Price-based Controller for Utility-aware HTTP Adaptive Streaming

Stefano D'Aronco[1], Laura Toni[2] and Pascal Frossard[1]

[1]LTS4, Ecole Polytechnique Fédérale de Lausanne (EPFL)
[2]Electrical and Electronic Departement, University College London (UCL)

January 7, 2017

Abstract

HTTP Adaptive Streaming (HAS) permits to efficiently deliver video to multiple heterogenous users in a fully distributed way. This might however lead to unfair bandwidth utilization among HAS users. Therefore, network-assisted HAS systems have been proposed where network elements operate alongside with the clients adaptation logic for improving users satisfaction. However, current solutions rely on the assumption that network elements have full knowledge of the network status, which is not always realistic. In this work, we rather propose a practical network-assisted HAS system where the network elements infer the network link congestion using measurements collected from the client endpoints, the congestion level signal is then used by the clients to optimize their video data requests. Our novel controller maximizes the overall users satisfaction and the clients share the available bandwidth fairly from a utility perspective, as demonstrated by simulation results obtained on a network simulator.

keywords– HTTP adaptive streaming, Pricing and resource allocation

1 Introduction

In the last decade video traffic has drastically increased to become the largest share of the total data transmitted in the Internet. This trend is expected to continue with video traffic reaching an outstanding share of 82% by 2020 [1]. However, for sustaining this growth there is the need to solve many challenges that arise from the fact that the Internet was not originally designed for media data transmission. For example, the Internet has been designed as a best-effort delivery network, and does not provide any sort of guaranteed Quality of Service (QoS) in general. As a consequence, the video transmission rates have to be adapted to the network condition variations in order to avoid dramatic congestions and large delays in video playback. Adapting the transmission rate to the available resources means varying the encoding bitrate of the transmitted video, hence the name *adaptive bitrate streaming* or more simply *adaptive streaming*. The way bitrate adaptation is performed strongly depends on what protocol is used for the video delivery. Whereas adaptive streaming systems initially used User Datagram Protocol (UDP) to transmit video data, an approach based on the HTTP (over TCP) protocol, known as HTTP Adaptive Streaming (HAS), has established as the universal solution for video distribution over the Internet in the last decade [2]. Transmitting video data using the HTTP protocol offers the following advantages over the UDP systems: *i*) Network Address

Translation (NAT) and firewalls can easily handle HTTP transmissions, whereas UDP flows might be blocked; *ii*) standard and widely deployed web server technology can be used; *iii*) since HAS is pull-based the adaptation algorithm resides exclusively at the client side, which results in a fully distributed and scalable algorithm that does not require to keep per client state information at the server. Nevertheless, HAS only provides a framework to deploy adaptive video streaming services, and many challenges remain open, especially on the design of the client adaptation strategy.

In more details, the video content in HAS systems is made available at the main server in different coded versions, namely representations, each one with a given bitrate and resolution. The representations are generally subdivided into chunks of few seconds, which are then downloaded by clients using HTTP requests over TCP. Chunks represent video segments that can be decoded independently, so that the users can request different representations for different video chunks. At the client side, the video is played while it is being downloaded. A playout buffer usually resides at the user side, in order to store video chunks that have been downloaded but not played yet. The role of the buffer is to absorb network bandwidth variations and avoid freezings (interruptions) of the video playback. The size of the buffer represents a first design tradeoff in HAS: a large buffer reduces rebuffering events caused by network condition variations, whereas a small buffer leads to a more responsive adaptation, being able to quickly benefit for large bandwidth when available. Normally each HAS client then implements a strategy that selects the best representation to download from the server with the goal of maximizing the downloaded bitrate while minimizing the occurrence of rebuffering events. The bitrate selection usually takes into account the buffer status and the estimated bandwidth. In such a way, HAS systems are able to respond to the heterogeneous demands of several HAS clients in a fully distributed and adaptive way.

