On Improving Video Streaming Efficiency, Fairness, Stability, and Convergence Time Through Client–Server Cooperation

Oussama El Marai, Tarik Taleb, *Senior Member, IEEE*, Mohamed Menacer, and Mouloud Koudil

***Abstract*—Many studies have shown that the dynamic adaptive streaming over HTTP scheme is limited in achieving efficiency, fairness, and stability. Solutions proposed in the recent literature target these objectives, but tackle them from either client or server side separately. This paper argues the use of a client–server cooperation-based approach to achieve the three objectives. Effectively, information available at the client side, such as buffer occupancy, available throughput, and previous played representation levels, can be used to better control the video streaming efficiency and stability at the client side. On the other hand, information available at the server side, such as the server's shared bandwidth capacity, and the number of connected clients and their corresponding downloading bitrates, can be leveraged to better tune the system fairness at the server side. Furthermore, the envisioned client–server cooperation aims at shortening the convergence time of the different clients to the fair bitrate allocation without affecting the overall system smoothness while increasing or decreasing the bitrates. The proposed approach is evaluated through extensive simulations using the Network Simulator, NS-3. Its performance is compared against that of notable algorithms, such as the FESTIVE [1] and PANDA [2] schemes. The obtained results show that the cooperation between the client and the server defines a promising approach in enhancing the efficiency, the fairness, the stability as well as shortening the convergence time to the fair bandwidth share.**

***Index Terms*—DASH, adaptive streaming, fairness, stability, efficiency, bitrate adaptation, convergence time.**

I. Introduction

THE MARKET of multimedia streaming over the Internet is continuously and rapidly growing. It has been attracting millions of users from various fields, such as education, media and entertainment. This expansion in terms of the number of multimedia viewers among the community of Internet users has been motivated by the technology evolution, lower cost of the user equipment (e.g., smart phones), and faster Internet connectivity.

Video applications usually manipulate huge content that require large bandwidths and dedicated protocols and methods to efficiently deliver the video content. A highly important video streaming approach that has attracted many researchers and practitioners in the last years is the Dynamic Adaptive Streaming over HTTP (DASH) scheme [3]–[9]. Despite its wide popularity, previous studies showed that DASH faces several challenges relevant to stability, fairness and efficiency, particularly when multiple clients compete for the same link, creating a bottleneck link scenario [1], [10], [11].

To achieve a stable system, the client should detect the short term bandwidth variations and accordingly avoid frequent short-term bitrate switches [12], [13]. The efficiency means that the client should watch the highest possible quality with respect to its available throughput [14]. Also, the system must ensure fairness among the connected clients. Thus, the quality of experience (QoE) perceived by the clients having equivalent bandwidth capacities should be close to each other [15], [16]. All these challenges make the design and development of a robust DASH system highly difficult since they are challenging and conflicting each other. Moreover, an ideal DASH system should allow quick convergence of the clients to the fair share and ensure smoothness when increasing or decreasing the quality. Effectively, frame freezes shall be avoided when switching to the next higher/lower level and only the right number of chunks shall be requested from each level.

To mitigate these issues, many solutions have been proposed aiming to overwhelm one or more of the previously mentioned goals. These solutions strive to deal with these objectives from either client side [1], [2], [17], [18] or server side [19]. The main advantage of the client side solutions is that all the adaptation logic resides at the client's player and no modification is needed to the webserver, which allows a quick and easy deployment. However, relying only on the information available at client side may lead to unfair share of the bandwidth and adversely affect the QoE by making the clients unstable due to on-off periods. Alternatively, a server side solution disposes of the information (e.g., shared bandwidth and total number of competing clients) that may help maintain higher fairness. Thus, the cooperation between the client and the

Manuscript received September 6, 2016; revised January 10, 2017 and June 2, 2017; accepted June 21, 2017. Date of publication December 29, 2017; date of current version March 2, 2018. *(Corresponding author: Tarik Taleb.)*

O. El Marai is with LMCS Laboratory, Ecole nationale Supérieure d'Informatique ESI, Algiers 16309, Algeria, and also with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 15500 Espoo, Finland (e-mail: o_el_marai@esi.dz).

T. Taleb is with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 15500 Espoo, Finland, and also with the Department of Computer and Information Security, Sejong University, Seoul 143-747, South Korea (e-mail: talebtarik@ieee.org).

M. Menacer is with the Department of Information Systems, College of Computer Science and Engineering, Taibah University, Medina 42353, Saudi Arabia (e-mail: eazmm@hotmail.com).

M. Koudil is with LMCS Laboratory, Ecole nationale Supérieure d'Informatique ESI, Algiers 16309, Algeria (e-mail: m_koudil@esi.dz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2017.2781146

server may help designing a robust DASH-based system that efficiently achieves the aforementioned goals and considerably improves the users' QoE. It should be mentioned that any solution that involves the server in the rate adaptation logic is only possible when the content provider (e.g., YouTube) adopts and implements it in its servers.

In this paper, a new stable solution, dubbed ESTC (Enhancing Server and clienT Cooperation), is presented. The design goals of ESTC is to improve both the efficiency and the fairness when multiple clients compete for a bottleneck link. Furthermore, ESTC shall allow a quick convergence of the different clients to the fair share while ensuring smooth playback of the videos at the client side. The key idea beneath ESTC is to establish cooperation between the client and the server in order to take the right decision about the allocated bitrate. The client is responsible for taking the right decision that maximizes the video stream efficiency and stability using information available at the client such as the actual throughput, the buffer occupancy and the history of the bitrate switches. At the server side, the number of connected clients, their current downloading bitrates and the bottleneck link capacity are used to ensure fairness among the competing clients, similar in spirit to [20] and [21].

This paper is organized in the following fashion. Section II provides a global overview on previous research work relevant to the DASH technology. In Section III, the problem is formulated, and the proposed ESTC solution is introduced in Section IV. The different metrics used to evaluate ESTC are defined in Section V. The performance evaluation of ESTC is carried out in Section VI, comparing it with the FESTIVE [1] and PANDA [2] algorithms. The paper concludes in Section VII, recapping the main advantages of ESTC and highlighting some future research directions.

II. Related Work

DASH is the standard provided by MPEG for HTTP Adaptive Streaming (HAS). It defines the structure of the Media Presentation Description (MPD) file, the segment formats and the protocol used for delivering the segments, namely HTTP/1.1 [3]. A typical DASH client follows a pull-based paradigm to retrieve the video segments. This enables DASH clients to deal with the changing conditions in order to maintain a stall-free session. As a drawback of this approach, the clients' decisions are based merely on the available information at their ends, without taking into account the server and network conditions. Furthermore, the service providers may face a QoS-related issues, e.g., the reliability of the media content information (e.g., outdated MPD file at CDN) and clients' oscillations when competing for a bottleneck [22], which may lead to the degradation of the users' QoE. To mitigate these issues, an extended version of MPEG-DASH standard [23] was published recently, namely SAND (Server And Network-assisted DASH). SAND aims at promoting the clients' awareness of the network conditions by introducing new messages and mechanisms in order to address the aforementioned issues. A detailed description of the SAND architecture and its deployment scenarios can be found in [22].

Since the appearance of the DASH technology, many commercial implementations have been proposed, such as Microsoft Smooth Streaming over HTTP, Apple HTTP Live Streaming (HLS), Netflix players and Adobe HTTP Dynamic Streaming (HDS). Akhshabi *et al.* [10] describe an experimental evaluation of two DASH-based commercial players. They also provide an open-source player, called the OSMF player, specifically designed to cope with persistent and short-term bandwidth variations. They found that the three players present significant inefficiencies in terms of stability, fairness and bandwidth utilization which degrade the user QoE. This observation was confirmed in [1] and [11], especially when multiple players compete for a shared bandwidth.

