user i .

The second part of the algorithm is executed every T_{ck} and corresponds to the price update. In order to compute the error signal, \hat{e} , the coordinator node needs a reference

Algorithm 1 Coordinator Algorithm

```
1: if New downloading time received from user then
2:    $\tau_{MAX} \leftarrow \max(\tau_{MAX}, \tau_i)$ 
3:   Send most recent  $\lambda$  to the user
4:
5: loop ▷ executed every  $T_{ck}$ 
6:    $\hat{e} := \tau_{MAX} - \gamma T_{ck}$ 
7:    $e \leftarrow \alpha_e e + (1 - \alpha_e) \hat{e}$  ▷ LPF
8:    $e_I \leftarrow \max(0, e_I + e)$ 
9:    $\lambda \leftarrow \max(0, K_P e + K_I e_I)$  ▷ Update price
10:   $\tau_{MAX} \leftarrow 0$  ▷ Reset  $\tau_{MAX}$ 
```

signal γT_{ck} , which expresses the value of the maximum downloading time that the users must have at equilibrium, with $\gamma \in [0, 1]$ being a multiplicative factor². The error between the maximum downloading time and the reference downloading time is evaluated (line 6) and filtered by a Low Pass Filter (LPF), implemented as an Exponential Weighted Moving Average (EWMA) with a coefficient equal to α_e , (line 7). The filter is necessary since the maximum downloading time is a noisy measure in realistic settings, due to the random behavior of multiple coexisting TCP flows. Finally, the error is integrated according to Eq. (5b) (line 8). Since the complete control loop is composed also by non-linear blocks, e.g., the utility functions, we limit the value of the integral error to zero in order to avoid integral windup effects [10].

The value of the final price λ is then calculated by combining the integral error and the proportional error (line 9), where K_P and K_I represent the proportional and integral gain respectively. Compared to Eq. (5b) we add in the practical implementation a proportional error to improve the stability of the system without affecting the equilibrium point. The values of K_P and K_I must be set in order to guarantee the stability of the system, i.e., ensuring that the loop return ratio of the control loop has a positive phase margin at the cross frequency. Note that λ , as in Eq. (5b), is restricted to be positive since negative prices have no meaning.

5.2 Client Controller

We now describe the main steps of the HAS client controller. The behavior of the controller is strongly based on Eq. (5a). However, we cannot simply use the aforementioned equation since buffer level variations as well as discrete sets of available bitrates need to be taken into account in practice. The full client algorithm, provided in Algorithm 2, is executed every time a chunk can be downloaded, i.e., anytime a download is finished and the playback buffer is not full (see downloading conditions – line 1,2).

As a first step, the client controller calculates the value of the ideal bitrate r_{coord} from the last received price value λ according to Eq. (5a). The coefficient κ is necessary to normalize the value of the price accordingly to the shape of the utility function and to

²Ideally the value of γ should be set to 1 to fully utilize the channel. In practical systems, however, we observed that $\gamma = 0.9 - 0.95$ provides less noisy results at the cost of marginal channel under-utilization.

Algorithm 2 Client Controller Algorithm

```
1: if Buffer_full or download active then
2:   return
3:
4:  $r_{coord} := [U'(\lambda/\kappa)]^{-1}$ 
5:  $\hat{r}_{TCP} :=$  last chunk TCP throughput
6:  $\hat{\alpha}_{TCP} := \alpha_{TCP}(\text{now} - \text{last TCP throughput update})/T_{ck}$ 
7:  $r_{TCP} \leftarrow \hat{\alpha}_{TCP}r_{TCP} + (1 - \hat{\alpha}_{TCP})\hat{r}_{TCP}$  ▷ LPF
8:  $r := r_{coord}$ 
9: if ( $r_{TCP} < r_{coord}$ ) and ( $B < T_{ck}(0.6M)$ ) then
10:    $r \leftarrow r_{TCP}$ 
11:  $B :=$  BufferLevel()
12:  $\delta := \max\left(1.0, \min\left(0.25, \frac{B}{T_{ck}(0.7M)}\right)\right)$ 
13:  $l \leftarrow \arg \max_{b(l') < r\delta} b(l')$ 
14: if  $l < l_{old}$  then
15:    $l \leftarrow \max(l_{old} - 1, l_{min})$ 
16: if  $l > l_{old}$  then
17:    $l \leftarrow \min(l_{old} + 1, l_{MAX})$ 
18:  $\hat{\tau} := \min(\text{last downloading time}, 1.25T_{ck})$ 
19:  $\tau \leftarrow \alpha_{\tau}\tau + (1 - \alpha_{\tau})\hat{\tau}$  ▷ LPF
20:  $\hat{q} := \max(1.0, r_{coord,old}/b(l_{old}))$ 
21:  $q \leftarrow \alpha_q q + (1 - \alpha_q)\hat{q}$  ▷ LPF
22: send the chunk request for bitrate  $b(l)$ 
23: send the corrected downloading time to coordinator  $q\tau$ 
```

