Price-based Controller for Quality-Fair HTTP Adaptive Streaming

Stefano D'Aronco, Pascal Frossard
LTS4
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Email: name.lastname@epfl.ch

Laura Toni
Electrical and Electronic Departement
University College London (UCL)
Email: l.toni@ucl.ac.uk

***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.**

***Keywords*-HAS, quality-fairness, price-based mechanisms**

I. 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, 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 typically subdivided into chunks of few seconds, which are then downloaded by clients using HTTP requests over TCP. Each HAS client selects the best representation to download independently from the other clients with the aim of maximizing the downloaded bitrate while minimizing the possibility of rebuffering events. Therefore HAS systems are able to respond to the heterogeneous demands of several HAS clients in a fully distributed and adaptive way.

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 rate-fair utilization of the network resources where each user aims at maximizing the downloaded bitrate. Unfortunately, since video sequences generally have different characteristics, rate fairness does not necessarily translate into quality fairness. From this point of view, the selfish bitrate maximization of the users reveals its drawback. To overcome this main limitation, several works have been recently proposed in order to offer a coordinated solution for the rate allocation.

In this spirit and inspired by the Network Utility Maximization (NUM) framework commonly adopted in congestion control algorithms [2], we design a price-based distributed controller that maximizes the overall delivered Quality of Service (QoS) and improves the QoS fairness among users that share a common bottleneck. In order to reach this quality-fair rate allocation, we introduce a coordination node, that evaluates the congestion level, i.e. the price, of the network as a function of the downloading times of the chunks that are gathered from the HAS clients. Exploiting the computed price, the users can perform a proper bitrate selection in a fully distributed way. 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. 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.

The remaining of the paper is structured as follows. In Section II, we report some related works on the QoS enhancement in HAS systems. In Section III, we provide a description of the considered framework. In Section IV, we describe in detail the controller design. We present in Section V the simulations results. Finally, conclusions are provided in Section VI.

II. 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 [3], 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 [4]–[6] 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 [7], 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.

III. System Model

We now describe in detail the framework studied in this paper. We consider a HAS system with N users sharing a bottleneck link with an unknown available capacity C. This scenario reflects many realistic cases, for example a group of users sharing the same access link or connecting to the same server. The clients download video chunks of time duration T_{ck} by sending HTTP requests to the server. The clients store the received video data in the playout buffer, which can accomodate up to M chunks. After a download is over the next chunk 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. Therefore, when the buffer is full, requests are made every T_{ck}.

Let $\mathbf{r} = [r_1, r_2, \ldots, r_N]$ be the rate vector, with r_i being the last bitrate requested by user i. We then 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. We say that the rate vector $\mathbf{r}$ as *sustainable* if $\tau_i(\mathbf{r}) \leq T_{ck}$, $\forall i$, i.e., all users download their chunks in an amount of time that is sufficient to avoid buffer underruns.

Note that, since the bottleneck is shared by all users, $\tau_i(\mathbf{r})$ depends on the entire vector $\mathbf{r}$, and implicitly also on the capacity value C. In realistic environments the value of the downloading times depend also 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 assume in the theoretical design 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 characteristics of every rate-fair congestion control algorithm. To evaluate the proposed controller we then use, in the conducted simulations, a realistic packet level implementation of the TCP.

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. Users have different utility functions in order to model the different bandwidth requirements. We finally define the overall QoS of the system as the sum of the individual utility functions of every user: $\mathcal{U}(\mathbf{r}) = \sum_{i=1}^{N} U_i(r_i)$.

IV. Quality-Fair HAS Congestion Controller

In this section we describe in detail the proposed controller.