To cope with these limitations, various approaches have been proposed to cope with one or more of the three objectives, namely efficiency, fairness and stability. These approaches can be classified as either purely client- or server-based. Jarnikov and Özçelebi [24] proposed a stochastic control strategy model using Markov Decision Processes (MDP) to compute a set of optimal client strategies to switch between quality levels. This control strategy minimizes both the deadline misses of the chunks and the number of quality level changes. However, it has two major limitations: the first consists in the off-line calculation of the optimal control strategy that does not appropriately account for the dynamics of the throughput [25], and the second pertains to the high computation cost [17]. An improvement of this work was proposed by Bokani *et al.* [17] in order to reduce the MDP computation overhead by considering both online and offline MDP optimization in mobile environments.

Liu *et al.* [26] proposed a serial and parallel rate adaptation algorithm in Content Distribution Networks (CDNs). In serial method, the next segment cannot be requested until the complete reception of the previous one whilst in parallel method, multiple segments are requested. In this research, the authors introduced a new rate adaptation metric to detect the congested and spare network and give higher priority to new connected clients to fairly share the bandwidth. In their results, the authors reported that the parallel fetching method outperforms the serial algorithm in terms of achievable bitrates while the convergence time and the buffer are more efficiently tuned in the serial method.

Bradai *et al.* [27] proposed a content-aware approach for bandwidth allocation in Peer to Peer (P2P) layered video streaming systems that uses an auction game model. The objective is to first guarantee a minimum quality level to the neighbors peers, and then, allocate the left over bandwidth for the higher layers. A different approach is proposed by De Cicco *et al.* in [18] by employing the feedback control theory to design a Quality Adaptation Controller (QAC) for a stream-switching live adaptive video streaming system. The controller is located at the server side to avoid delays in the control loop and reduce the workload on the client by just decoding and playing the video segments. Similarly to [18], Tian and Liu proposed in [28] a closed-loop feedback control to adjust the video bitrate but at the client-side rather than at the server-side. Lee *et al.* [29] proposed a multimedia content adaptation framework in wireless networks. This approach

allows a systematic multimedia content conversion that takes into account the user preferences, device capabilities and network conditions.

Huang *et al.* [30], [31] proposed a buffer-based approach for video rate adaptation. This approach uses only the buffer occupancy to pick the highest possible video rate without unnecessary re-buffering and does not require any estimation of the available bandwidth. Jiang *et al.* [1] studied the problem of bitrate selection with respect to the three conflicting metrics: fairness, efficiency and stability. They proposed a client-based adaptation algorithm, called FESTIVE, composed of three main modules. The first module estimates the available throughput using the Harmonic Mean over the last 20 observed values. The second module is responsible for the bitrate selection and implements a gradual switching strategy that ensures a convergence of the players to a fair allocation irrespective of their current bitrates. To avoid frequent bitrate switching, the authors introduce the notion of *delayed update* which consists of calculating a trade-off between efficiency and fairness, on one hand, and stability, on the other hand, for both current and reference bitrates. It then picks the lowest one. The *randomized scheduler* is the last module. It ensures there is no start time biases by scheduling the next chunk randomly within a randomized target buffer size if the buffer playback is greater than or equal to a target buffer size. Otherwise, the next chunk is immediately downloaded.

Li *et al.* [2] first identified the root causes of the current solutions' oscillations. They concluded that the causes are the discrete nature of available video bitrates resulting in under-utilization of the network bandwidth, and the client on-off periods resulting in a biased perception of the client to its throughput. To deal with these problems, they proposed a client-side algorithm, called PANDA, based on Probe-AND-Adapt approach that consists of incrementing the data rate during the off-intervals to efficiently utilize the bandwidth. Zhao *et al.* [32] design a set of video streaming QoE key factors and propose a bandwidth-based dynamic average QoE adaptive algorithm that tries to maximize the average QoE using a moving average approach. A server-based approach is proposed by Akhshabi *et al.* [19] to address the instability issue when multiple players share a bottleneck link. The proposed shaping mechanism aims at eliminating the root cause of instability which is the OFF behavior of the players. It also helps in preventing frequent oscillations between the different quality levels.

Many recent studies [33]–[37] showed that when using traditional HTTP for video delivery, high numbers of HTTP requests, each corresponding to a segment, generate additional overhead. All these studies rely on the newly published HTTP/2 protocol in order to reduce the latency in HTTP communications. One of the main introduced features in HTTP/2 standard is the server push whereby a server pushes multiple content to the client without making individual requests for each content [33]. Wei *et al.* [35] discuss the impact of the requests overhead in HAS on the power consumption for mobile devices, and propose a power aware *K*-push strategy for HAS in order to achieve a lower power consumption. Van der Hooft *et al.* [37] propose a server push strategy wherein a server actively pushes super-short segments to mobile client in HAS live streaming scheme. Using this approach, the startup time as well as the end-to-end delay are considerably reduced, compared to HAS over HTTP/1.1.

III. Problem Formulation and Network Model

In this section, we formulate the general client/server behavior of a typical DASH-based solution. Next, we describe the three main issues we want to tackle in this paper. After that, we discuss our approach to resolve the issues. Finally, a description of the network environment to which the proposed solution will be deployed is provided.

A. Problem Formulation

The DASH technology aims at provisioning a high QoE to the users by dynamically adapting the bitrate of the played video to the clients' conditions (e.g., bandwidth fluctuations and device capabilities) [38]. This can be achieved by storing multiple versions of the same video at the server side where each video is encoded into a set of m discrete representation levels with different encoding rates $L = \{l_0, l_1, \ldots, l_{m-1}\}$, with $l_0 < l_1 < \cdots < l_{m-1}$. Each representation level l_j, $0 \leq j < m$, is chopped into n small segments of fixed τ seconds called chunks. We denote by $\zeta_{i,j}$ the i^{th} chunk from the j^{th} representation level. The server is connected to the Internet with a W bandwidth link which gives an upper-bound value on how fast the server's connection can possibly transmit data [39]. The server's link capacity, also called bottleneck link, is shared among $\mathcal{N}$ competing clients where each client k has its own end-to-end estimated link capacity T_i after downloading the i^{th} chunk. The downloaded chunks are buffered into a space memory of β_{max} seconds maximum. The buffer length is measured in seconds since the buffered chunks might be from different representation levels (i.e., with different sizes). The rate of consumption from the buffer is constant with one second of the buffered video time each real-time second. Based on both client's estimated bandwidth and buffer state, the client selects the adequate representation level j to request. When the buffer goes empty, the end user will experience video playback freezes which is the main problem that should be avoided as much as possible even if the video bitrate is decreased. For this aim, it might be relevant to define a minimum buffer threshold β_{min} which serves as an alarm for the client to decrease the played representation level once attempted. On the other hand, if the buffer is full, the download process goes idle (OFF period) for a calculated θ period until enough space is available for at least one chunk. The OFF periods cause alternations in the observed throughput which ultimately impact the perceived QoE. Table I lists and defines the symbols and notations used in this paper.