assure the stability of the system. In a theoretical model, the controller would request a chunk of rate r_{coord} . However, we cannot fully neglect the experienced TCP throughput as well as the client buffer status in realistic implementations. For example, if the buffer level is very low and the rate suggested by the coordinator system is remarkably high compared to the measured TCP throughput, it might be a good idea to ignore r_{coord} and select the chunk according to the measured bandwidth only. This situation can occur during the startup phase or after a sudden drop of the available bandwidth. Therefore, the controller estimates the TCP throughput as described in [11] (lines 5-7) and selects which rate to use between the TCP throughput, r_{TCP} , and the ideal rate, r_{coord} . Basically, it selects the TCP throughput estimation only if $r_{TCP} < r_{coord}$ and if the video buffer level is below a certain threshold (we set this threshold to be equal to 60% of the maximum buffer occupancy since it offers a good tradeoff between avoiding buffer underruns and trusting generally the coordinator price) (lines 9-10).

Next, rather than selecting exactly r , the controller will search for a discounted value $r\delta$, with the discount factor δ defined in line 12, depends on the buffer level occupancy B . The discount factor is usually 1 at regime, but it reduces during re-buffering phases in order to decrease the rate of the requested chunks and refill the buffer faster. The discount factor takes values between 0.25 in low buffer conditions, and 1 when the buffer occupancy

is higher than 70%. Finally, the controller selects the chunk with the encoded bitrate that is closest to $r\delta$. In order to select the bitrate level l we first select the maximum bitrate lower than $r\delta$ (line 13) ($b(l)$ is the encoding bitrate for the representation l). Secondly, since large quality variations can be badly perceived by the user, we limit the variation of the representation index with respect to the previous selection l_{old} (lines 14-17). We then consider the downloading time of the previous chunk τ_{old} and we filter this variable using a LPF implemented as an EWMA with a coefficient equal to α_τ (line 18-19). The clipping and filtering of the downloading time is necessary to improve the coexistence with TCP in practice. When users compete against TCP flows, they can experience episodic downloading times that are remarkably larger than the average one, which may cause an unjustified price increase.

The last step of this practical implementation takes into account the quantization of the selected chunk rates, which affects the granularity of the downloading time values. Due to the rate discretization we cannot always guarantee an average maximum downloading time that matches exactly γT_{ck} . This can lead the controller to frequent oscillations in the price that then translate into annoying oscillations in the users bitrates selection. To overcome this problem, we introduce a new variable q , which keeps track of the ratio between the ideal rate and the actual requested bitrate (line 20-21). The value α_q corresponds to the coefficient of the EWMA, and $r_{coord,old}/b(l_{old})$ is the ratio between the previous ideal request, $r_{coord,old}$, and the previous chunk request, $b(l_{old})$. The key point is to perform an upscaling of the measured downloading time based on the experienced quantization step. In this way we are able to decrease the difference between the average downloading time of the most demanding user and the reference signal γT_{ck} , reducing the variations of the price. Note that the main drawback of the downloading time correction technique is an under-utilization of the channel (at regime $r_{coord,old} \geq b(l_{old})$), which is however balanced with the reduction of the frequent oscillations of the video quality. An alternative way to solve the bitrate discretization problem is to select the chunks in such a way that the average bitrate is equal to the coordinator rate. This method might be useful if we consider dynamic video complexity, i.e. dynamic utility functions. This is however beyond the scope of this work. As last step the controller sends to the video server the request for a chunk of bitrate equal to $b(l)$ and sends to the coordinator the scaled downloading time measurement equal to $q\tau$.

