Price-Based Controller for Utility-Aware HTTP Adaptive Streaming

Stefano D'Aronco
Swiss Federal Institute of Technology

Laura Toni
University College London

Pascal Frossard
Swiss Federal Institute of Technology

A network-assisted HTTP Adaptive Streaming (HAS) system lets network elements infer network link congestion using measurements collected from client endpoints, helping clients optimize video data requests. A novel controller maximizes user satisfaction, and the clients share available bandwidth fairly.

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 percent by 2020.[1] However, to sustain this growth, we must solve many challenges arising from the fact that the Internet was not originally designed for media data transmission. The Internet was designed as a best-effort delivery network with no sort of guaranteed quality of service (QoS). Consequently, video transmission rates must be adapted to network condition variations to avoid dramatic congestions and large delays in video playback.

Adapting the transmission rate to the available resources means varying the encoding bit rate of the transmitted video—called *adaptive bit-rate streaming* or, more simply, *adaptive streaming*. The way bit-rate adaptation is performed strongly depends on what protocol is used for the video delivery. Whereas adaptive streaming systems initially used the User Datagram Protocol (UDP) to transmit video data, over the past decade, an approach based on the HTTP (over TCP) protocol, known as HTTP Adaptive Streaming (HAS), has become the universal solution for video distribution over the Internet.[2]

A HAS system efficiently delivers video to multiple heterogeneous users in a fully distributed way. However, it can lead to unfair bandwidth utilization among HAS users. Therefore, network-assisted HAS systems have been proposed, where network elements operate alongside with the client's adaptation logic to improve user satisfaction. Current solutions assume that network elements have full knowledge of the network status, but this isn't always realistic.

In this work, we propose a practical network-assisted HAS system, where the network elements infer the network link congestion using measurements collected from the client end points. The clients then use the congestion-level signal to optimize their video data requests. Our novel controller maximizes the overall user satisfaction, and the clients share the available bandwidth fairly from a utility perspective, as demonstrated by simulation results obtained on a network simulator.

HTTP Adaptive Streaming

Transmitting video data using the HTTP protocol offers several advantages over UDP systems. First, network address translation (NAT) and firewalls can easily handle HTTP transmissions, but they sometimes block UDP flows. Second, standard and widely deployed webserver technology can be used. Finally, because HAS is pull-based, the adaptation algorithm resides exclusively at the client side, which results in a fully distributed and scalable algorithm that doesn't require keeping per-client state information on the server. Nevertheless, HAS only provides a framework for deploying adaptive video streaming services, and many challenges remain open—in particular, how best to design the client adaptation strategy.

Video content in HAS systems is made available on the main server in different coded versions—that is, representations, each with a given bit rate and resolution. These representations are generally subdivided into chunks of a few seconds, which are then downloaded by clients using HTTP requests over TCP. Chunks

1070-986X/17/$33.00 © 2017 IEEE

Published by the IEEE Computer Society

represent video segments that can be decoded independently, so users can request different representations for different video chunks. On the client side, the video is played while it's being downloaded. A playout buffer usually resides at the user side to store video chunks that have been downloaded but not yet played. 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 the 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 that can quickly exploit large bandwidth when available. Normally, each HAS client implements a strategy that selects the best representation to download from the server with the goal of maximizing the downloaded bit rate while minimizing the occurrence of rebuffering events. The bit-rate selection usually takes into account the buffer status and the estimated bandwidth. In such a way, HAS systems can respond to the heterogeneous demands of several HAS clients in a fully distributed and adaptive way.

It's 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 bit rate of the video chunk. This TCP behavior complicates the bandwidth estimation in the adaptation logic on 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 bit-rate selection becomes network-assisted, which means that the network elements provide some sort of support (bit-rate selection guidelines, for example) to help the client adaptation logic to improve fairness and network efficiency. The latter approach seems to go against the original principles of HAS, because HAS aims to have 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 improve the video delivery.

In network-assisted HAS systems, we can distinguish two main approaches: methods that only supervise the client controller during the bit-rate selection, and methods that modify the network behavior, which eventually affects the users' bit-rate selection. The works in the first category (such as work by Stefano Petrangeli and his colleagues[4]) typically rely on elements that monitor the client requests and network link usage and that transmit information to the client controllers to guarantee fairness among the users. In the second category, Ali El Essaili and his colleagues,[5] for example, propose having network elements perform rate allocation among the video flows and reserve a defined bandwidth for 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 bit-rate selection. Both types of methods have pros and cons: forcing a defined bandwidth for each user doesn't require any modification of the client controller, and it's resilient toward misbehaving users, but it poses important conditions on the network elements, which must be able to perform a per-user bandwidth reservation. On the other hand, signaling methods have only limited assumptions about network elements but require a modification of the client controller for all users of a particular video service.