B. Addressed Issues

Despite its numerous advantages, the DASH scheme still cannot provide an optimal solution when considering the three following objectives: *Efficiency, Stability and Fairness*. A client may be efficiently receiving a video stream when the stream comes from the highest possible representation

TABLE I
LIST OF SYMBOLS AND NOTATIONS USED IN THE PAPER

Notation	Definition
β_{min}/β_{max}	Minimum/Maximum buffer threshold (in seconds)
β	Current buffer (in seconds)
τ	Chunk Duration (in seconds)
θ	Idle Duration (in seconds)
$\zeta_{i,j}$	The i^{th} chunk from the j^{th} representation level
μ_j	The expected fetch time (in seconds) of a chunk from the j^{th} representation level
λ_j	The bitrate of the j^{th} representation level
udf(j)/*ovf*(j)	Boolean function that tests the risk of underflow/overflow with the j^{th} representation level
T_i	Observed throughput at the i^{th} chunk
$\widehat{T_i}$	Smoothed throughput at the i^{th} chunk using Harmonic Mean over the last twenty T_i values
W	Total Shared Bandwidth Capacity
t_{br_i}	Time before requesting chunk i
t_{ar_i}	Time after fully reception of chunk i
ω_{avg}	Average Bandwidth
ω_{rmn}	Remaining Bandwidth
ω_{cmt}	Cumulated used Bandwidth
$\mathcal{N}$	Number of Connected Clients
Ω	Set of clients requesting bitrate greater than the average bandwidth
δ	The minimum number of chunks to be requested from each representation level
f_t	The system fairness metric at time t
F	The overall system fairness metric
$r_{k,t}$	The allocated bitrate to the k^{th} client at time t
e_t	The system efficiency metric at time t
E	The overall system efficiency metric
b_k	The buffer metric of the k^{th} client
B	The overall system buffer metric
s_k	The stability metric of the k^{th} client
S	The overall system stability metric

level which is lower than his end-to-end estimated bandwidth. However, due to throughput's fluctuations, this can cause frequent switches between the representation levels, ultimately degrading the user's QoE [40]. Another issue is to ensure fairness among competing clients for the same bottleneck link since the clients may have different link capacities. A fair system should give to the clients having equivalent bandwidth capacities representation levels close to each other. It should equally prohibit clients connecting first from dominating the server's shared bandwidth.

C. Problem Resolution Approach

In the previous section, we mentioned that the three objectives are in conflict with each other. This conflict can be resolved only through a trade-off. For this purpose, we envision using separate, ordered controllers for maintaining every objective whereby the output decision of each controller forms the input of the next controller. The order of these controllers define a kind of priority between the three objectives, whereby the first in the sequence has the lowest priority and the last has the highest priority since the final decision on the representation level is taken by the latter. In our work, the three controllers are ordered as follows: Efficiency, Stability and then Fairness controller. The idea behind this sequence is that the proposed system should ensure first for each client its *fair share* and then the *highest stable* representation level. In addition, each objective is delegated to either the client or the server based on the available information at each side. At the client side, the buffer state and the estimated throughput are available. This makes the client in a better position to control the *Efficiency*. Moreover, the information about the previously downloaded chunks and their corresponding bitrates are known. Therefore, the *Stability* can be efficiently managed by the client too. The issue of fairly sharing the server's bandwidth among competing clients can be resolved if the following information are available: the amount *W* of the bandwidth resource to share, the number of competing clients for the shared bandwidth, and their corresponding estimated bandwidth [20]. For the last information, it can be communicated to the Fairness controller with every client's request while the two others are already known by the server which makes the server in better position to control *Fairness*. The way the server can share its bottleneck link among competing clients is by allocating the right representation level based on their estimated bandwidth.

D. Network Environment and Scope

The proposed DASH-based solution can be deployed in a typical Internet-based network environment whereby clients have different link capacities and perform simple HTTP GET requests to a multimedia server to receive video steams of desired content. Our proposed solution consists of two algorithms. The first runs at the client and operates as a player. The second is implemented at the server.

The server is a dedicated HTTP video streaming server.[1] connected to the Internet through a link (bottleneck link) with a given *W* bandwidth capacity shared among competing clients.

[1]Whilst we study ESTC considering the case of one streaming server, ESTC can be used in case of multiple servers with one orchestration server (e.g., based on open source tools such as ActiveMQ or RabbitMQ) that stores the different clients' capacities and allocates the clients to the servers based on the servers' load.

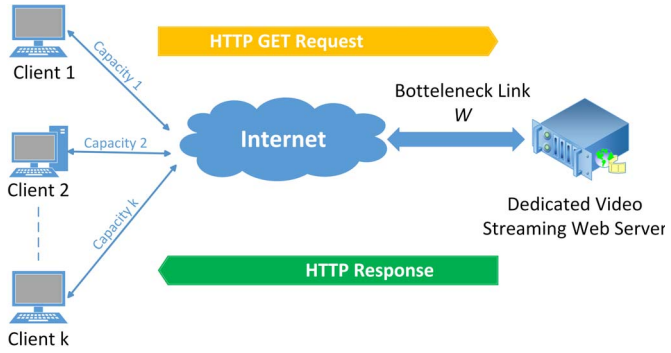


Fig. 1. Envisioned network environment.

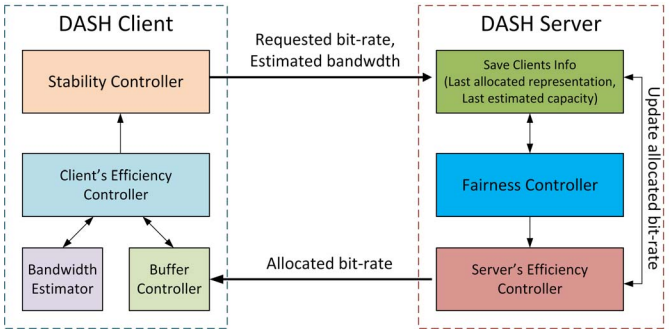


Fig. 2. Global architecture of the proposed ESTC scheme.

For the sake of simplicity and without any loss of generality, we assume that the bottleneck link capacity is fully dedicated for video streams and is not used by other traffic types. Another assumption is that the server's shared bandwidth is not saturated by the aggregate bitrates of the serviced clients [1]. For this purpose, we ensure that the link does not operate at its full capacity, but rather at a capacity of $(W - C)$. From the content perspectives, we assume that the server streams Video-on-Demand (VoD) content, pre-encoded and segmented into τ chunks. Fig. 1 illustrates the envisioned network model.

IV. Proposed Approach

As elaborated in Section II, the existing solutions and protocols deal with one or more of the three conflicting objectives (i.e., efficiency, stability and fairness) from either client or server side. The main objective of our approach is to ensure fairness among clients taking into account the shared bandwidth W and the client's capacity $\widehat{T}_i$ at each chunk request while still allocating the highest stable representation level for each client. However, an accurate decision cannot be achieved without taking into account the conditions of both client and server sides. At the client side, the information about the available throughput, buffer occupancy and the history of downloaded chunks and their corresponding bitrates are known. Thus, the client has all required information to take the right decision that maximizes the video quality while ensuring its stability. At the other side, the server disposes of the shared link capacity, the total number of connected clients and their current downloading bitrates. Therefore, the fairness objective can be efficiently tuned at the server side than at the client side. In addition, if the server gets information about the current client capacity, this can considerably improve the stability. Hence, an efficient and robust video streaming solution should involve both client and server in the process of bitrate adaptation and should leverage the crucial information available at both sides to achieve the three objectives, namely efficiency, stability and fairness.

In this section, we describe our proposed bitrate adaptation algorithm. The global architecture of the proposed solution is depicted in Fig. 2. It consists of client and server controllers. The client controller chooses the bitrate of the next chunk based on the observed throughput and buffer state; this would control the efficiency objective. Moreover, the client controls the video stream stability by minimizing the switches between the different levels. After taking the decision, the client sends a HTTP GET request to the server with the selected representation level and its current bandwidth capacity. Based on the information available at the server side and the information received from the different clients, the server shares the W bottleneck link fairly between the connected clients. The following sub-sections give details on the functionalities of the client and server controllers.