Finally, note that all the clipping operations implemented in the client algorithm are active exclusively during transitory phases, e.g., rebuffering events, therefore they do not affect the bitrate selection at regime.

5.3 Summary of the Proposed Controller

We conclude this section by listing some benefits of the proposed system.

- The coordinator node is extremely simple as it does not require any per user state information. The coordinator uses measurement collected from the users to compute a unique global signal that is then sent back to the users. Each user uses then this signal in the bitrate selection in order to increase the overall QoS. The bitrate selection is done in a fully distributed way to meet the HAS paradigm.
- The algorithm requires every user to send the downloading time of every chunk to the

coordinator node and to receive the price. However, the size of both measurement and price messages is very small (few bytes). Moreover the communication overhead grows only linearly with both the number of users and the number of chunks. As a result, the proposed system has a limited overhead also for large multi-user systems.

- The coordinator node can be located anywhere in the network as long as it is able to communicate with the HAS users that share the bottleneck link.
- In case of broken communication link between the client controller and the coordinator node, the client can simply fall back to a classical rate/buffer-based HAS controller. Users that are not able to communicate with the coordinator node will be perceived as cross-traffic by the other HAS users, which can still perform the optimal selection strategy.

6 System Evaluation

We now provide simulation results to evaluate the performance of the proposed system. We implement the algorithm described in Section 5 in the NS3 network simulator and evaluate it in different representative scenarios.

6.1 Experimental Setup

In order to evaluate the proposed algorithm we use the well-known Structural Similarity (SSIM) metric [12] as a utility function. We consider four types of videos with different properties: a high motion sport video, two medium complexity videos, a cartoon and a documentary, and a low complexity lecture video. The original videos have been downscaled to smaller resolutions and every resolution has been encoded at different bitrates using h264 codec [13]. We have then extracted the average SSIM of the encoded sequences at different bitrates. We have derived the following continuous model of the SSIM:

$$U_i(r) = a_i \cdot r^{b_i} + c_i, \quad (6)$$

where the coefficients a_i , b_i and c_i for the encoded sequence i are derived by curve fitting with the SSIM experimental points. The experimental SSIM data points and the fitting curves are depicted in Fig. 4. Note that a visually pleasant video usually has a SSIM score above 0.8 and a gain in SSIM of 0.05 might correspond to an increase of one point in Mean Opinion Score (MOS) [14].

In our simulations we identify each user with a single video at a given resolution, therefore with a single constant utility curve that is then used to execute the adaptation logic described in Subsection 5.2. We assume that each user knows the utility function of the requested video. In reality, this is possible by including this information in the Media Presentation Description (MPD) file of the video, or alternatively, the service provider can make it available on the server as secondary information. Another possibility is that the users implement a no-reference distortion model to assess the quality of the displayed video sequence.

We compare our algorithm with three HAS controllers proposed in the literature, namely a *conventional* HAS controller as described and implemented in [11], the Probe and Adapt

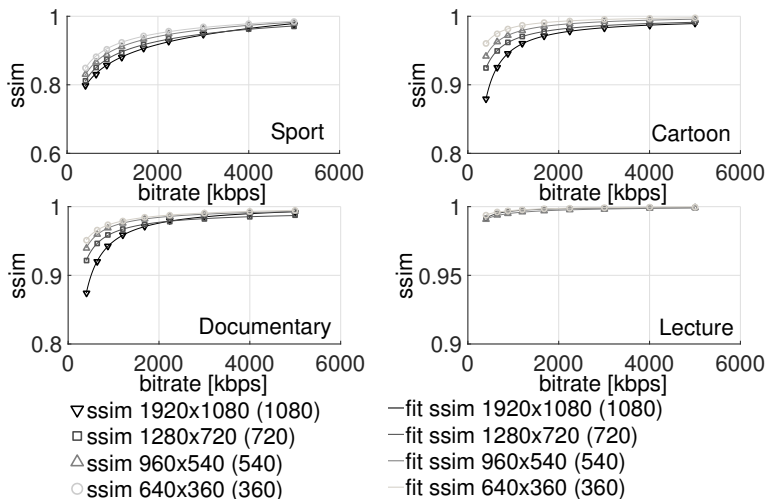


Figure 4: Quality-rate utility functions for the video sequences under consideration. Solid lines represent the continuous model of Eq. (6) while symbols are experimental measurements.