A. Theoretical Foundations

We now focus on the bitrate selection of the users at regime. In this phase users must exhibit an average downloading time smaller than T_{ck} in order to avoid buffer underruns, and consequently request one chunk every T_{ck}. The goal is to find a rate vector $\mathbf{r}$ that is sustainable and yet maximizes the aggregate utility. This can be achieved by solving the classical NUM problem:

$$\underset{\mathbf{r}}{\text{maximize}} \sum_{i=1}^{N} U_i(r_i) \quad \text{s. t.} \quad \sum_{i=1}^{N} r_i \leq C. \tag{1}$$

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

If we consider a continuous bitrate selection, the optimization problem in (1) can be solved distributively using dual decomposition (see [2], [8]) obtaining the following iterative system of discrete dynamic equations:

$$r_i^{k+1} = \left[U'_i(\lambda^k)\right]^{-1} \quad i = 1...N \tag{2a}$$

$$\lambda^{k+1} = \left(\lambda^k + \beta\left(\sum_{i=1}^{N} r_i^{k+1} - C\right)\right)_+ . \tag{2b}$$

Where λ is the dual variable, or price, associated to the bottleneck capacity constraint, $[U'_i(\cdot)]^{-1}$ represents the inverse of the derivative of the utility function of user i, $()_+$ denotes the projection onto the positive orthant, β is a simple parameter to control the step length of the dual variable

update and k simply denotes the iteration number which can be seen as a time stamp index. To evaluate the price update in Eq. (2b), 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 prospective cross traffic (which cannot be known in advance).

We therefore modify the second step of the iterative algorithm in order to avoid the explicit use of the capacity value. Following the assumptions made in the previous section about an ideal TCP behavior (a single active TCP connection is sufficient to fully utilized the channel and users request at least one chunk every T_{ck} in order to avoid buffer underruns) and the definition of sustainability, we can map the sum rate constraint into a downloading time constraint:

$$\sum_{i=1}^{N} r_i \leq C \iff \tau_{MAX}(\mathbf{r}) \leq T_{ck} \tag{3}$$

where $\tau_{MAX} = \max_{i=1...N} \tau_i$. In a more realistic scenario the equivalence has to be considered in terms of time-averaged values due to the random behavior of the packet transmission. By using the above equivalency we modify the dynamic system in (2) as follows:

$$r_i^{k+1} = \left[U'_i(\lambda^k)\right]^{-1} \quad i = 1...N \tag{4a}$$

$$\lambda^{k+1} = \left(\lambda^k + \beta(\tau_{MAX}(\mathbf{r}^{k+1}) - T_{ck})\right)_+ . \tag{4b}$$

The second step of Eq. (4b) can now be easily computed since every user knows the downloading time of the requested chunks, and the maximum value can easily be extracted.

We give now a brief discussion about how the iterative steps of system (4) can be computed in reality. In the first step of Eq. (4a), which corresponds to the adaptation logic, all the users independently compute the optimal bitrate and request the chunks to download at the next iteration. After a chunk 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. (4b). 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.

B. Controller Implementation

We now show how to adjust the iterative solution in (4) for it to be used in HAS systems in practice. In particular we take into account the discrete rate selection and the playout buffer management.

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. 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.

Algorithm 1 Coordinator Algorithm

```
1:  if New downloading time received from user then
2:      τ_MAX ← max(τ_MAX, τ_i)
3:      Send most recent λ to the user
4:
5:  loop                                   ▷ executed every T_ck
6:      ê := τ_MAX − γT_ck
7:      e ← α_e e + (1 − α_e)ê             ▷ LPF
8:      e_I ← max(0, e_I + e)
9:      λ ← max(0, K_P e + K_I e_I)        ▷ Update price
10:     τ_MAX ← 0                          ▷ Reset τ_MAX
```

The price update is executed every T_{ck}. In order to compute the error signal, $\hat{e}$, the coordinator node needs a reference signal γT_{ck}, which expresses the value of the maximum downloading time that the users must have at equilibrium[1]. The error between the maximum downloading time and the reference value is filtered by a Low Pass Filter (LPF) (line 7) with $\alpha_e = 0.75$. The filter is necessary to reduce the noise caused by the random TCP fluctuation. The error is integrated according to Eq. (4b) (line 8). The value of the final price λ is then calculated by combining the integral error and the proportional error (line 9), where $K_P = 1$ and $K_I = 0.25$ represent the proportional and integral gain respectively. Note that λ, as in Eq. (4b), is restricted to be positive since negative prices have no meaning.