A. Client-Side Controllers

The roles of the client's controllers consists of maximizing the perceived video quality, minimizing the oscillations and ensuring smoothness in the video playback upon changes in the streaming bitrate, while maintaining the buffer above a predefined threshold. The output of the client controllers is the final decision on the next chunk's bitrate. This decision, along with the client's estimated bandwidth capacity, is communicated to the server. In the following, we portray the different client-side controllers.

1) Bandwidth Estimator: The bandwidth estimator module computes the download time of the chunk to be requested. After the complete reception of the chunk, it estimates the end-to-end bandwidth by dividing the bitrate of the chunk by the download time according to Equation (1). However, the observed bandwidth is not always stable and may fluctuate during different time intervals [41] and that is due to several factors such as traffic congestion and newly arriving/departing clients. To cope with this issue, we use the harmonic mean of the last 20 chunks to smooth the observed bandwidth [1]. This value may result in a slow convergence to the fair share but provides more reliable values to pick the adequate next bit rate. A too small value may lead to instability. Similarly, a too large value may delay the convergence time to the fair share since it may comprise more bandwidth outliers' values. The choice of the harmonic mean against the other smoothing methods, e.g., the moving average, is justified by the fact that the harmonic mean is more suitable when working with rates and having outliers values - since it gives greater importance to small values. Note that, we start the streaming session by requesting the first chunk from the lowest available representation level, and we do not employ any smoothing method during the first 20 chunks. This allows a quick representation level improvement for both short and long videos without badly affecting

the overall performance. In our simulations, we have tested the use of the harmonic mean over the previously initial estimated values of the bandwidth and noticed that it delayed the improvements of the representation levels, compared to when directly using the estimated bandwidth.

$$T_i = \frac{\lambda_j \cdot \tau}{t_{ar_i} - t_{br_i}} \tag{1}$$

where τ denotes the chunk duration. t_{ar_i} and t_{br_i} denote the time after and before requesting the i^{th} chunk, respectively.

2) Buffer Controller: One of the most important components in the DASH system is the buffer. Its main role consists of absorbing the mismatch between the video bitrate and the available throughput. Additionally, it forms a reserve of chunks in case of a sudden drop in the throughput. Another benefit of the buffer is to ensure stable playback of the video stream as the representation level changes only if there is a risk of underflow occurrence. However, a large buffer may result in waste of resources (i.e., in case clients decide to leave or switch to another video) and inefficiency (i.e., when the available throughput improves). To efficiently manage the buffer, a minimum (β_{min}) and a maximum (β_{max}) buffer thresholds are defined. Whilst β_{max} allows switching the selection process to idle status, β_{min} serves as an alarm for the client buffer controller. Once β_{min} is reached, the client should decrease the current downloading bitrate by selecting the highest possible bitrate which is lower than the observed throughput. Based on many empirical results and also results using Microsoft smooth streaming [10], β_{min} and β_{max} are set to 14 and 30 seconds, respectively. It shall be noted that for the sake of simplicity, the buffer occupancy is measured in seconds since the buffer can contain chunks of different bitrates.

3) Efficiency Controller: The goal of the client's efficiency controller is to select the highest possible video quality taking into account the buffer state and the available bandwidth to avoid the buffering phenomenon. Its role consists of taking a decision on either increasing, decreasing, maintaining the current level, or idling the process of requesting chunks for a given calculated period according to Equation (2):

$$j = \begin{cases} Q(.), & \text{if } udf(j) = true; \\ j+1, & \text{if } (udf(j+1) \text{ and } ovf(j+1)) = false; \\ IDLE(), & \text{if } ovf(j) = true; \\ j, & \text{Otherwise keep the current level} \end{cases} \tag{2}$$

where $Q(.)$ is a quantization function that returns the highest level index with the Expected Fetch Time (*EFT*) lower than $(\beta - \beta_{min})$, which means that during this period the buffer occupancy remains above the minimum buffer threshold. The boolean functions *udf*(j) and *ovf*(j) test for the buffer underflow and overflow, respectively. The representation level is taken as an input and the return value is either *true* or *false* if an underflow/overflow may occur according to Equations (3) and (4), respectively:

$$udf(j) = \begin{cases} true & \text{if } \mu_j \geq (\beta - \beta_{min}) \\ false & \text{otherwise} \end{cases} \tag{3}$$

$$ovf(j) = \begin{cases} true & \text{if } \tau \leq \left(\beta_{max} - \left(\beta - \mu_j\right)\right) \\ false & \text{otherwise} \end{cases} \tag{4}$$

Algorithm 1: Client-Side Controllers

```
i ← 0; j ← 0; c_count ← 0;
while i < n do
    t_br_i ← GETCURRENTTIME()
    REQUESTCHUNK(ζ_i,j)
    t_ar_i ← GETCURRENTTIME()
    T_i ← λ_j · τ / t_ar_i − t_br_i
    T̂_i ← CALCULATEHARMONICMEAN()
    begintest:
    if udf(j) then
        if j > 0 then
            j ← Q(.);          /* Decrease level */
            c_count ← 0;       /* Reset counter */
    else if (c_count ≥ δ) and (T̂_i > λ_j+1) and
    (udf(j + 1) = false) and (ovf(j + 1) = false) then
        if j < m − 1 then
            j ← j + 1;         /* Increase level */
            c_count ← 0;       /* Reset counter */
    else if ovf(j) then
        IDLE();                /* Idling */
        GOTO(begintest)
    end
    i ← i + 1; c_count ← c_count + 1;
end
```

The EFT value is given by Equation (5):

$$\mu_j = \frac{\lambda_j \cdot \tau}{\widehat{T}} \tag{5}$$

The above efficiency controller decisions are mainly based on the buffer occupancy and the observed throughput. The bitrate decreasing decision can be taken only if the EFT value of the next chunk from the current level may lead to a buffer value lower than β_{min}. In this case, the quantization function $Q(.)$ is called to maintain the buffer state above the minimum buffer threshold. With this strategy, a large buffer size helps the client to overcome the short-term bandwidth fluctuations and be more *stable*. Switching to the next high level is enforced only if there is no risk of underflow with the current level or risk of overflow when the next high level is selected. Additionally, the smoothed throughput should be higher than the next level bitrate. The next test of the controller checks for the risk of overflow with the current representation level. In this case, the controller goes idle for a duration, given by Equation (6) and long enough to ensure space in the buffer for the chunk to be requested, and then re-evaluates the different tests. Otherwise, the controller keeps the current level j

$$\theta = \tau - \mu_j. \tag{6}$$

4) Stability Controller: If chunks are picked just based on the estimated throughput, frequent oscillations between the different bitrates may be experienced, which may ultimately result in QoE degradation [9], [13]. This feature also affects negatively both the stability and smoothness properties of the

video stream. To cope with these issues, the decision of the efficiency controller should be first validated by the stability controller to minimize the occurrence of oscillations. This can be achieved by requesting at least δ chunks from the same representation level before each level increase even if higher throughput is observed. Moreover, to ensure the smoothness of the video stream when changing the video quality, the client switches to the next higher/lower level only when the buffer occupancy is above the minimum threshold β_{min}. The pseudo algorithm of the client controllers is shown in Algorithm 1.