(PANDA) algorithm also proposed in [11], and the ELASTIC algorithm proposed in [15]. These three algorithms represent well the different behavior that rate-fair controllers can exhibit: PANDA is more conservative since it prefers to slightly underutilize the channel at the benefit of having a more constant bitrate selection. ELASTIC, on the other hand, strives to fully utilize the channel at the cost of more frequent quality variations. The conventional controller offers somehow an average behavior compared to the other two. To have a fair comparison among the different controllers, we fix the maximum buffer size of all the algorithms to M chunks, and we modify accordingly the parameters that control the buffer size in the baseline algorithms. In particular, the parameters B_{min} of PANDA and q_T of ELASTIC are both set to $6T_{ck}$. The other parameters of the baseline algorithms are set accordingly to the cited works. Note that in our work we do not consider freezing events as metric of comparison, therefore reducing the size of the buffer does not penalize any of the algorithms. For the proposed quality-fair algorithm, the value of the parameters are listed in Table 1. We set the values of these parameters in order to have: *i*) a good reactivity, thus good speed to convergence, *ii*) and clean signals, thus reducing the noise introduced by the network measurements.

Finally, the proposed controller as well as the baseline algorithms are tested over the network topology depicted in Fig. 5, where all users share the same bottleneck link. The links that connect the HAS users to the bottleneck link are local high-speed links. Lastly, the cross-traffic, if present, shares only the bottleneck link with the other HAS users.

6.2 Simulation Results

We now provide the simulation results carried out in the settings described above. We show first in detail how the proposed algorithm behaves. Then, we show the gain of our controller with respect to rate-fair controllers when the bottleneck is shared by many HAS

Table 1: Parameters used in the implementation

Parameter	value
All algorithms	
T_{ck}	2 s
M (Max buffer size)	10
Bitrates available	[400 640 880 1200 1680 2240 2800 3600 4400 6000] kbps
Proposed algorithm	
γ	0.95
α_e	0.75
K_p	1
K_i	0.25
κ	1e6
$\alpha_{TCP}, \alpha_q, \alpha_\tau$	0.75

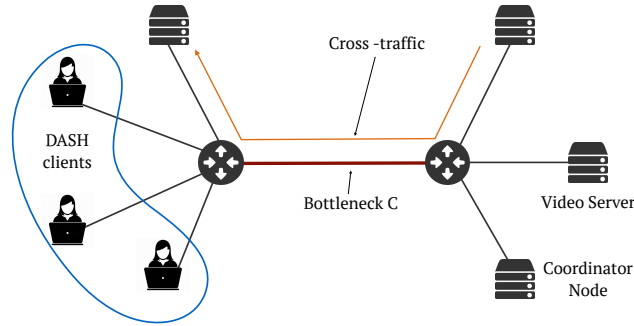


Figure 5: Topology used in the different simulated scenarios.

users. In a last set of simulations we evaluate the performance of the proposed algorithm when competing with cross-traffic.

In the first test case three HAS clients share a common bottleneck link that has a capacity of 5 Mbps. The Users 2 and 3, download the cartoon video at resolution 1080 and the lecture video at resolution 720, respectively, from the beginning of the simulation and stay always active, while user 1 downloads the sport video at resolution 540, between the timestamps 250s and 600s. The results are depicted in Fig. 6. In Fig. 6a, we provide both the video bitrate selected by the users and the ideal bitrates (r_{coord}) as described in Subsection 5.2. This plot shows the ability of the algorithm to fairly allocate the available bandwidth when client have different utility functions. Since user 1 is the one consuming the most complex video sequence, it is also the one that gets a larger portion of the channel link. From the SSIM curves, we know that for a bitrate of about 2.2 Mbps, user 1 experiences a SSIM value of approximately 0.94, while user 3 already achieves a SSIM value above 0.98 at 0.4 Mbps. Thus, the proposed controller is clearly able to improve the quality fairness among the users with respect to a rate-fair controller, which would allocate approximately