Beyond having different methodologies, network-assisted HAS systems might also have different objectives, even if they all generally aim at increasing overall user satisfaction. Some proposals use a policy that takes into account the perceived video quality of the different videos,[5,6] while other works are content agnostic and ignore video quality metrics.[4] The latter methods are, in general, easier to implement, because they don't require any information about the video content—that is, all videos have the same priority. However, it's 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

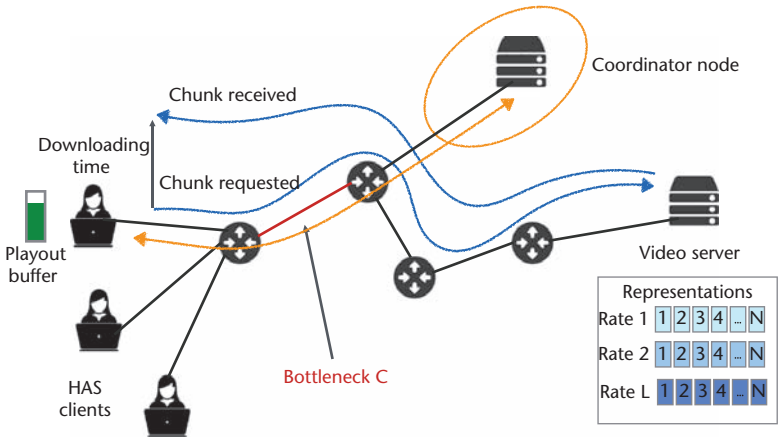

Figure 1. Overview of the considered scenario: The conventional HAS system and the infrastructure modification required by the proposed method (highlighted in orange).

design of network-assisted HAS systems. For example, some recent works[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. Because the SDN provides tools for enabling QoS management for the Internet, network-assisted HAS systems can leverage SDN features to improve video delivery performance. However, the SDN technology isn't currently widely deployed, and it's not clear how deployment of this technology will take place. Therefore, for an easy deployment, a network-assisted HAS must pose very limited assumptions regarding the technology used by the inner network nodes and should avoid modifying 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 to standardize such 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—for example, 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 of these works is that the network link capacities are assumed to be known—which is not necessarily a practical assumption. 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' end points? 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 to maximize an overall user satisfaction metric, which we refer to as *total utility*.

Moreover, we assume that the network elements can't alter the downloading rate of the users and that 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. To enable our utility-aware rate allocation method, we introduce a coordination node that evaluates the congestion level—that is, 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, doesn't have to lie on the delivery path of the video. Using that price information, the users can perform a proper bit-rate selection in a fully distributed way. 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.

System Overview

The scenario we investigate in this article is depicted in Figure 1. The HAS system comprises 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 connecting to the same server. Each client downloads video chunks of bit-rate 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.