It is important to note however that the use of HTTP requests for downloading video chunks prevents the HAS client controller from having full control of the transmission rate. When a chunk request is served, the TCP downloads the data as fast as possible without taking into account the actual bitrate of the video chunk. This TCP behavior complicates the bandwidth estimation in the adaptation logic at the client side and might further lead to unfair resource allocation when multiple users share the same bottleneck [3]. A large body of research has focused on designing HAS client controllers that guarantee a stable and rate-fair utilization of the network resources. Some of these works focus exclusively on improving the decision strategy made at the client side. Others investigate solutions where the bitrate selection becomes network-assisted, which means that the network elements provide some sort of support (e.g., bitrate selection guidelines) to help the client adaptation logic in order to improve fairness and network efficiency. The latter approach seems to go against the original principles of HAS, which ideally aims at having a fully distributed controller. However, due to the expected increase of the Internet video traffic which will put increasing pressure on the network infrastructure, it becomes important to reasonably relax the design constraints and consider any effective method that could help improving the video delivery.

In network-assisted HAS systems, we can distinguish two main approaches: *i*) methods that only supervise the client controller during the bitrate selection and *ii*) methods that modify the network behavior, which eventually affects the users bitrate selection. The works in the first category, e.g., see [4], typically rely on elements that monitor the client requests and the network link usage, and transmit information to the client controllers in order to guarantee fairness among the users. In the second category, the authors of [5] for example propose to have network elements performing a rate allocation among the video flows and reserving a defined bandwidth to each client. When the client controller estimates the download bandwidth, the estimation then matches the value of the reserved bandwidth, which eventually leads to a consistent bitrate selection. Both types of methods have pros and cons: forcing a defined bandwidth to each user does not require any modification of the client controller and it is resilient towards misbehaving users but it poses important conditions on

the network elements that must be able to perform a per user bandwidth reservation. On the other hand, signaling methods have only limited assumptions about the network elements but require a modification of the client controller for all the users of a particular video service.

Beyond having different methodologies, network-assisted HAS systems might also have different ultimate objectives, even if they all generally aim at increasing the overall users satisfaction. Some proposals use a policy that takes into account the perceived video quality of the different videos, e.g., [5,6], while other works are content agnostic and ignore video quality metrics, e.g., [4]. The latter methods are in general easier to implement, since they do not require any information about the video content, i.e., all videos have the same priority. However, it is well known that video sequences might have very different characteristics. The mere video rate is not an accurate measure of the quality seen by users and video quality fairness can only be achieved with methods that are adaptive to the video content. Finally, the proper consideration of the network technology is another critical factor in the design factor in network-assisted HAS systems. For example, the recent works in [6–8] operate in a Software Defined Networking (SDN) environment. SDN is an emerging technology that promises to bring more advanced features, such as network configurability, to the future Internet. Since the SDN provides tools for enabling QoS management to the Internet, network-assisted HAS systems can leverage SDN features to improve video delivery performance. However, the SDN technology is not currently widely deployed, and it is not clear whether and how the deployment of this technology will take place. Therefore, for an easy deployment, a network assisted HAS has to pose very limited assumptions regarding the technology used by the inner network nodes and possibly avoid the modification of the network elements that lie on the delivery path. Aside from these works which propose possible implementations of networked-assisted HAS, there are also efforts for the standardization of such a technology. The MPEG group is developing an extension of the Dynamic Adaptive Streaming over HTTP (DASH) standard [2] called Server and Network Assisted DASH (SAND) [9]. The extension provides guidelines about the communication between network nodes and the features that the network-assisted framework should possess, e.g., the system should be resilient to clients that ignore the network assistance. Standardization efforts certainly motivate the development of network-assisted systems and contribute to prepare future deployments.