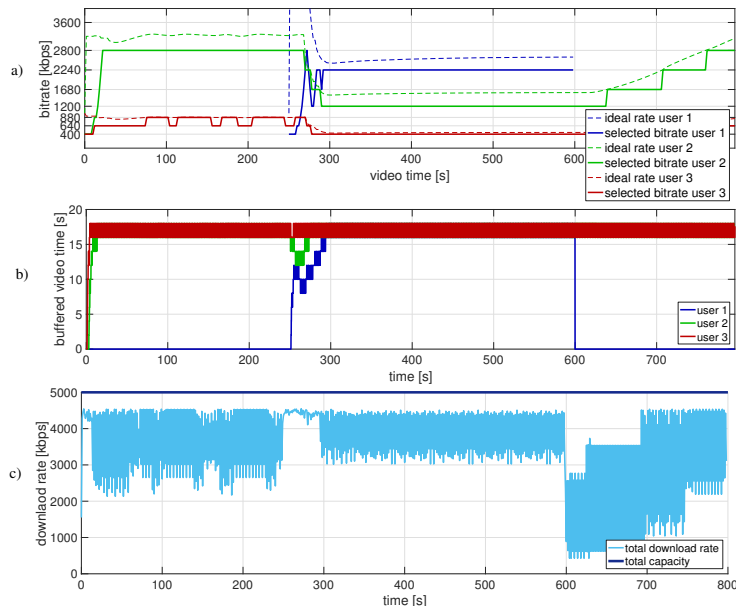


Figure 6: Performance of the proposed algorithm when three HAS users implementing our algorithm compete for the same bottleneck channel. The three plots respectively show the selected and ideal bitrates, the buffer occupancy and the channel utilization.

1.5 Mbps per user, making user 1 suffer of poor video quality while only slightly increasing the quality of user 3. Fig. 6b further shows the buffer level of the users. The playout buffers of all the three users have an occupancy level close to the maximum value, and no underruns are experienced during the simulation. The channel utilization, depicted in Fig. 6c, is also satisfactory. In fact the total download rate, given by the sum of the bitrates requested by all users, settles to a value that is close to the channel capacity. Note that the reported channel capacity corresponds to the physical bandwidth, which does not take into account the TCP/IP protocols overhead, thus it is not possible to exactly match its value.

We now consider N users, and we randomly assign to each of them a video at a given resolution and thus a corresponding utility curve. We then set the bottleneck capacity C to NC_{usr} , where C_{usr} is the average per user capacity. We consider 10 different realizations of the utility-user random selection, every metric shown in the final plots is the result of the average operation among the different realization. For each realization, we simulate the video streaming session where all users are simultaneously active for 460 seconds and we evaluate the average SSIM experienced at regime, i.e., after 60 seconds of video. Beyond the average SSIM, we also compute for each user the average SSIM variation per downloaded chunk as follows:

$$\Delta\text{SSIM}_{ck} = \frac{1}{L-1} \sum_{l=2}^L |\text{SSIM}(l) - \text{SSIM}(l-1)|, \quad (7)$$

where L is the total number of chunks downloaded by the user and $\text{SSIM}(l)$ is the SSIM value for chunk l . After we compute the average SSIM variation for every user, we average this value among the user population of the simulation. This metric quantifies the average