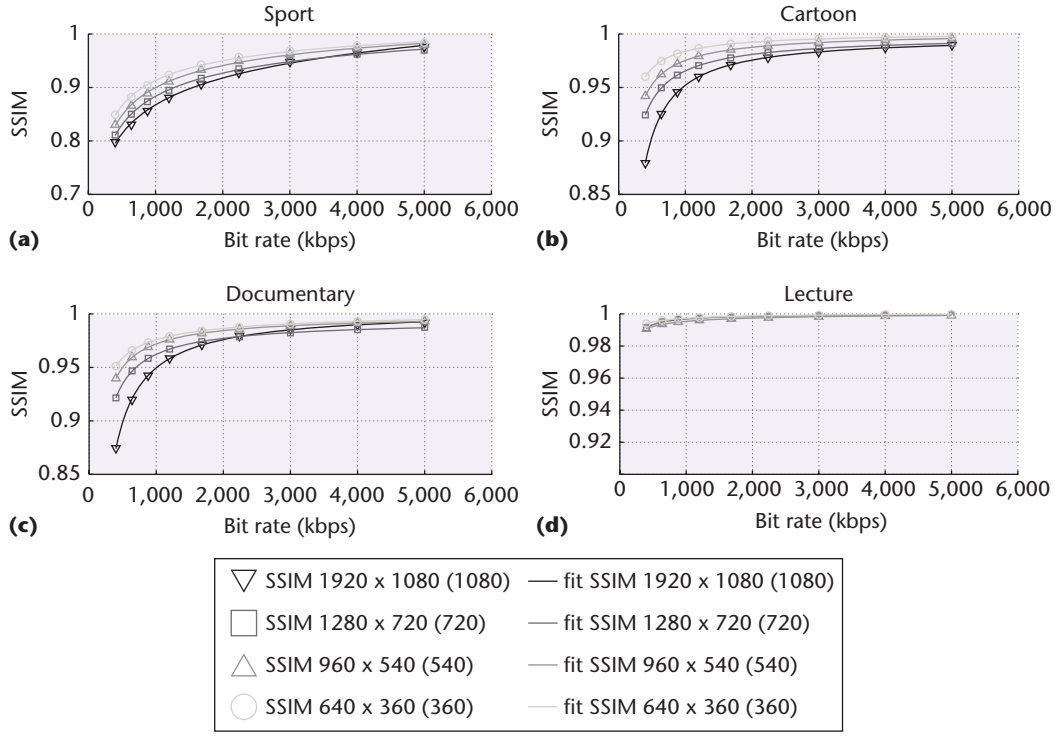

Figure 2. Structural similarity (SSIM) curves for sample videos with different content. As can be seen, different video types exhibit different dependency on the encoding bit rate, ranging from (a) high complexity (such as in a sports video) to (b–c) moderate complexity (such as in a cartoon or documentary) to (d) low-complexity (such as a lecture).

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

In general, the visual quality of the video sequence with low-complexity content grows quickly at low bit rates but saturates for larger bit rates. For high-complexity content, the quality grows more slowly with the bit rate so that a large bit rate is necessary to reach a satisfactory quality. To exemplify the importance of having heterogeneous utility curves among users, in Figure 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 bit rates. The SSIM is one of the possible video quality metrics that can be used as a utility function, but our work also extends to other quality metrics. The video quality scores achieved at the same bit rate for the different sequences are strikingly different. It becomes obvious that, because user satisfaction depends on video quality, the client bit-rate selection should also be driven by the video content heterogeneity to fully benefit from the capabilities of the network-assisted HAS system.

Finally, we point out that the concavity assumption of the utility function is necessary to have a more tractable mathematical problem. Even if this assumption doesn't hold in all settings,[12] it has been successfully used both in the literature and in practice. In particular, the resulting utility model corresponds to the framework used in conventional rate-fair HAS systems, which basically use a homogeneous concave utility function for the different HAS users. In addition, it lets us extend the framework to cases with heterogeneous (concave) utility functions for different video streams.

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.

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

$$\max_{r} \sum_{i=1}^{N} U_i(r_i) \qquad \text{s.t.} \quad \sum_{i=1}^{N} r_i \leq C \qquad (1)$$

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

$$r_i^{k+1} = \arg \max_{r_i'} U_i(r_i') - \lambda^k r_i' = [U_i'(\lambda^k)]^{-1} \qquad i = 1 \ldots N \qquad (2)$$

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

where λ is the dual variable, or price, associated with the bottleneck capacity constraint, and the operator $[U_i'(\cdot)]^{-1}$ represents the inverse of the derivative of the utility function of user i. 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. Because the algorithm is an iterative method, we index the iterations with the variable k in Equation 2 and Equation 3. In Equation 2, each user independently optimizes its transmission rate according to the most recent value of the price λ^k, whereas in Equation 3, 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 with 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 users can obtain the price value from the end-to-end delay measurements to optimize the transmission rate according to Equation 2.

Applying this NUM framework to the HAS system is not completely straightforward. Equation 1 can be defined in the HAS framework with r_i corresponding to the selected bit rate of user i. The bit-rate update equation, Equation 2, also remains meaningful in the HAS context, but the price update equation, Equation 3, becomes problematic. In the HAS scenario, the queuing delay at the bottleneck buffer is completely uncorrelated with the selected bit rates, because chunks are transferred on the top of TCP. Consequently, the price must be evaluated differently. A viable solution is to use a coordination node, which then communicates the current price to the users.