B. Server-Side Controllers

At the server side, the shared bandwidth W as well as the number of connected clients $\mathcal{N}$, their current downloading bitrate and last smoothed throughput are known (i.e., since each client sends these information to the server in the HTTP GET requests). This places the server in a better position to efficiently control and fairly distribute the shared bandwidth among the connecting clients. To this end, the server uses the information received from the different clients (e.g., selected bitrate, smoothed throughput) to make a fair decision on which bitrates should be allocated to the clients. Furthermore, it exploits the remaining throughput of the clients who do not fully use their allocated bandwidth to enhance the overall system efficiency. The benefit of this strategy is to maximize the bandwidth utilization and, at the same time, it allows a quick and smooth convergence of the different clients, depending on their bandwidth capacity, to the fair share. It also ensures stable video streaming to the different clients by prohibiting others from exploiting the remaining bandwidth randomly due to OFF periods. Note that the server can supply a client with either the requested or lower bitrate, and does not allocate higher than the requested bitrate. In the following sub-sections, more details are provided on how the server ensures fairness among competing clients and maximizes the bandwidth utilization.

1) Fairness Controller: In each request made by a client for a video chunk, the server receives the client's last smoothed throughput. This information is stored at the server to be used by both fairness and efficiency controllers. It shall be noted that information on the smoothed throughput of clients is required as clients can request lower bitrates to avoid violating the smoothness or stability properties. If the requested bitrate is higher than the average bandwidth, we believe that a fair system should give to each client the highest bitrate that is lower than the average bandwidth $W/\mathcal{N}$ (i.e., after subtracting the amount C to avoid saturating the bottleneck link as discussed before). Otherwise, it allocates the requested bitrate.

2) Server's Efficiency Controller: Some clients may request bitrates smaller than the average bandwidth due to their limited bandwidth capacity. The system should then take advantage of the difference between the average bandwidth and the requested one to improve the bitrates of the clients having requested higher bitrate than the average bandwidth. To this end, the accumulated unused bandwidth ω_{cmt} is calculated and the set of clients Ω, having smoothed throughput $\widehat{T}_i$ higher than ω_{avg}, are identified to be used by the server's efficiency controller. Here, it is worth noting that during the calculation of the unused bandwidth, the server uses the clients' smoothed throughput instead of the requested bitrates. This should guarantee for each client its reserved bandwidth and prohibit the clients with high bandwidth usage or connecting first to the server from dominating the shared bandwidth. The pseudo algorithm of the server controllers is shown in Algorithm 2.

Algorithm 2: Server-Side Controllers

Inputs: cid: Requesting Client ID,
$\zeta_{i,j}$: Requested chunk,
$\widehat{T}_i$: Smoothed throughput

$s\_count \leftarrow 0$; $\Omega = \emptyset$; $W \leftarrow W - C$;
$\omega_{avg} \leftarrow W/\mathcal{N}$; $\omega_{rmn} \leftarrow 0$; $\omega_{cmt} \leftarrow 0$;
$ReqLevels[cid] \leftarrow j$; $Capacity[cid] \leftarrow \widehat{T}_i$;
if $(\lambda_j > \omega_{avg})$ **then**
  for $id \in Capacity$ **do**
    if $(Capacity[id] \leq \omega_{avg})$ **then**
      $\omega_{cmt} \leftarrow \omega_{cmt} + Capacity[id]$
    else
      $s\_count \leftarrow s\_count + 1$
      $\Omega = \Omega \cup \{id\}$;
    end
  end
  while $(\lambda_j > \omega_{avg})$ *and* $(s\_count > 0)$
  and $(cid \in \Omega)$ **do**
    $Sort(\Omega)$;
    $\omega_{rmn} \leftarrow W - \omega_{cmt}$
    $\omega_{avg} \leftarrow \omega_{rmn}/s\_count$
    $s\_count \leftarrow 0$
    for $id \in \Omega$ **do**
      if $(Capacity[id] \leq \omega_{avg})$ **then**
        $\omega_{cmt} \leftarrow \omega_{cmt} + Capacity[id]$
        $\Omega = \Omega \setminus \{id\}$
      else
        $s\_count \leftarrow s\_count + 1$
      end
    end
  end
  if $(\lambda_j > \omega_{avg})$ **then**
    $ReqLevels[cid] \leftarrow \mathrm{Q}(\omega_{avg})$
  else
    $ReqLevels[cid] \leftarrow j$
  end
end

V. Evaluation Metrics

In this section, we fine the different metrics to use in the performance evaluation of our proposed solution. As comparison terms, we use the FESTIVE and PANDA algorithms, implemented using the configurations described in [1] and [2], respectively. The evaluation is based on the following four metrics: buffer occupancy, the overall bandwidth utilization, the fairness when different clients compete for a bottleneck link and finally the stability of each client.

A. Buffer Occupancy

The first objective of any video bitrate adaptation solution is to avoid as much as possible the main cause of users' frustration which is the buffering phenomenon. So, it is important to measure the buffer occupancy to see if it is violated or not, and for how many times. The buffering metric of the client k (b_k) is defined by the number of seconds when the buffer is empty divided by the total number of video seconds. The closer this value is to null, the better the system is. The buffering metric is related to the client and it is therefore calculated at the client side for the whole session. To get the overall system buffering (B), the mean of the different clients' buffering values is calculated using Equation (7)

$$B = \frac{\sum_{k=1}^{\mathcal{N}} b_k}{\mathcal{N}}. \tag{7}$$

B. Bandwidth Utilization

The bandwidth utilization (i.e., Efficiency) metric is calculated at the server side at each real second for the different connected clients. It indicates how efficiently the shared bandwidth is used at time t using Equation (8), where the cumulative bitrates of all clients is divided by the shared bandwidth capacity. The system is more efficient when the value is close to one and vice versa. The mean of the different calculated e_t values measures the overall system efficiency metric (E)

$$e_t = \frac{\sum_{k=1}^{\mathcal{N}} r_{k,t}}{W}. \tag{8}$$

C. Fairness

In order to measure how the different solutions are fair with all connected clients competing for a bottleneck link, the Jain's Fairness Index is used: (JFI) $= \frac{(\sum x_i)^2}{(n\sum x_i^2)}$ [42]. This index provides a bounded (between 0 and 1) and continuous measurement. However, JFI can be adopted only when the different clients have the same bandwidth capacity or higher than the average bandwidth. Otherwise, this index returns inaccurate measurements. This can be explained by the following example. Let's assume the case whereby two clients compete for a bottleneck of 4 Mbps. The first client has a maximum bandwidth of 1 Mbps only, while the second one has 3 Mbps. If each client gets a bitrate equals to its maximum bandwidth, which is the fair share, in this case the system should attempt the higher fair allocation measure which is the value 1. But the use of JFI gets a fairness measure value of 0.8 while the system is 100% fair. To deal with this issue, a normalization must be employed that takes into account the bandwidth capacity of each client. The normalization process depends on whether the client's bandwidth capacity is greater than the average bandwidth or not. For the first category that contains clients having a bandwidth capacity lower than or equal to the average bandwidth, we divide the client's current bitrate by their bandwidth capacity. For the others, we divide by the average bandwidth after accumulating the left over bandwidth of the different clients from the first category. Then, JFI is applied on the normalized values to measure the fairness. If we apply this logic to the above example, we get a normalized value equal to 1 for the first client by dividing 1 by 1. After that, the remaining bandwidth from the first client (1 Mbps) is added to the average bandwidth (2 Mbps) giving a total of 3 Mbps. Similarly, the current bitrate of Client 2 is divided by 3 Mbps resulting in a normalized value equal to 1. Therefore, the JFI measure returns 1 which means that the system is 100% fair with the clients. Using this approach, the fairness metric (f_t) is calculated at the server side at each second, where the mean of the different calculated f_t values measures the overall fairness metric (F) of the whole system.