variation of quality level among consecutive chunks and captures possible SSIM oscillations rather than the simple heterogeneity of the SSIM over the all video sequence. Since it has been shown that frequent quality switches result in QoE degradation [16], the lower the ΔSSIM_{ck} the better the QoE. The last metric that we compute is the capacity usage, which is the time average cumulative downloaded bitrate of the users divided by the total capacity. A capacity usage close to 1 means an efficient use of the available resources. The three metrics above are evaluated in scenarios with different numbers of users, i.e., $N = [2\ 4\ 8\ 12\ 25\ 50\ 100]$, and different per user capacities, i.e., $C_{usr} = [0.75\ 1.25\ 2.0]$ Mbps. The corresponding results are depicted in Fig. 7. Every element of the box-plot is composed of *i*) a rectangle, which represents the first and third quartile divided by the median value *ii*) the whiskers, which delimit the minimum and the maximum value of the time-average SSIM among the user population and *iii*) the black dot which corresponds to the mean value over the population. We can notice that our algorithm is in general able to achieve better average quality compared with the rate-fair controllers. In particular the proposed algorithm is able to allocate more rate to the users that are watching high demanding videos. The minimum average SSIM of the proposed algorithm is remarkably higher than the one of the rate-fair controllers. By looking at the numerical values, it can be seen that our method can achieve a gain up to 0.05 points of SSIM for large values of N , and a gain of around 0.01 points of SSIM for small values of N . In general, the SSIM gain is larger for larger value of C_{usr} , since there is a larger margin of optimization in this case thanks to the larger amount of total bandwidth that can be re-allocated among the users. It is also worth noting that all the baseline algorithms show comparable performance among each other since they all target a rate-fair allocation. Beyond increasing the average SSIM, the proposed algorithm also reduced the average SSIM variations. As it is shown in the second column of Fig. 7, this value is substantially smaller than the variations experienced by the rate-fair controllers. The PANDA algorithm, since it is the most conservative, is the one behaving best among the three controllers used for comparison, as expected. From the third column of Fig 7, we can notice that the proposed algorithm is the one achieving the lowest bandwidth utilization. Nevertheless, the efficient usage of the bandwidth permits to the proposed algorithm to have better performances in the other metrics. The low bandwidth utilization is caused by the policy of selecting always a bitrate that is lower than the ideal bitrate. By applying a selection policy that targets a bitrate selection that is on average equal to the ideal rate the capacity usage can be increased, at the cost of more quality variations. Finally, note that we vary the number of users from a simple 2 users scenario to a scenario with 100 users, the proposed algorithm always achieves a better quality fairness with respect to rate-fair controllers, showing that our system scales well to large population of clients.

We further analyze the performance of our algorithm when the bottleneck capacity is shared with TCP cross-traffic for different amounts of TCP connections. We set the number of HAS users to $N = 16$ and then add different numbers of TCP connections, i.e., $N_{TCP} = [2\ 4\ 8\ 16]$; in percentage the amount of TCP cross-traffic varies accordingly from 11% to 50% of the total connections. We also vary the amount of the total capacity: $C = (N + N_{TCP})C_{usr}$, and the per user capacity is set to $C_{usr} = [0.75\ 1.25\ 2.0]$ Mbps in different simulations. We then compute the same metrics of the previous tests and the results are shown in Fig. 8. The average SSIM shows that the different algorithms are able to

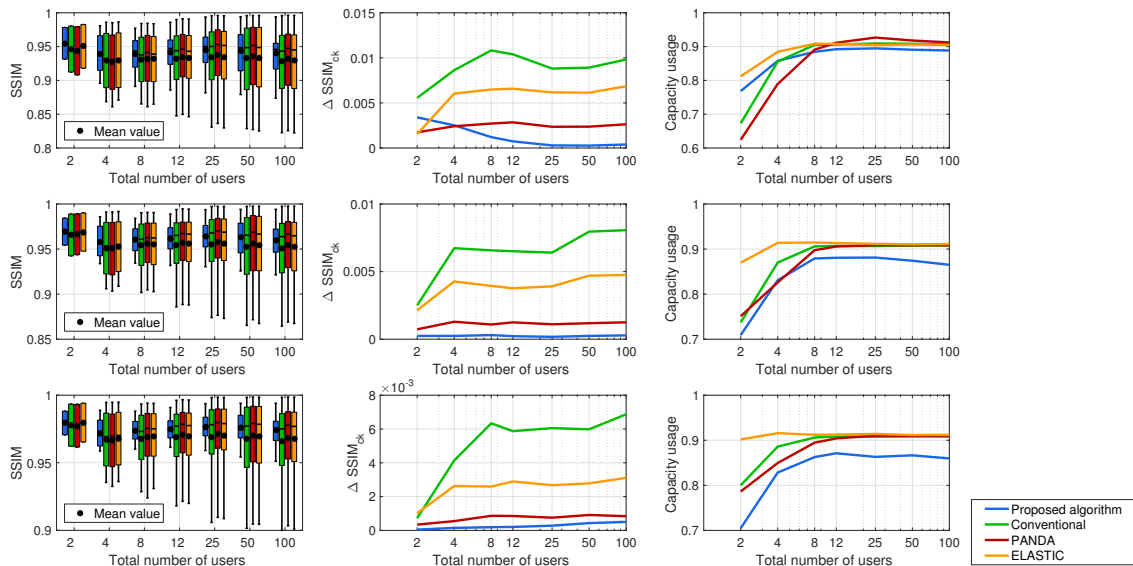


Figure 7: SSIM statistics, SSIM variations and channel utilization for the four implemented controllers for different numbers of users N . The per user capacity C_{usr} has been set to 0.75, 1.25 and 2.0 Mbps for the first, second and third row of plots respectively.