Because we further assume that the value of capacity C is unknown, we also need to find an alternative update rule that doesn't explicitly use the value C. The price evolution in Equation 3 is governed by a simple rule: the price increases if the downloaded total bit rate exceeds the link capacity and vice versa. Therefore, any other quantity that can signal the overuse (or 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. Similar to queuing delays or packet losses in congestion control, a downloading time that exceeds the chunk video time can be interpreted as a signal that the network can't sustain the selected bit rates. The playout buffer of the HAS clients 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. Consequently, regardless of the bit rates 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 users experience a downloading time that is higher than T_{ck}, the price should increase so that the selected bit rates 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 other users, can we guarantee that it doesn't compromise the efficient use of the network resources? Under the assumptions

of an ideal TCP behavior, that shouldn't happen. These assumptions, which represent the ideal characteristic of every rate-fair congestion control algorithm, are that the bandwidth is always equally shared among the active connections and that the channel is fully utilized when at least one connection is active. In this case, we obtain the following equivalence:

$$\sum_{i=1}^{N} \tilde{r}_i \leq C \iff \tilde{\tau}_{MAX}(r) \leq T_{ck}, \qquad (4)$$

where $\tilde{\tau}_{MAX} = \max_{i=1..N} \tilde{\tau}_i$ and $\tilde{\tau}_i$ denotes the average selected bit rate 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 about the TCP, a connection that is always active means that any bandwidth left free by other users isn't wasted, which guarantees efficient network usage. The equivalency of the two conditions in Equation 4 is true only if the ideal characteristic of the congestion control is verified. If this assumption doesn't hold, the equivalency is only an approximation whose accuracy depends on the actual behavior of the congestion control. In practice, congestion control algorithms are never ideal, so using the downloading time to detect channel overutilization is a heuristic approximation (suggested by the ideal model of the congestion control algorithms).

This equivalence lets us rewrite Equations 2 and 3 as follows:

$$r_i^{k+1} = [U_i'(\lambda^k)]^{-1} \qquad i = 1 \ldots N \qquad (5)$$

$$\lambda^{k+1} = \left(\lambda^k + \beta\left(\tilde{\tau}_{MAX}(\boldsymbol{r}^{k+1}) - T_{ck}\right)\right)_+. \qquad (6)$$

The price update operation of Equation 6 can now be easily computed, because every user knows the downloading time of the requested chunks. More specifically, the entire operation flow is as follows: in the first step of Equation 5, which corresponds to the adaptation logic, all users independently compute the optimal bit rate 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 Equation 6. The value of λ is then sent to the users for the next bit-rate selection. By performing these steps iteratively, the system converges to the optimal equilibrium point.

System Implementation

The iterative procedure described in the previous section can't be used directly in realistic settings. Even though the iterative system of Equations 5 and 6 naturally leads to a condition where the playout buffers are, on average, full, it's advisable to take into account the current buffer status in the bit-rate selection to avoid undesired rebuffering events. Moreover, because the system can't instantly adapt the price if there's a sudden capacity variation, the rate selection can't be completely agnostic of the TCP throughput prediction, which thus must be considered.

We'd also like to point out that the dynamic system of Equations 5 and 6 can actually be implemented in many different ways, and our solution is not unique. However, the actual implementation should aim to operate with the same equilibrium point as for the theoretical system.

Coordinator Node

As depicted in Figure 1, the coordination node represents a general network end point that can communicate with the HAS users who share the bottleneck link. This end point can also coincide with the video server or with one of the user end point. 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 to track the maximum value among all users.

The coordinator node periodically performs the price update described in Equation 6. It can also perform other operations, such as a smoothing of the price value for a more stable bit-rate selection, or the addition of a proportional error to the price value to improve the dynamic performance of the system. Finally, it's worth noting that the coordinator node operations are extremely simple, thus preserving system scalability.

Client Node

A client needs to know the utility curve to select the bit-rate values. In the case where the utility

function reflects some video quality metric, the quality information has to be extracted from the video data and provided to the client controller. This information can be estimated at the client side by using a no-reference method for the video quality evaluation (which estimates video quality using only the compressed signal). Alternatively, it can be measured during the video encoding process and made available at the server side as an auxiliary file. The first method has the advantage that it can be implemented without involving the server side, but the second option is much simpler and provides more accurate values for the actual video quality.

Every time a chunk must be downloaded, the user can select the bit rate using Equation 5. The client controller should also take into account the buffer status and the average TCP throughput when making the bit-rate selection 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 bit rate computed from Equation 5, then we neglect the price value and select the bit rate to download according to the TCP estimate. In this case, the downloaded bit rate 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 Equation 4a is a continuous variable, whereas the available bit rates are discrete.