D. Stability

The instability metric for a client k ($instability_k$) is defined within a range of requested chunks by the number of the bitrate drops divided by the number of seconds in that range. Moreover, if the client decreases the video quality by more than one level, the counter is increased by the number of levels between the current and new representation levels. Unlike the instability metric proposed in [1] that considers all switch steps observed within the last 20 seconds, in our proposed metric, we only consider the bitrate decrease since selecting a higher bitrate is the desired behavior that will not negatively affect the client stability. To get the *stability* metric s_k for a client k, $s_k = 1 - instability_k$ is used. Since the stability metric is proper to each client, it is calculated at the client side. Then, the average of all clients' stability values is calculated to measure the overall system stability using Equation (9)

$$S = \frac{\sum_{k=1}^{\mathcal{N}} s_k}{\mathcal{N}}. \tag{9}$$

VI. Simulation Results

To evaluate the performance of the ESTC scheme, several simulations have been conducted using the NS-3 simulator [43]. The evaluation and comparison to FESTIVE and PANDA is based on the above-mentioned metrics.

A. System Configuration

In this paper, a star topology, composed of a server and four clients connecting to a central router, is used to conduct various simulations. The communication between the server and the clients is established with TCP/IP protocol. The used network topology is depicted in Fig. 3. The clients request a video sequence of 10 minutes length, pre-encoded into the following 12 representation levels: 30, 100, 400, 800, 1200, 1800, 2200, 3000, 5000, 7000, 9000, 11000 Kbps. The server's shared bandwidth is set to $W = 10Mbps$ and the amount of bandwidth to subtract to avoid the saturation of the bottleneck link is set to $C = 400Kbps$. The length of a chunk is set to $\tau = 2s$, while the minimum number of chunks to be requested from each representation level is set to $\delta = 5$ chunks. β_{min} and β_{max} are set to 7 and 15 chunks, respectively.

B. Evaluation

In the performance evaluation of ESTC against both FESTIVE and PANDA, we consider a realistic scenario

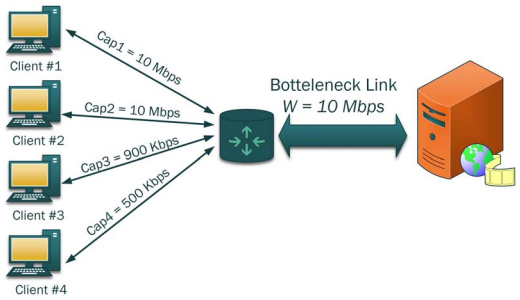


Fig. 3. A star topology used in the simulation environment.

TABLE II
THE LINK CAPACITIES AND ARRIVAL TIMES OF THE FOUR SIMULATED CLIENTS

	Link capacity	Arriving time
Client 1	10 Mbps	t = 0 s
Client 2	10 Mbps	t = 300 s
Client 3	900 Kbps	t = 100 s
Client 4	500 Kbps	t = 150 s

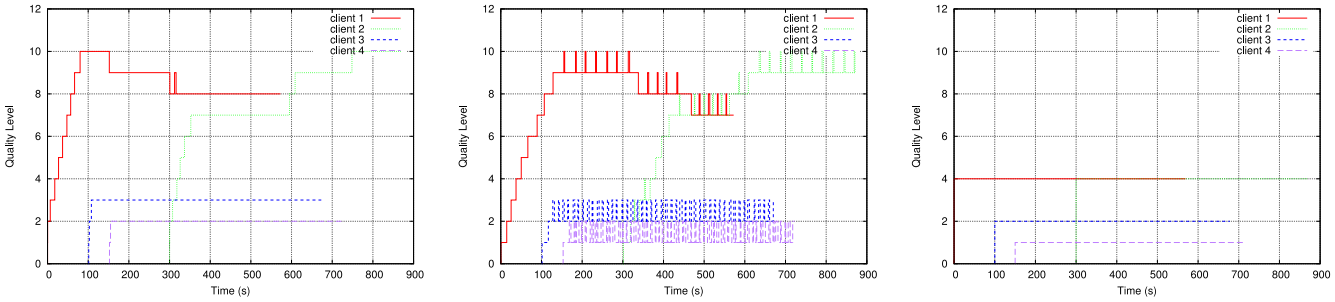
whereby four clients, having different bandwidth capacities, connect to a streaming server at different times. Table II shows the link capacity and arrival times of the simulated clients. For the sake of illustration and ease of discussion, we first provide the results of four clients to show the behaviour of the different solutions and how each of them converges to the fair share, which is not possible when running a high number of clients. Other simulations were conducted using a larger number of clients (20, 50 and 100 clients) and the results are discussed in Section VI-B6. We have noticed that the fundamental observations we made about the outperformance of ESTC in comparison to FESTIVE and PANDA remain the same, regardless of the clients' number.

1) Efficiency: The played representation levels of the different clients are plotted in Figs. 4(a), 4(b) and 4(c) for the ESTC, FESTIVE and PANDA schemes, respectively. As shown in Fig. 4(a), Client 1 connects alone to the server at $t = 0$ and benefits from the whole bandwidth to reach l_{10} at time $t = 80s$, which is the highest level lower than its maximum bandwidth capacity. At $t = 100s$, Client 3 joins and tries to improve its acquired representation level until achieving the highest level (l_3) it can visualize. This does not affect Client 1 as the shared bandwidth is sufficient to satisfy the requests of both clients and keep them stable. At $t = 150s$, Client 4 connects and causes a decrease in the representation level of Client 1 only. However, Client 3 is not affected as its played level is lower than the average bandwidth. At $t = 300s$, Client 2 starts its session and increases its representation level gradually until reaching l_7 which is the highest level it can attempt (i.e., lower than the average bandwidth after exploiting the remaining bandwidth from the parts of Clients 2 and 3). It can be noticed that Clients 2 and 3 keep their current level and remain stable since they get a level lower than the average bandwidth. Furthermore, the representation level played by Client 1 is decreased only by one level allowing Client 2 to increase its level. This strategy permits at the same time maintaining the smoothness property and avoiding unnecessary level switches which makes the clients more stable.

In case of the FESTIVE algorithm, Fig. 4(b) shows that the different clients encounter a considerable delay in achieving the fair allocation: level l_{10} at $t = 155s$, level l_3 at $t = 129s$, level l_2 at $t = 168s$ and level l_8 at $t = 435s$ for Clients 1, 3, 4 and 2, respectively. Moreover, the clients cannot maintain the fair allocation level for longer times as Client 1 (which is the first client) gets a level higher than its fair allocation. The main goal of the PANDA algorithm is to improve the clients' stability. This is clearly shown in Fig. 4(c) where the clients are 100% stable. However, we observe that the different clients got a representation level lower than their fair share, even if the observed TCP throughput allows higher bit rate. This is caused by the asymmetry rate level shifting feature that allows a *conservative* rate level upshift and more responsive downshift [2] due to the formula employed at the Estimate Bandwidth Share step. This formula aims at mitigating the impact of the observed throughput when it improves, making the clients more stable but less efficient in terms of bandwidth utilization.