2) Client Controller: We now describe the main steps of the HAS client controller. The behavior of the controller is strongly based on Eq. (4a). However, we cannot simply use the aforementioned equation since buffer level variations as well as the discrete set 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 (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. (4a). The coefficient κ is necessary to normalize the value of the price accordingly to the shape of the utility function and to assure the stability of the system. The controller estimates the TCP throughput as described in [9] (lines 4-6) and selects which rate to use between the TCP throughput, r_{TCP}, and the ideal rate, r_{coord}. The TCP throughput estimation is selected only if $r_{TCP} < r_{coord}$ and the video buffer level is below a defined threshold (lines 7-9). This is done in order to reduce the occurrence of rebuffering events. The controller calculates a discounted rate $r\delta$, where the discount factor δ depends on the buffer level occupancy B. The discount factor reduces the requested bitrate during

[1]Ideally the value of γ should be set to 1. In practical systems, however, we observed that using $\gamma = 0.95$ 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: $r_{coord} := [U'(\lambda/\kappa)]^{-1}$
4: $\hat{r}_{TCP}$:= last chunk TCP throughput
5: $\hat{\alpha}_{TCP} := \alpha_{TCP}$(now - last TCP throughput update)$/T_{ck}$
6: $r_{TCP} \leftarrow \hat{\alpha}_{TCP} r_{TCP} + (1 - \hat{\alpha}_{TCP})\hat{r}_{TCP}$ ▷ LPF
7: $r := r_{coord}$
8: **if** $(r_{TCP} < r_{coord})$ and $(B < T_{ck}(0.6M))$ **then**
9: $r \leftarrow r_{TCP}$
10: $B :=$ BufferLevel()
11: $\delta := \max\left(1.0, \min\left(0.25, \frac{B}{T_{ck}(0.7M)}\right)\right)$
12: $l \leftarrow \arg\max_{b(l')<r\delta} b(l')$
13: **if** $l < l_{old}$ **then**
14: $l \leftarrow \max(l_{old} - 1, l_{min})$
15: **if** $l > l_{old}$ **then**
16: $l \leftarrow \min(l_{old} + 1, l_{MAX})$
17: $\hat{\tau} := \min($last downloading time$, 1.25T_{ck})$
18: $\tau \leftarrow \alpha_\tau \tau + (1 - \alpha_\tau)\hat{\tau}$ ▷ LPF
19: $\hat{q} := \max(1.0, r_{coord,old}/b(l_{old}))$
20: $q \leftarrow \alpha_q q + (1 - \alpha_q)\hat{q}$ ▷ LPF
21: send the chunk request for bitrate $b(l)$
22: send the corrected downloading time to coordinator $q\tau$

re-buffering phases in order to refill the buffer faster. It then selects the chunk with the largest bitrate $b(l)$ that is smaller than $r\delta$, limiting the variation with respect to the previous chunk to one quality level (lines 10-16). Due to the bitrate discretization we cannot always guarantee an average maximum downloading time that matches exactly γT_{ck}, and this can possibly lead to frequent oscillations in the bitrates selection. To overcome this problem, we introduce a new variable q, which keeps track of the mismatch between the ideal rate and the actual requested bitrate (line 19-20) ($\alpha_q = 0.75$). The key point is to perform an upscaling of the measured downloading time based on the experienced quantization step in order to decrease the difference between the average downloading time and the reference signal γT_{ck}. Finally, 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 signal equal to $q\tau$, where the downloading time τ is filterd by a LPF (line 17-18) ($\alpha_\tau = 0.75$).