To deal with this problem, there are two possible choices: always select the largest available bit rate that is smaller than the ideal rate r_{ideal}, or adopt a selection strategy where the average selected bit rate is equal to the ideal rate r_{ideal}. The first choice offers more stable bit-rate selection, but it might lead to lower channel utilization, because it's 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 problems described earlier, the client controller selects the most suitable bit rate 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.

Experimental Evaluation

We implemented the proposed system in the NS3 network simulator with different users requesting different types of video. We use the SSIM score[11] as utility functions, depicted in Figure 2. In our simulations, we identify each user with a single video at a given resolution and thus with a single constant utility curve that serves the adaptation logic. The length of the chunks is $T_{ck} = 2$s, and the available bit rates correspond to 0.4, 0.64, 0.88, 1.2, 1.68, 2.24, 2.8, 3.6, 4.4, and 6 megabits per second (Mbps). For a detailed description of the actual implementation of the controller, and for more simulation results, see our earlier paper[14] and an extended version of the paper available online.[15]

In the first test case, three HAS clients share a common bottleneck link that has a physical capacity of 5 Mbps. Users 2 and 3 download the cartoon and lecture video, respectively, and are active for the entire simulation, while user 1 downloads the sports video (which is the most complex one) between 250 s and 600 s (see Figure 3). In Figure 3a, we provide both the video bit rate selected by the users and the ideal bit rates (r_{ideal}) as described earlier. This plot shows the ability of the algorithm to allocate the available bandwidth consistently with the different utilities: user 1, who has the most complex video sequence, gets the largest amount of bandwidth when active. Figure 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 16 s of video). The channel utilization, depicted in Figure 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: the Probe and Adapt (PANDA) algorithm,[16] which is a conservative rate-based controller that aims at having constant bit-rate selection; the ELASTIC (*fe*edback *l*inearization *a*daptive *st*ream*i*ng *c*ontroller) algorithm,[17] which is a very aggressive buffer-based controller that strives to fully utilize the channel; and a conventional HAS controller,[16] which offers intermediary behavior compared to the other two. We investigate the performance of our algorithm for different numbers 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 users are simultaneously active for 460 s, 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 aggregate download rate of the users divided by the total capacity. The three metrics are evaluated in scenarios with a different number of users—$N = [2, 4, 8, 12, 25, 50, 100]$—with $C = N \times 1.25$ Mbps.

The corresponding results are depicted in Figure 4a. The box-plot shows the minimum, the mean (the first and third quartile divided by the median), and the maximum of the time-average SSIM value among the user population. Notice that our algorithm in general can 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 graph of Figure 4a. From the third graph in Figure 4a, note 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 bit-rate selection offers better performances in the other evaluated metrics. The low bandwidth utilization is caused by the policy of always selecting a bit rate that is lower than the ideal bit rate. One way to improve this metric is to select a bit rate 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, because in realistic settings, the network resources can be shared with other traffic types, which commonly correspond 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—that is, $N_{TCP} = [2, 4, 8, 16]$. The capacity is set to $C = (N + N_{TCP})1.25$ Mbps. We then compute the same metrics as in the previous tests and the results are shown in Figure 4b.