A common aspect in all the above works, which is not necessarily a practical assumption, is that the network link capacities are assumed to be known. HAS systems might also use third party networks and might not have access to this information. In the case where no prior information on the network resources is available, is it still possible to properly coordinate the HAS users using measurements collected exclusively from the clients' endpoints? That is specifically the problem we address in this work. Our goal is to coordinate a set of HAS users sharing a common bottleneck of unknown capacity in order to maximize an overall users satisfaction metric, which we refer to as total utility. Moreover, we assume that *i*) the network elements are not able to alter the downloading rate of the users and *ii*) the user satisfaction depends on the content of the individual downloaded videos. We formulate the problem as a Network Utility Maximization (NUM) problem and we design a price-based distributed controller inspired by congestion control algorithms [10] that maximizes the overall utility. In order to enable our utility-aware rate allocation method, 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 by the HAS clients. This coordinator node however does not have to lie on the delivery path of the video. Using that price information, the users can perform a proper bitrate selection in a fully distributed way. In our algorithm, the users who achieve a satisfactory utility level with low bandwidth, do not increase the bitrate of the requested chunks in congested periods in favor of users downloading videos that are more demanding in terms of bandwidth. We test the proposed solution in a network simulator (NS3) under different network conditions and compare it with other rate-fair controllers proposed in the literature. The simulation results confirm the advantages that

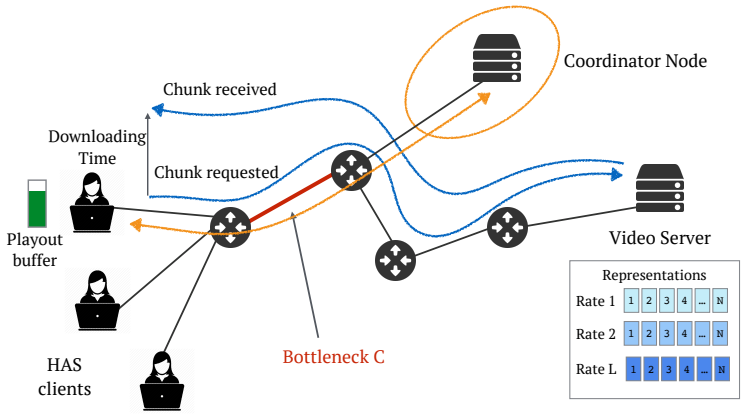


Figure 1: Overview of the considered scenario: Conventional HAS system and the infrastructure modification required by the proposed method (highlighted in orange).

can be achieved by utility-aware rate allocation methods with respect to the baseline rate-fair HAS controllers. The main advantage of our method is that it does not involve modifications of nodes that lie on the delivery path of the video data and it relies only on end-to-end measurements, the downloading time of the video chunks, for estimating the available resources and coordinate the users. Therefore, this work shows that the downloading time is not only an important variable that is used for conventional single user HAS controllers but it is a useful quantity for the development of network-assisted HAS systems.

The remaining of the paper is structured as follows. In Section 2, we provide a description of the considered framework. In Section 3, we describe how the NUM framework can be extended to the HAS scenario. We provide some details about the implementation of the proposed controller in Section 4. We present in Section 5 the simulations results. Finally, conclusions are provided in Section 6.

2 System Overview

The scenario investigated in this paper is depicted in Fig. 1. The HAS system is composed by N clients connected to a video server through a common bottleneck link of unknown capacity C. This scenario reflects many realistic cases, for example a group of users sharing the same access link, or a group of users connecting to the same server. Each client downloads video chunks of bitrate r and of fixed time duration T_{ck} by sending HTTP requests to the server. The clients then store the received video data in the playout buffer, which has a maximum capacity of M video 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 buffer slot becomes free. In particular, requests are made every T_{ck} when the buffer is full.

We denote the utility delivered to the user i with $U_i(r_i)$, where r_i represents the downloaded bitrate of user i. We define the total utility in the system as the sum of the individual utilities: $\mathcal{U}(\mathbf{r}) = \sum_{i=1}^{N} U_i(r_i)$. We assume that the shape of the utility function is a strictly increasing concave function, as it is common in the NUM framework. Considering that the users are ultimately interested in is the visual quality, we argue that the users' utility functions should also reflect some sort of visual