V. System Evaluation

We now provide simulation results of the proposed system implemented in the NS3 network simulator and evaluate it in different representative scenarios.

A. Experimental Setup

In order to evaluate the proposed algorithm we use the well-known Structural Similarity (SSIM) metric [10] as utility function. We consider four types of video 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 encoded at different bitrates and resolutions using h264 codec [11]. We then fit the experimental SSIM points in order to have a continuous utility function model (The experimental SSIM data points and the fitting curves are depicted in Fig. 1). In our simulations we identify each user with a single video at a given resolution. The utility curve associated to the video is then used in the adaptation logic. We assume that each user knows the utility function of the requested video. This information can easily be known in advance and transmitted from the server to the clients.

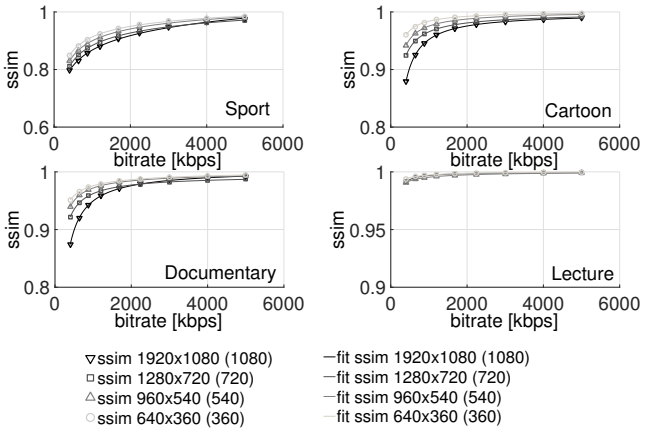

Figure 1. Quality-rate utility functions for the video sequences under consideration. Solid lines represent the continuous model while symbols are experimental measurements.

The length of the chunks is $T_{ck} = 2$ s and the available bitrates correspond to [0.4 0.64 0.88 1.2 1.68 2.24 2.8 3.6 4.4 6] Mbps. All the parameters of the proposed controller are set as specified in Section IV, while κ is set equal to $1e6$. We compare our algorithm with three HAS controllers proposed in the literature, namely a *conventional* HAS controller as described and implemented in [9], the Probe and Adapt (PANDA) algorithm also proposed in [9], and the ELASTIC algorithm proposed in [12]. To have a fair comparison among the different controllers, we fix the maximum buffer size of all algorithms to $M = 10$, which corresponds to having a $B_{min} = 6T_{ck}$ for PANDA and $q_T = 6T_{ck}$ for ELASTIC.

Finally, the proposed controller as well as the baseline algorithms are tested over the network topology depicted in Fig. 2, where all users share the same bottleneck link.

B. Simulation Results

In the first test case, three HAS clients share a common bottleneck link that has a physical capacity of 5 Mbps. The Users 2 and 3 download the cartoon and lecture video respectively, and are active for the entire simulation, while

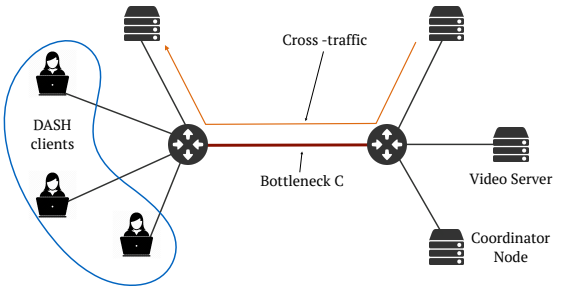

Figure 2. Topology used in the different simulated scenarios.