achieve approximatively the same performance. However, the proposed algorithm achieves higher values of minimum SSIM with respect to the rate-fair controllers. From the second column in Fig. 8, we see that the proposed method achieves the lowest SSIM variations in most of the cases, confirming the behavior of Fig. 7. In terms of channel utilization, ELASTIC is the algorithm that achieves the highest utilization ratio. Our algorithm instead has the lowest channel utilization together with the PANDA algorithm. We further notice that the sum of the HAS users utilization plus TCP utilization (in dashed lines) is close to one, as expected. We finally point out that our algorithm achieves approximatively the same average quality as the other algorithms using less bandwidth.

Finally, in the last set of simulations, we consider the scenario where only HAS users share the bottleneck channels, but with different controllers implemented at the client side. More in details, we have 4 HAS users, two with the proposed algorithm, two with one of the other baseline controllers. The users 1 and 2, which implement the proposed algorithm, download a high complexity video and a low complexity one respectively. The baseline controllers (users 3 and 4) are content agnostic, thus their behavior does not depend on the utility curve of the videos. The bottleneck capacity is set to 8Mbps, and all the users are simultaneously active during the simulation. The results are shown in Fig. 9. The green and blue bars correspond to the average bitrate requested by the clients implementing the proposed algorithm, while the two red bars correspond to the average bitrate requested by clients implementing one of the other controllers. The least fair scenario is the one in which the proposed algorithm competes with PANDA. This is expected since, as we have observed in the previous results PANDA is a very conservative algorithm. On the other hand ELASTIC, which is the most aggressive controller, achieves a larger downloading rate when competing with the proposed controller. The goal of this final tests is to show that the

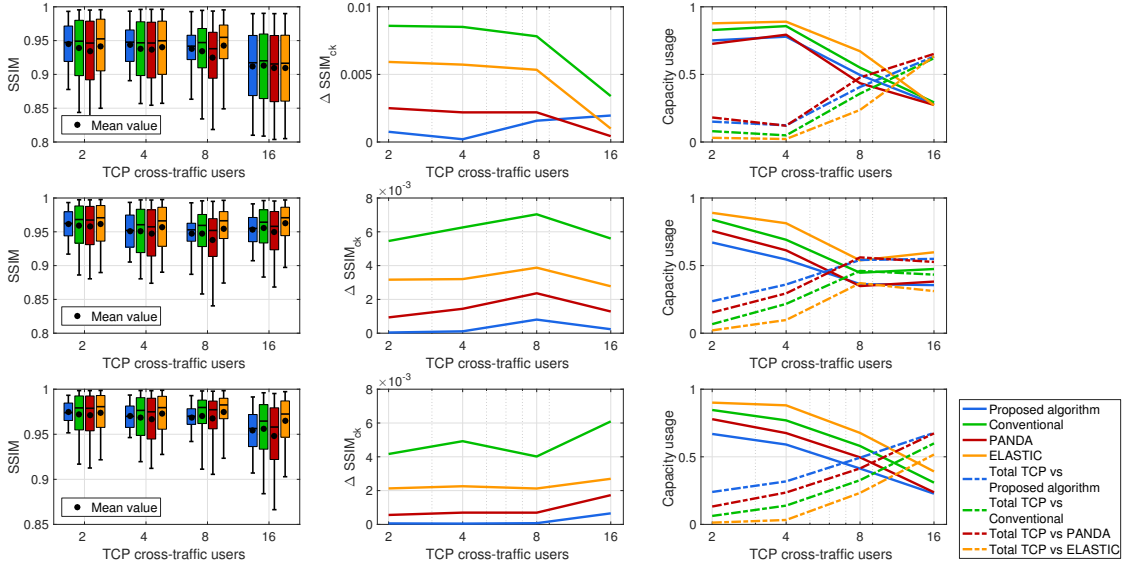


Figure 8: SSIM statistics, SSIM variations and channel utilization for the four implemented controllers for a set 16 HAS users sharing the bottleneck with a varying number of TCP flows. The per user capacity C_{usr} has been set to 0.75, 1.25 and 2.0 Mbps for the first, second and third row of plots respectively.

rate-fair HAS controllers neither dominate, nor are dominated by the proposed algorithm and that they can effectively coexist. Consequently we expect that in a scenario with a large number of rate-fair HAS controllers the performance achieved by our controller are comparable to the TCP cross-traffic results of Fig. 8.