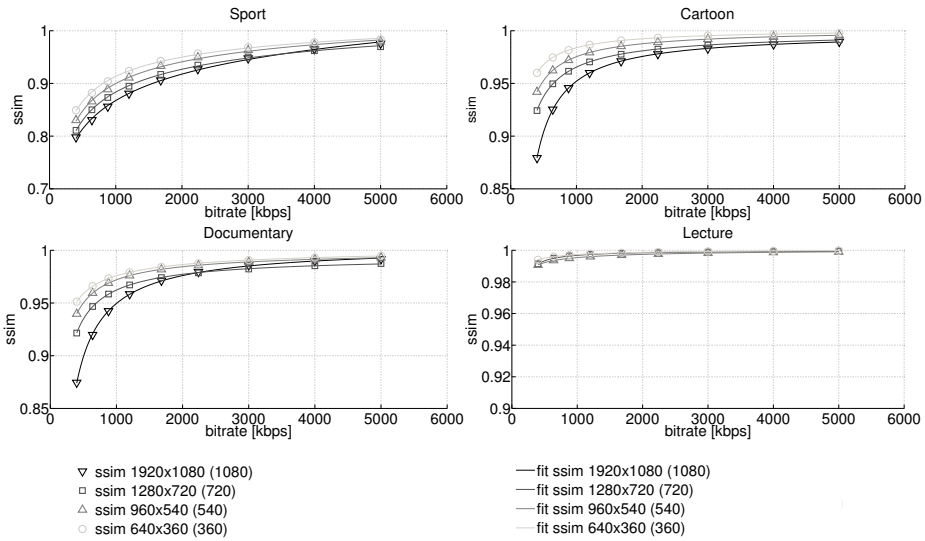


Figure 2: SSIM curves for sample videos with different content. As can be seen different video types exhibit different dependency on the encoding bitrate, low-complexity videos (lecture) achieve a much higher SSIM score when compared to high-complexity videos (sport).

quality metric for the downloaded video sequences. Since the content of the video sequences is usually different for the users, the utility functions are also different. In general, the visual quality of video sequence with low-complexity content grows quickly at low bitrate but saturates for larger bitrates. For high-complexity content, the quality grows more slowly with the bitrate so that a large bitrate is necessary to reach a satisfactory quality. In order to motivate the above statement, in Fig. 2 we show the average visual quality captured by the Structural Similarity (SSIM) metric [11] for four different video sequences encoded at different resolutions and at different bitrates. The SSIM is one of the possible video quality metrics that can be used as utility function but our work extends also to other quality metrics. The video quality scores achieved at the same bitrate for the different sequences are strikingly different. It becomes obvious that, since the user satisfaction depends on the video quality, the client bitrate selection should also be driven by the video content heterogeneity in order to fully benefit from the capabilities of the network-assisted HAS system.

Finally, we briefly motivate the concavity assumption made for the utility functions. In the literature some works argued that the concavity assumption does not always hold and proposed some other utility models for different settings (e.g. [12]). A discussion about the different implications that arise from the different shapes of the utility function goes beyond the scope of this work. However, concave functions, though being a restricted class of function, have been successfully used in the literature and in practice. In particular, the proposed method generalizes the conventional rate-fair HAS systems, which basically use a homogeneous concave utility function for all the different HAS users, and it extends them to cases with heterogenous (concave) utility functions for different video streams. The reasonable and commonly admitted assumption of concave utility functions further permits to develop a simple distributed rate allocation algorithm and to incentivize cooperation among the users.

3 Network Utility Maximization for HAS

We now briefly describe the NUM framework for the congestion control scenario, we then show how the problem and the solution method can be adapted to the HAS environment.

We now consider N users sending data packets through a link of capacity C. Moreover, as for our HAS model, we associate a utility function that depends on the transmitted rate to every user. The goal of maximizing the overall utility given the available network resources mathematically translates into the following optimization problem:

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

The variables r_i simply denote the rate at which packets of user i are transmitted. If the utility functions are concave, then the above optimization problem is a convex problem with a linear inequality constraint. The Problem (1) can be solved distributively using dual decomposition methods (see [10, 13]) obtaining the following iterative system of discrete dynamic equations:

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

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

where λ is the dual variable, or price, associated to the bottleneck capacity constraint, the operator $[U'_i(\cdot)]^{-1}$ represents the inverse of the derivative of the utility function of user i, and the notation $()_+$ denotes the projection onto the positive orthant. The variable β is a simple parameter that controls the step length of the dual variable update. Since the algorithm is an iterative method we index the iterations with the variable k in Eq. (2a) and Eq. (2a). In Eq. (2a) each user independently optimizes its transmission rate according to the most recent value of the price λ^k. Whereas in Eq. (2b) the dual variable is updated using the most recent rate values r_i^{k+1}. In the congestion control framework, the dual variable update corresponds to the dynamic law that governs the evolution of the packet queue in the buffer located before the bottleneck link. Consequently the price update operation is carried out implicitly by the network and the users can obtain the value of the price from the end-to-end delay measurements in order to optimize the transmission rate according to Eq. (2a).

Applying the above NUM framework to the HAS system is not completely straightforward. Problem (1) can be defined in the HAS framework with r_i corresponding to the selected bitrate of user i. The bitrate update equation, Eq (2a), also remains meaningful in the HAS context, but the price update equation, Eq. (2b), becomes problematic. In the HAS scenario, the queuing delay at the bottleneck buffer is completely uncorrelated from the selected bitrates because chunks are transferred on the top of TCP. Consequently, the price has to be evaluated differently, a viable solution is to use a coordination node, which then communicates the current price to the users. Since we further assume that the value of the capacity C is unknown, we also need to find an alternative update rule that does not explicitly use the value C. The price evolution in Eq. (2b) is governed by a simple rule: the price increases if the downloaded total bitrate exceeds the link capacity and vice versa. Therefore any other quantity that can signal the overuse (underuse) of the bottleneck capacity can be used to update the price. We argue that the average downloading time of the video chunks is a good candidate for representing the use of the bottleneck capacity in a HAS scenario. Similarly to queuing delay or packet losses in congestion control, a downloading time that exceeds the chunk video time can be interpreted as a signal that the network cannot sustain the selected bitrates. The

playout buffer of HAS client can handle occasional downloading times larger than chunk video times, but this is clearly not a sustainable situation in the long run. In this case the playout buffer would empty and the video playback would freeze. Hence, regardless of the bitrates selected by the users, we want all the users to experience an average downloading time smaller than the chunk video time T_{ck}, $\tilde{\tau}_i \leq T_{ck}$, where $\tilde{\tau}_i$ denotes the average downloading time of user i. If any of the users experiences a downloading time that is higher than T_{ck} the price should increase so that the selected bitrates decrease and consequently also the downloading times. Now, since a price increase caused by a high downloading time of a single user affects the selection of all the other users, can we guarantee that it does not compromise the efficient use of the network resources? Under the assumptions of an ideal TCP behavior that should not happen. These assumptions, which represent the ideal characteristic of every rate-fair congestion control algorithm, are: *i*) the bandwidth is always equally shared among the active connections, *ii*) the channel is fully utilized when at least one connection is active. In this case, we obtain the following equivalence:

$$\sum_{i=1}^{N} \tilde{r}_i \leq C \iff \tilde{\tau}_{MAX}(\mathbf{r}) \leq T_{ck} \tag{3}$$

where $\tilde{\tau}_{MAX} = \max_{i=1...N} \tilde{\tau}_i$ and $\tilde{r}_i$ denotes the average selected bitrate for user i. The equivalence can be understood by noting that if one user experiences an average downloading time equal to T_{ck}, it means that the user's connection is basically always active ensuring that the channel is fully utilized (note that users make one chunk request every T_{ck} when the buffer is full). Due to the ideal assumptions on the TCP, a connection that is always active means that any bandwidth left free by other users is not wasted, which guarantees an efficient network usage. The equivalency of the two conditions in Eq. (3) 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 of the congestion control. In practice, we therefore 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.

The above equivalence permits us to rewrite Eq. (2) as follows:

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

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

The price update operation of Eq. (4b) can now be easily computed since every user knows the downloading time of the requested chunks. More in detail, the entire operation flow is the following: 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 be downloaded at the next iteration. After a chunk download, every user sends the average measured downloading time to the coordinator node. 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.