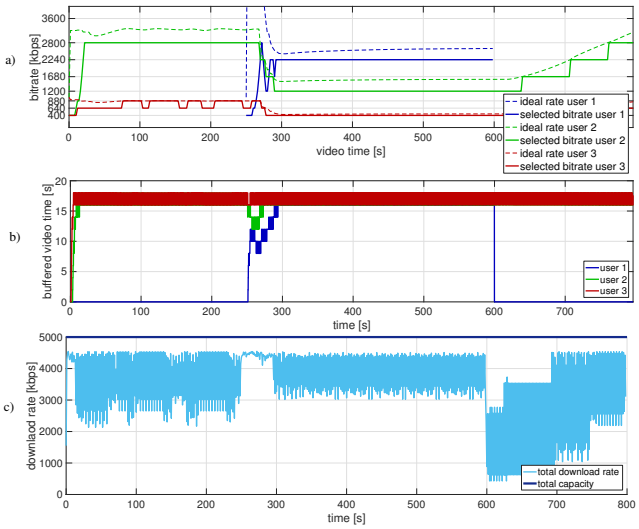

Figure 3. Three HAS users implementing our algorithm compete for the same bottleneck channel. The three plots respectively show the selected and ideal bitrates, the buffer occupancy and the channel utilization.

user 1 downloads the sport video (which is the most complex one) between 250s and 600s. The results are depicted in Fig. 3. In Fig. 3a, we provide both the video bitrate selected by the users and the ideal bitrates (r_{coord}) as described in Section IV. This plot shows the ability of the algorithm to allocate the available bandwidth consistently with the different utilities: user 1, being the one with the most complex video sequence, gets the largest amount of bandwidth when active. Fig. 3b 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. The channel utilization, depicted in Fig. 3c, is also satisfactory. In fact, the cumulative download rate of the users settles to a value that is close to the physical channel capacity.

We now consider the performance of our algorithm for different number of N users sharing a bottleneck link. We consider 10 different realizations of random utility-user assignments and we average every metric over these realizations. In this scenario, all the users are simultaneously active for 460 seconds and we evaluate the time-average SSIM value over the user population at regime. We also compute the average SSIM variation per downloaded chunk (ΔSSIM), which corresponds to the average absolute value of the SSIM difference between consecutive chunks. The last computed metric is to the capacity usage, which is the time average cumulative downloaded bitrate of the users divided by the total capacity. The three metrics above are evaluated in scenarios with a different number of users, i.e., $N = [2\ 4\ 8\ 12\ 25\ 50\ 100]$, with $C = N \cdot 1.25$ Mbps. The corresponding results are depicted in Fig. 4. The box-plot shows the minimum, the first and third quartile divided by the median and the maximum of the time-average SSIM value among the user 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 minimum average SSIM for the proposed algorithm is remarkably higher than the one of the rate-fair controllers. By looking at the numerical values, our method achieves a gain up to 0.05 points for the minimum SSIM score for large N. Beyond increasing the average SSIM, the proposed algorithm also reduces the average SSIM variations, see the second column of Fig. 4. From the third column of Fig 4, we can also 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. One way to improve this metric is by targeting a bitrate selection that is equal on average to the ideal rate, however, in this case the value of ΔSSIM would also increase.

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., $N_{TCP} = [2\ 4\ 8\ 16]$. The capacity is set to $C = (N + N_{TCP}) \cdot 1.25$ Mbps. We then compute the same metrics of the previous test, the results are shown in Fig. 5. The average SSIM shows that the different algorithms are able to 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. 5, we see that the proposed method achieves the lowest SSIM variations, confirming the behavior of Fig. 4. In terms of channel utilization, ELASTIC is the algorithm that has the highest utilization ratio. Our algorithm instead has the lowest channel utilization together with the PANDA algorithm.