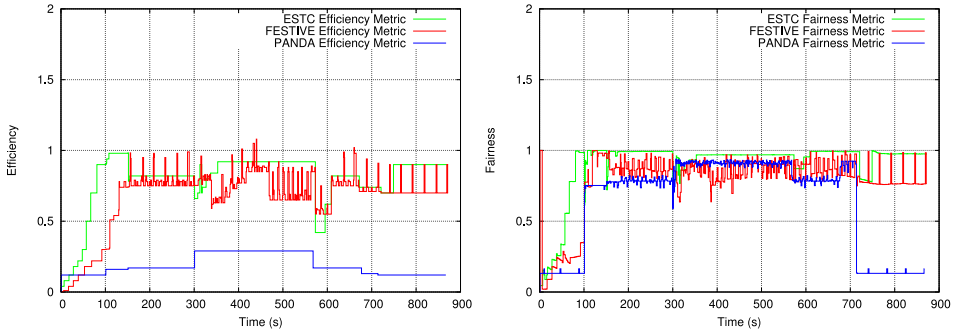
The different values of the efficiency metric for the three algorithms are plotted in Fig. 5(a). When Client 1 connects to the server, the maximum efficiency value reached was $E = 0.9$ at $t = 81s$, this is due to the discrete nature of the representation levels since the next level ($11Mbps$) is greater than the maximum bandwidth capacity of Client 1 and the move to that level will make it unstable. This efficiency value was improved to a value in the vicinity of 1 ($E = 0.98$) after the connection of Client 3 at $t = 100s$. The start of Client 4 at $t = 150s$ causes a drop in the efficiency value to $E = 0.82$. This is justified by the fact that the increase in the quality level of Client 4 causes a decrease in the played level of Client 1 but the decreased amount is greater by far than the increased bandwidth amount. Similarly, the start of Client 2, having the same bandwidth capacity as Client 1, causes another drop in the efficiency measure ($E = 0.66$) for the reason that the representation level of Client 1 falls to l_8 ($5Mbps$) with a difference of $2Mbps$ between l_8 and l_9. This value is gradually rising as the representation level of Client 2 increases until it reaches the value $E = 0.92$ at $t = 353s$. From this time, the efficiency remains stable until Client 1 leaves causing a considerable drop in its value ($E = 0.42$) since Client 1 occupied the largest amount of the shared bandwidth. This gives the opportunity to Client 2 to improve its played quality level to reach l_9 ($E = 0.82$) before Clients 3 and 4 leave, then l_{10} ($E = 0.9$) when they finish consuming all video chunks.

Compared to FESTIVE, ESTC exhibits better efficiency, while both ESTC and FESTIVE are by far more efficient than PANDA. Indeed, from Fig. 5(a), it can be observed that FESTIVE's efficiency metric exceeds occasionally the value of 1 which means that the accumulated bitrate of all clients is greater than the shared bandwidth. This maximizes the bandwidth utilization. However, it is dangerous since it leads either to buffering, when keeping higher levels for longer periods, or instability, when decreasing rapidly the video quality.



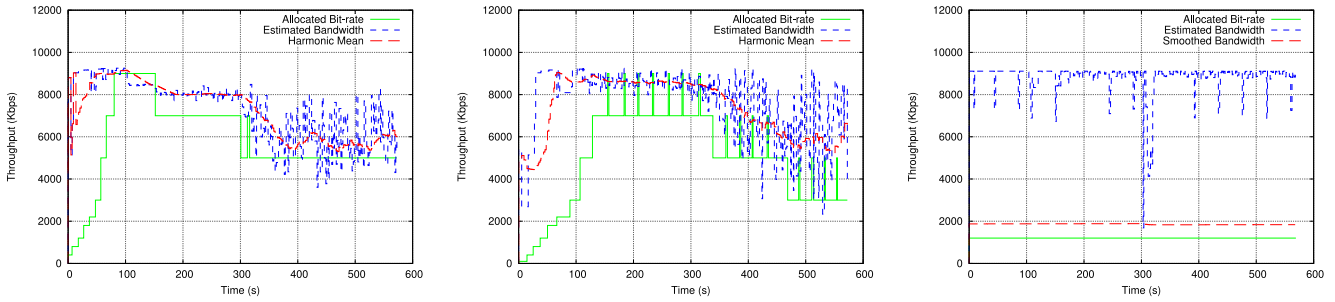
(a) ESTC. (b) FESTIVE. (c) PANDA.

Fig. 4. The played representation levels of the different clients.



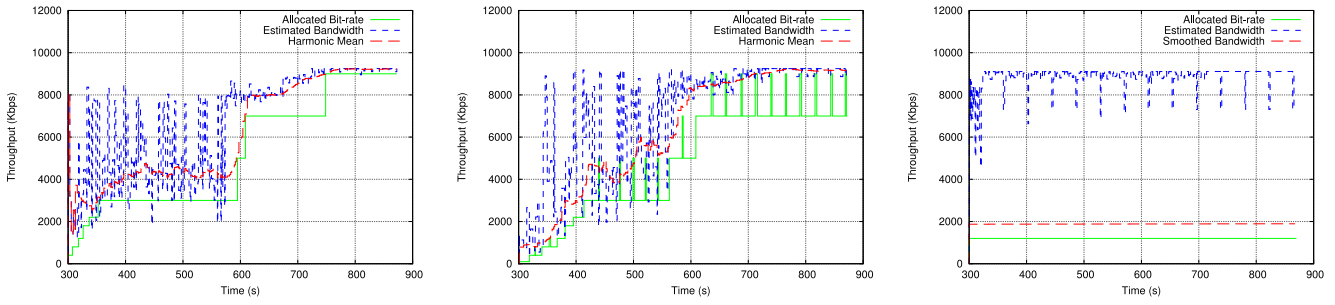
(a) Efficiency evaluation. (b) Fairness evaluation.

Fig. 5. The efficiency and fairness evaluation of ESTC, FESTIVE and PANDA.



(a) ESTC. (b) FESTIVE. (c) PANDA.

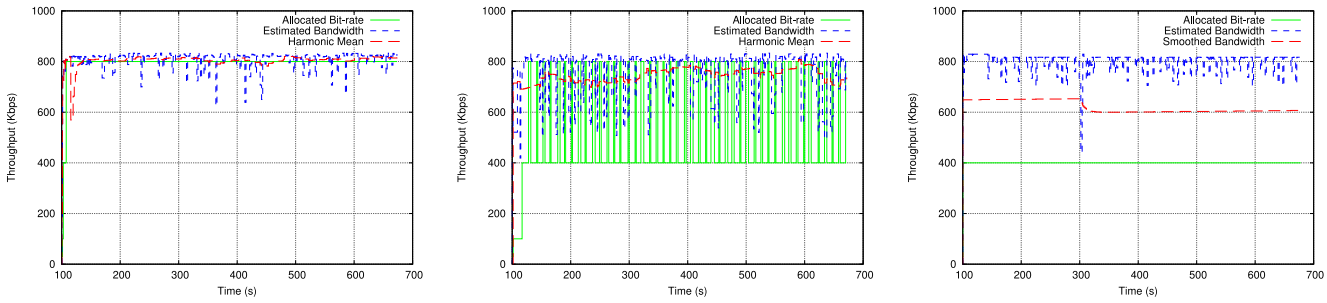
Fig. 6. The comparison of the allocated bitrates to Client 1 between ESTC, FESTIVE and PANDA.



(a) ESTC. (b) FESTIVE. (c) PANDA.