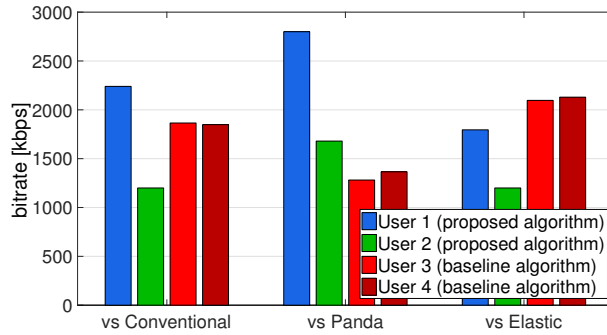


Figure 9: Average equilibrium bitrate achieved by the proposed controller when competing with the other rate-fair HAS controllers.

7 Conclusions

In this paper, we have proposed a price-based HAS controller that is able to enhance the overall QoS and improve quality fairness among HAS clients sharing a common bottleneck

link. Based on the experienced downloading times, a coordinator node evaluates the bottleneck price that reflects the congestion level of the network. The users then perform a quality-fair bitrate selection based on this price information. The ideal controller is adapted to work in realistic settings and tested in the network simulator NS3. The proposed algorithm is extremely scalable in terms of both computation and communication requirements. The simulation results show the ability of the proposed algorithm to work under different network conditions, and to improve the quality fairness of the users when compared to classical rate-fair controllers. The proposed controller is also able to work properly in scenarios where the bottleneck link is shared with TCP and other HAS cross-traffic. As future work, we plan to extend the proposed algorithm to multiple bottlenecks scenarios and to the case of dynamic utility functions, e.g., time varying video complexity.

References

- [1] T. Stockhammer, "Dynamic adaptive streaming over HTTP—: standards and design principles," in *Second annual ACM conference on Multimedia systems*. ACM, 2011.
- [2] E. Thomas, M. van Deventer, T. Stockhammer, A. Begen, and J. Famaey, "Enhancing MPEG DASH performance via server and network assistance," *The Best of IET and IBC*, 2015.
- [3] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Springer Journal of the Operational Research society*, vol. 49, no. 3, 1998.
- [4] A. El Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, "QoE-based traffic and resource management for adaptive HTTP video delivery in LTE," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, 2015.
- [5] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming," in *ACM SIGCOMM workshop on Future human-centric multimedia networking*. ACM, 2013.
- [6] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming," in *7th International ACM Conference on Multimedia Systems*. ACM, 2016.
- [7] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering stable high-quality video: An SDN architecture with dash assisting network elements," in *7th International ACM Conference on Multimedia Systems*. ACM, 2016.
- [8] S. Petrangeli, M. Claeys, S. Latré, J. Famaey, and F. De Turck, "A multi-agent Q-learning-based framework for achieving fairness in HTTP adaptive streaming," in *IEEE Network Operations and Management Symposium*. IEEE, 2014.

- [9] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, 2006.
- [10] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, *Control system design*. Prentice Hall New Jersey, 2001.
- [11] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, 2014.
- [12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, 2004.
- [13] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H. 264/AVC video coding standard," *Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, 2003.
- [14] L. K. Choi, Y. Liao, and A. C. Bovik, "Video QoE models for the compute continuum," *Multimedia Communications Technical Committee E-Letters*, 2013.
- [15] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: a client-side controller for dynamic adaptive streaming over HTTP (DASH)," in *20th IEEE International Packet Video Workshop*. IEEE, 2013.
- [16] D. Z. Rodríguez, Z. Wang, R. L. Rosa, and G. Bressan, "The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over HTTP," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, 2014.