VI. Conclusions

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

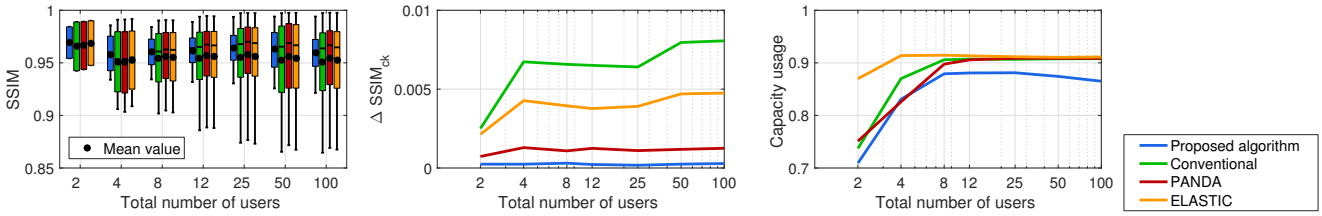

Figure 4. SSIM statistics, SSIM variations and channel utilization for the four implemented controllers for different numbers of users N, with $C = N \cdot 1.25$ Mbps.

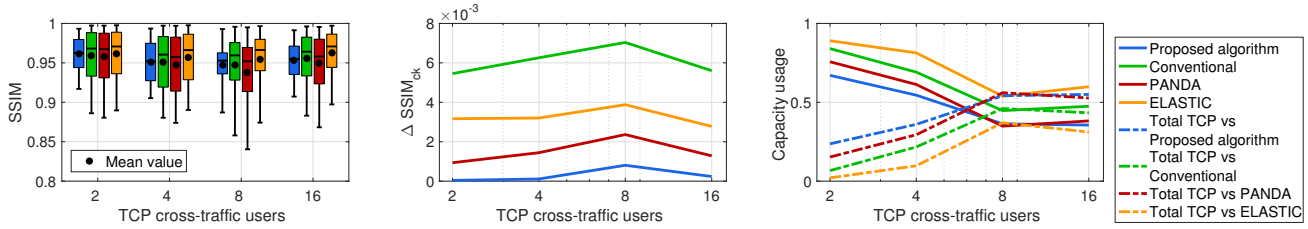

Figure 5. SSIM statistics, SSIM variations and channel utilization for the four implemented controllers for a set 16 HAS users sharing the bottleneck with a varying number of TCP flows, with $C = (N + N_{TCP}) \cdot 1.25$ Mbps.

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 simulation results show the ability of the proposed algorithm to work under different network conditions and for a large number of clients, and yet to improve the quality fairness of the users when compared to classical rate-fair controllers. As future works, 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.

Acknowledgment

This work has been supported by the Swiss National Science Foundation under grant CHISTERA FNS 20CH21 151569.

References

[1] T. Stockhammer, "Dynamic adaptive streaming over HTTP– : standards and design principles," in *Second annual ACM conference on Multimedia systems*. ACM, 2011.

[2] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, 1998.

[3] A. El Essaili, D. Schroeder *et al.*, "QoE-based traffic and resource management for adaptive HTTP video delivery in LTE," *Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, 2015.

[4] P. Georgopoulos, Y. Elkhatib *et al.*, "Towards network-wide qoe fairness using openflow-assisted adaptive video streaming," in *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, 2013.

[5] G. Cofano, L. De Cicco *et al.*, "Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. ACM, 2016.

[6] J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering stable high-quality video: An SDN architecture with dash assisting network elements," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. ACM, 2016.

[7] S. Petrangeli, M. Claeys *et al.*, "A multi-agent Q-learning-based framework for achieving fairness in HTTP adaptive streaming," in *Network Operations and Management Symposium*. IEEE, 2014.

[8] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *Journal on Selected Areas in Communications*, vol. 24, no. 8, 2006.

[9] Z. Li, X. Zhu *et al.*, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *Journal on Selected Areas in Communications*, vol. 32, no. 4, 2014.

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Transactions on Image Processing*, vol. 13, no. 4, 2004.

[11] 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.

[12] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: a client-side controller for dynamic adaptive streaming over HTTP (DASH)," in *International Packet Video Workshop*. IEEE, 2013.