4 System Implementation

The iterative procedure described in the previous section cannot be used directly in realistic settings. Even though the iterative system of Eq. (4) naturally leads to a condition where the playout buffers are full in average, it is advisable to take into account the current buffer status in the bitrate selection in order to avoid undesired rebuffering events. Moreover since the system is not able to instantly adapt

the price in case of a sudden capacity variation, the rate selection cannot be completely agnostic of the TCP throughput prediction, which should also therefore be taken into account.

4.1 Coordinator Node

As depicted in Fig. 1 the coordination node represents a general network endpoint that is able to communicate with the HAS users that share the bottleneck link. This endpoint can also coincide with the video server or with one of the user endpoints. Every time a user downloads a video chunk from the server, the updated average downloading time is sent to the coordinator node, which replies by sending the most updated value of the price back to the user. It also stores the received downloading time in order to track the maximum value among all the users. The coordinator node periodically performs the price update described in Eq. (4b). It can also perform other operations, such as a smoothing of the price value for a more stable bitrate selection, or the addition of a proportional error to the price value in order to improve the dynamic performance of the system. Finally, it is worth noticing that the coordinator node operations are extremely simple, thus preserving system scalability.

4.2 Client Node

A client needs to know the utility curve in order to select the bitrate values. In the case where the utility function reflects some video quality metric, there are several available options for providing this information to the client controller. For example: *i*) the quality information is computed when the video is encoded and it is made available at the server side in an auxiliary file; *ii*) the client application estimates the video quality using a no-reference method (no-reference methods estimate the video quality of a video using only the compressed signal). The second method has the advantage that it can be implemented without involving the server side, but the first choice is much simpler and provides more accurate values for the actual video quality.

Every time a chunk has to be downloaded the user is able to select the bitrate using Eq. (4a). The client controller should also take into account the buffer status and the average TCP throughput when making the bitrate selection in order to reduce the chances of rebuffering events. In our implementation, if the buffered video time is low and the estimated TCP throughput is smaller than the bitrate computed from Eq. (4a), then we neglect the price value and we select the bitrate to download according to the TCP estimate. In this case the downloaded bitrate is smaller and therefore safer in terms of rebuffering probability. A secondary problem arises in real world implementations: the ideal rate r_{ideal} computed using Eq. (4a) is a continuous variable whereas the available bitrates are discrete. In order to deal with this problem there are two possible choices, namely *i*) to always select the largest available bitrate that is smaller than the ideal rate r_{ideal}; *ii*) to adopt a selection strategy where the average selected bitrate is equal to the ideal rate r_{ideal}. The first choice, in contrary to the second one, has the advantage of a more stable bitrate selection but might lead to a lower channel utilization, since it is more conservative. In our case we implement the former solution and privileged stability with respect to channel utilization. Taking into account the different real-world problematics described above, the client controller selects the most suitable bitrate to download and issues the chunk request to the server. Once that the video chunk is downloaded the user updates the average downloading time and sends this information to the coordinator node, which sends back the most recent price value.

As final remark we would like to point out that the dynamic system of Eq. (4) can actually be implemented in many different ways and that our solution is not unique. In any case however, the system of Eq. (4) should represent the core of the actual implementation, which would aim at operating