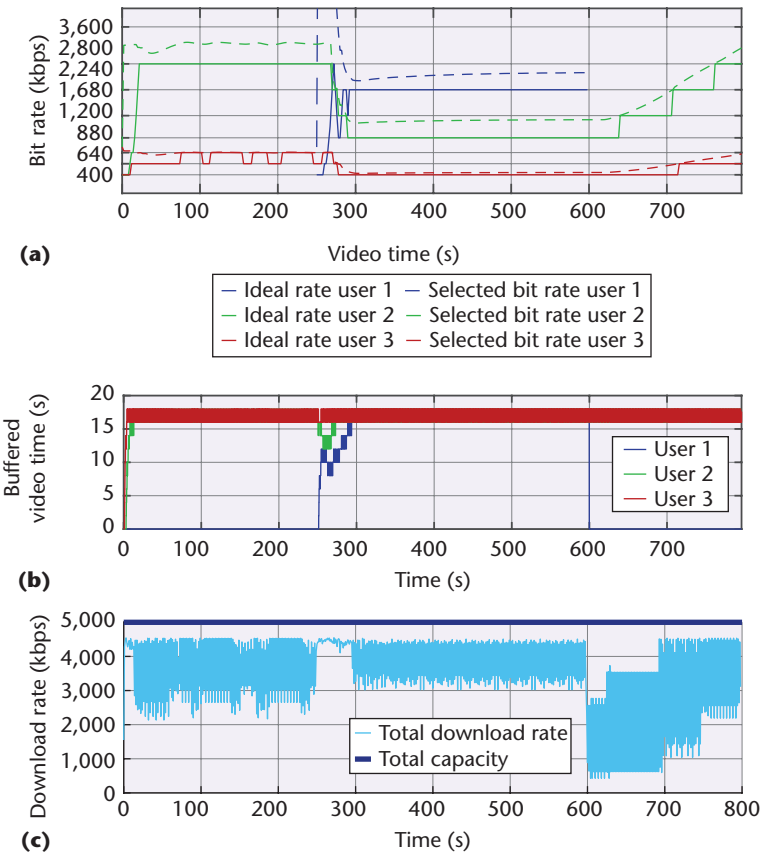

Figure 3. Three HAS users implementing our algorithm compete for the same bottleneck channel. The three plots show (a) the selected and ideal bit rates, (b) the buffer occupancy, and (c) the channel utilization.

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 graph in Figure 4b, we see that the proposed method achieves the lowest SSIM variations, confirming the behavior observed in Figure 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. The proposed algorithm achieves approximatively the same average quality as the other algorithms using less bandwidth. This spared bandwidth is not wasted, but it's used by the other TCP connections. Therefore, we can state that considering the whole set of users (HAS users plus cross-traffic users),

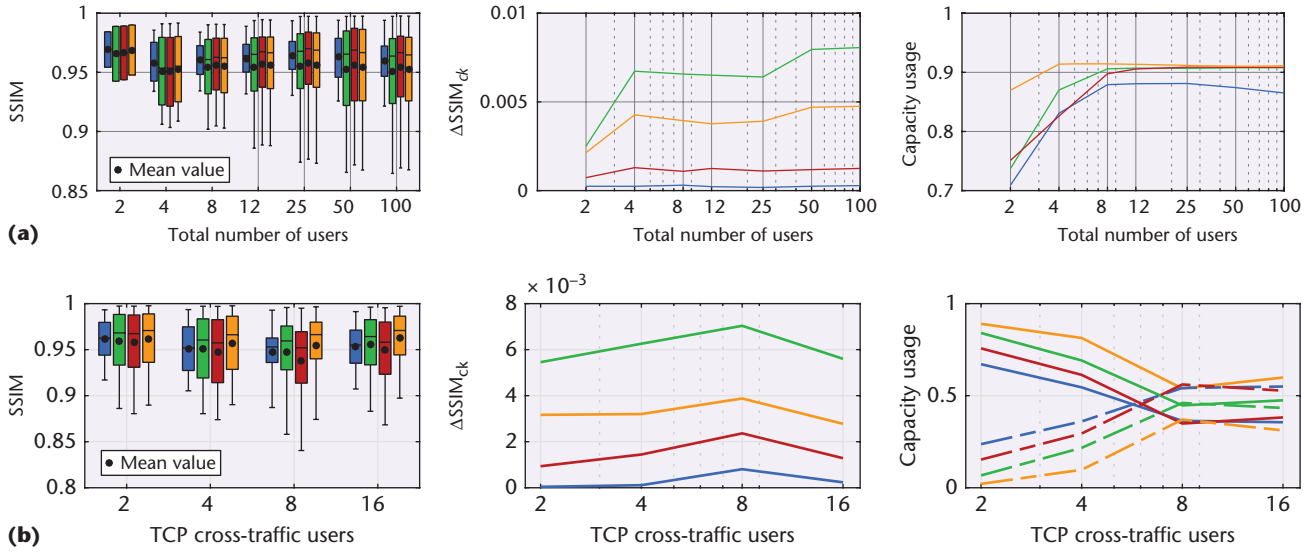

Figure 4. Comparison with other rate-fair algorithms: (a) SSIM statistics, SSIM variations, and channel utilization for the four implemented controllers for different numbers of users N, with $C = N \times 1.25$ megabits per second. (b) SSIM statistics, SSIM variations, and channel utilization for the four implemented controllers for a set of 16 HAS users sharing the bottleneck for a different numbers of TCP flows, with $C = (N + N_{TCP})1.25$ Mbps.

our method provides higher overall users satisfaction.