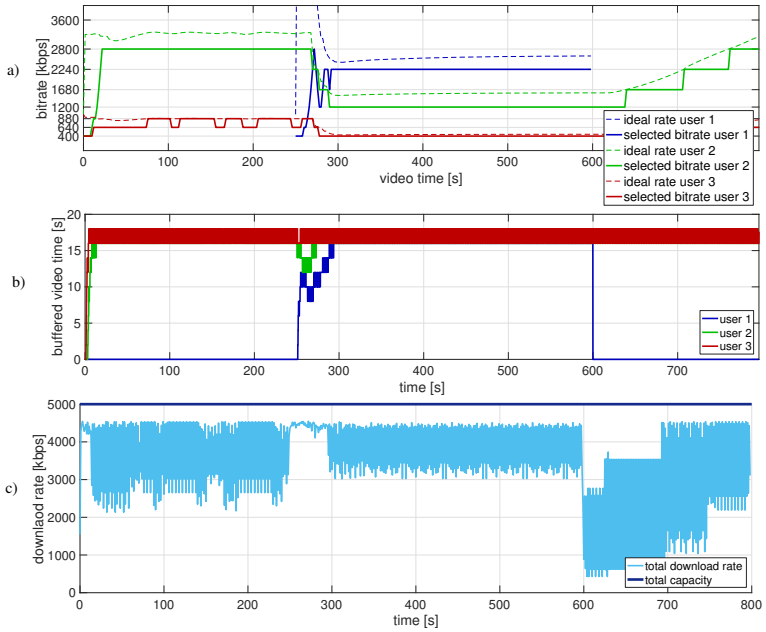


Figure 3: Three HAS users implementing our algorithm compete for the same bottleneck channel. The three plots respectively show the selected and ideal bitrates, the buffer occupancy and the channel utilization.

with the same equilibrium point as for the theoretical system.

5 Experimental Evaluation

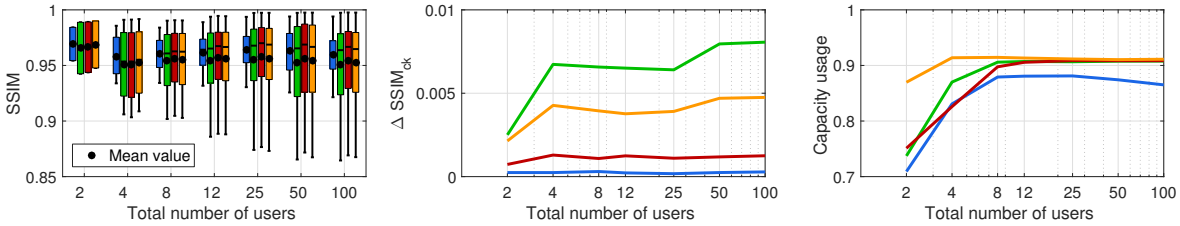
We have implemented the proposed system in the NS3 network simulator with different users requesting the different types of video. We use the SSIM score [11] as utility functions, depicted in Fig. 2. In our simulations we identify each user with a single video at a given resolution, therefore with a single constant utility curve that serves the adaptation logic. 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. For a detailed description of the actual implementation of the controller and for more simulation results, we refer the reader to our former paper [14] and to an extended version available online [15].

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 the 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_{ideal}) as described in Section 3. 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 (which has been set to 16s of video). 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.

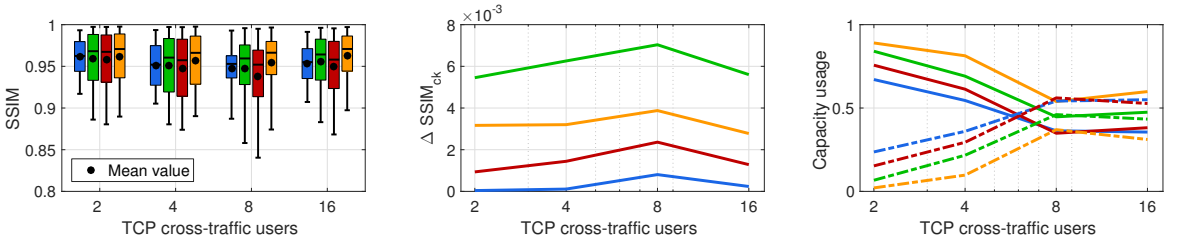
In the next simulations, we compare our algorithm with three HAS controllers proposed in the

literature, namely: the Probe and Adapt (PANDA) algorithm also proposed in [16], which is a conservative rate-based controller, that aims at having constant bitrate selection; the ELASTIC algorithm proposed in [17], which is a very aggressive buffer-based controller that strives to fully utilize the channel; and a *conventional* HAS controller as described and implemented in [16], which offers somehow an intermediary behavior compared to the other two. We investigate 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 metric is 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. 4a. 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 can achieve a gain of up to 0.05 point for the minimum SSIM score for large N. Beyond increasing the average SSIM, the proposed algorithm also reduces the average SSIM variations, as shown in the second column of Fig. 4a. From the third column of Fig 4a, we can also notice that the proposed algorithm is the one achieving the lowest bandwidth utilization compared to the baseline algorithms. Nevertheless, the more efficient usage of the bandwidth achieved by a smart bitrate selection permits to have better performances in the other evaluated metrics. The low bandwidth utilization is caused by the policy of always selecting a bitrate that is lower than the ideal bitrate. One way to improve this metric is to select a bitrate that is equal on average to the ideal rate; however, in this case the value of ΔSSIM would also increase.