In future work, we aim to extend the system to the multiple bottlenecks scenario, where a different price is associated with every bottleneck. This poses important challenges for designing a proper price update strategy guarantee the efficient use of resources. **MM**

Acknowledgments

This special issue is a collaboration between the 2016 IEEE International Symposium on Multimedia (ISM 2016) and *IEEE MultiMedia*. This article is an extended version of "Price-Based Controller for Quality-Fair HTTP Adaptive Streaming," presented at ISM 2016. 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*, white paper, Cisco, 2016.
2. T. Stockhammer, "Dynamic Adaptive Streaming over HTTP–: Standards and Design Principles," *Proc. 2nd Ann. ACM Conf. Multimedia Systems* (MMSys), 2011, pp. 133–144.
3. S. Akhshabi et al., "What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth?" *ACM 22nd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, 2012, pp. 9–14.
4. S. Petrangeli et al. "QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming," *ACM Trans. Multimedia Computing, Comm., and Applications*, vol. 12, no. 2, 2015, article no. 28.
5. A. El Essaili et al., "QoE-Based Traffic and Resource Management for Adaptive HTTP Video Delivery in LTE," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 25, no. 6, 2015, pp. 988–1001.
6. P. Georgopoulos et al., "Towards Network-Wide QoE Fairness Using Openflow-Assisted Adaptive Video Streaming," *Proc. 2013 ACM SIGCOMM Workshop on Future Human-Centric Multimedia Networking* (FhMN), 2013, pp. 15–20.
7. G. Cofano et al., "Design and Experimental Evaluation of Network-Assisted Strategies for HTTP Adaptive Streaming," *Proc. 7th Int'l ACM Conf. Multimedia Systems* (MMSys), 2016, article no. 3.
8. J. W. Kleinrouweler, S. Cabrero, and P. Cesar, "Delivering Stable High-Quality Video: An SDN Architecture with Dash Assisting Network Elements," *Proc. 7th Int'l ACM Conf. Multimedia Systems* (MMSys), 2016, article no. 4.
9. E. Thomas et al., "Enhancing MPEG DASH Performance via Server and Network Assistance," *The Best of IET and IBC*, 2015, pp. 48–53.
10. F.P. Kelly, A.K. Maulloo, and D.K. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Springer J.*

Operational Research Soc., vol. 49, no. 3, 1998, pp. 237–252.
11. Z. Wang et al., "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Trans. Image Processing*, vol. 13, no. 4, 2004, pp. 600–612.
12. S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE J. Selected Areas in Comm.*, vol. 13, no. 7, 1995, pp. 1176–1188.
13. D.P. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE J. Selected Areas in Comm.*, vol. 24, no. 8, 2006, pp. 1439–1451.
14. S. D'Aronco, L. Toni, and P. Frossard, "Price-Based Controller for Quality-Fair HTTP Adaptive Streaming," *IEEE Int'l Symp. Multimedia* (ISM), 2016, pp. 113–118.
15. S. D'Aronco, L. Toni, and P. Frossard, "Price-Based Controller for Quality-Fair HTTP Adaptive Streaming—Extended Version," arXiv preprint, 2017; https://arxiv.org/pdf/1701.01392.pdf.
16. Z. Li et al., "Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale," *IEEE J. Selected Areas in Communications*, vol. 32, no. 4, 2014, pp. 719–733.
17. L. De Cicco et al., "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," *Proc. 20th IEEE Int'l Packet Video Workshop*, 2013; doi: 10.1109/PV.2013.6691442.

Stefano D'Aronco is a PhD candidate in the Signal Processing Laboratory (LTS4) at the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. His research interests include communication networks and multiagent systems. D'Aronco received an MS in electrical engineering from the University of Udine. Contact him at stefano.daronco@epfl.ch.

Laura Toni is a lecturer at the University College London. Her research interests wireless networks, interactive multimedia systems, and online decision-making strategies. Toni received her PhD in electrical engineering, from the University of Bologna. Contact her at l.toni@ucl.ac.uk.

Pascal Frossard is a professor at the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, where he heads the Signal Processing Laboratory (LTS4). His research interests include signal processing and image communications. Frossard received his PhD in electrical engineering from EPFL. Contact him at pascal.frossard@epfl.ch.