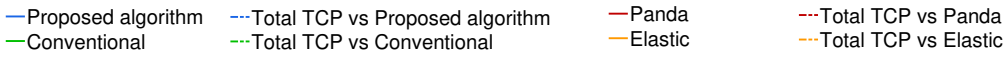
We further analyze the performance of our algorithm when the bottleneck capacity is shared with TCP cross-traffic for different amounts of TCP connections. This test is important since in realistic settings the network resources can be shared with other type of traffic, which commonly corresponds to TCP traffic. In this scenario, we verify that the proposed algorithm is effectively able to estimate the available resources and it does not get starved by the competing flows. 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 as in the previous tests and the results are shown in Fig. 4b. 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. 4b, we see that the proposed method achieves the lowest SSIM variations, confirming the behavior observed in Fig. 4a. In terms of channel utilization, ELASTIC is the algorithm that has the highest utilization ratio while our algorithm has the lowest channel utilization together with the PANDA algorithm. It is worth noting that the proposed algorithm achieves approximatively the same average quality as the other algorithms using less bandwidth. This spared bandwidth is not wasted, but it is used by the other TCP connections. Therefore, we can state that considering the whole set of users (HAS users plus cross traffic users) our method provides a higher overall users satisfaction.



(a) 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.



(b) 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.



(c) Legend.

Figure 4: Comparison with other rate-fair algorithms

6 Discussion and Future Works

The huge popularity of HAS is associated to mainly two characteristics: *i*) a full integration with the current deployed Internet infrastructure; *ii*) a high scalability due to a fully distributed controller with no per user state at the server side. Some recent studies [3] have revealed unfairness issues of some HAS adaptation logic because of erroneous bandwidth estimations at the client side. To overcome this limitation, several works in the literature proposed network-assisted HAS systems. In such systems some network elements interact with the HAS framework in order to assist the users and make a smarter bitrate selection that is able to ultimately improve the users satisfaction. Even though these modifications go against the original HAS principles, where the adaptation logic is completely distributed, their advantages are worth to be considered and studied.

Inspired by the NUM framework used in congestion control, we design a distributed network-assisted HAS system that coordinates users sharing a common bottleneck in order to maximize the delivered utility. The framework is based on the definition of a price that measures the congestion level of the bottleneck in order to coordinate the users. The price is related to the downloading time of the video chunks measured by the users and captures the underuse or overuse of the network resources. The coordination of users via this price signal permits to overcome the main limitation of most of the network-assisted HAS systems that rely on the prior knowledge of the available capacity. In the simulation results, we further 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 work, we aim at extending the system to the multiple bottlenecks scenario, where a different price is associated to every bottleneck. This poses important challenges for designing a proper price update strategy in order to guarantee an efficient use of the resources.

Acknowledgment

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

References

[1] "The zettabyte era: Trends and analysis," *Cisco, White Paper*, 2016.

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

[3] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *ACM 22nd international workshop on Network and Operating System Support for Digital Audio and Video*. ACM, 2012.

[4] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. D. Turck, "QoE-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 12, no. 2, 2015.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on selected areas in communications*, vol. 13, no. 7, 1995.

[13] 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.

[14] S. D'Aronco, L. Toni, and P. Frossard, "Price-based controller for quality-fair HTTP adaptive streaming," in *International Symposium Multimedia*. IEEE, 2016.

[15] ——, "Price-based controller for quality-fair HTTP adaptive streaming – extended version," *arXiv preprint arXiv:1701.01392*, 2017.

[16] 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.

[17] 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.