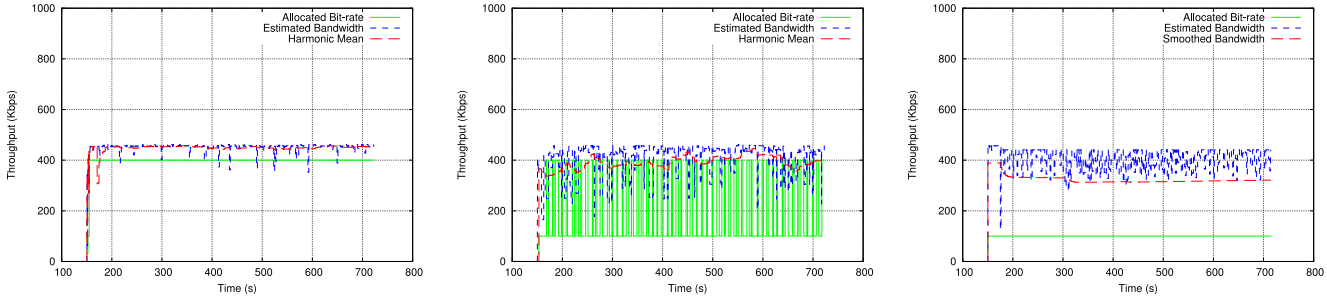
Fig. 7. The comparison of the allocated bitrates to Client 2 between ESTC, FESTIVE and PANDA.

The throughputs of the different four clients in ESTC are shown in Figs. 6(a), 7(a), 8(a), 9(a), while the clients' throughputs in case of FESTIVE and PANDA are illustrated in Figs. 6(b), 7(b), 8(b), 9(b) and Figs. 6(c), 7(c), 8(c), 9(c), respectively. In general, our results show that the clients select the highest bitrate, lower than the smoothed throughput and,



(a) ESTC. (b) FESTIVE. (c) PANDA.

Fig. 8. The comparison of the allocated bitrates to Client 3 between ESTC, FESTIVE and PANDA.



(a) ESTC. (b) FESTIVE. (c) PANDA.

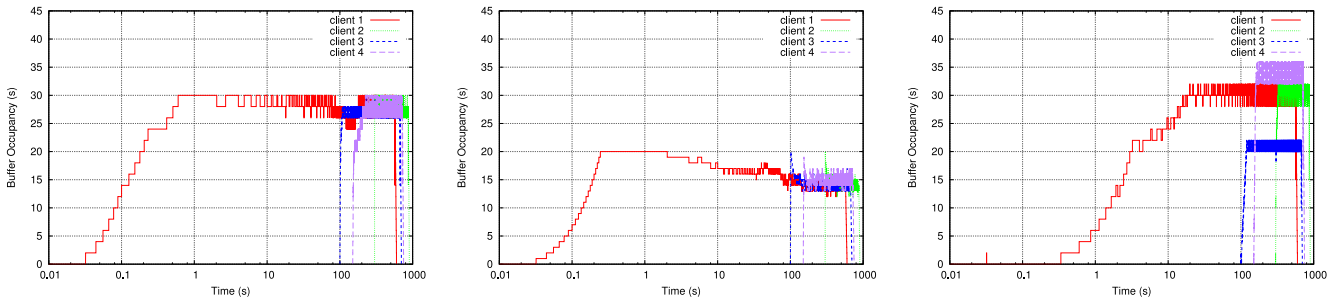
Fig. 9. The comparison of the allocated bitrates to client 4 between ESTC, FESTIVE and PANDA.

in rare cases, they select a bitrate higher than the smoothed throughput, such as $t \in [109, 151]$ for Client 1, as long as the buffer remains above the β_{min}. However, the frequency of selecting a bitrate higher than the smoothed throughput in case of the FESTIVE scheme is higher than in case of ESTC, which can result in better bandwidth utilization but makes clients less stable. In both solutions, Client 1 begins alone and benefits from the whole bandwidth. At time $t = 100s$, Client 3 joins but without affecting Client 1 since its bandwidth capacity is limited and the shared bandwidth support the demands of both Clients 1 and 3. At $t = 150s$, Client 2 connects but this time the shared bandwidth cannot support the three clients together without affecting one of them since the shared bandwidth is saturated by the two first clients. In this case, our proposed fairness controller prohibited Client 1 from dominating the whole shared bandwidth and decreases its representation level without affecting Client 3 which results in a fair bandwidth observation as we can see in the Harmonic Mean curve (blue lines). This also contributed to the stability of the video streams at the clients. In case of FESTIVE, as Client 1 connects first, it takes bandwidth higher than its fair share; a fact that ultimately impacts both fairness and stability of the overall system. As to PANDA, we can vividly see from Figs. 6(c), 7(c), 8(c), 9(c) that PANDA picks the highest representation level which is lower than the smoothed bandwidth. However, the formula used for the estimate bandwidth share is too conservative for upshift level, resulting in a lower value of the estimated bandwidth share and by consequent a lower selected bitrate compared to the fair share.

2) Fairness: Fig. 5(b) shows the variations of the fairness metric values calculated throughout the clients' sessions. It is clear from this figure that the proposed approach outperforms both the FESTIVE and PANDA algorithms in terms of fairness among competing clients. Indeed, the proposed approach achieves in most times high scores of fairness. The lowest values of the fairness during the sessions occur only when the clients connect or leave the server (i.e., there are five main lowest values at $t = 100s$, $t = 150s$ and $t = 300s$ corresponding to the arrival times of Clients 3, 4 and 2, respectively, and at $t = 572s$, $t = 722s$ corresponding to the leaving times of Clients 1 and 4, respectively). It can be noticed as well that the proposed system achieves a very high fairness values most of the times.

3) Stability: From Fig. 4(a), all clients are 100% stable except Client 1 which is slightly unstable ($s_1 = 0.005$). This results in an overall system stability value of $S = 0.99875$. The instability of the system is normal since it occurs only when a new client joins the server and does not badly affect the user experience except at $t = 312s$ when Client 1 switches to level l_9 and drops again to level l_8 after 3 seconds only. Fig. 4(b) shows that FESTIVE's clients are less stable compared to their ESTC counterparts. Indeed, the overall stability experienced in case of FESTIVE is $S = 0.944167$. This instability is caused by Client 1 which plays a bitrate (i.e., l_{10} before Client 2 joins and l_9 after the arrival of Client 2) higher than its fair allocation (i.e., l_9 and l_8, respectively). From Fig. 4(c) we can clearly see that PANDA's clients are 100% stable. Fig. 11 provides a comparison between the clients' stability metric in case of the ESTC, FESTIVE and PANDA schemes.

4) Buffer Occupancy: It is clear from Figs. 10(a), 10(b) and 10(c) that the three approaches ensure that there are always packets in the buffer to be displayed and that the buffer never



(a) ESTC. (b) FESTIVE. (c) PANDA.

Fig. 10. The buffer occupancy evaluation of the different clients.

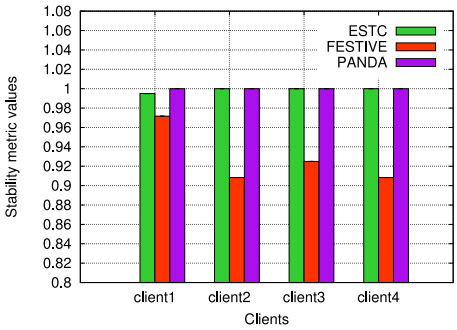


Fig. 11. The evaluation of the clients' stability in case of ESTC, FESTIVE and PANDA.

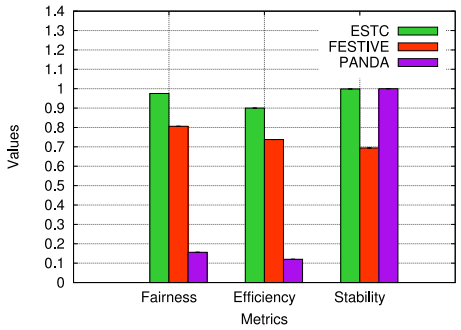
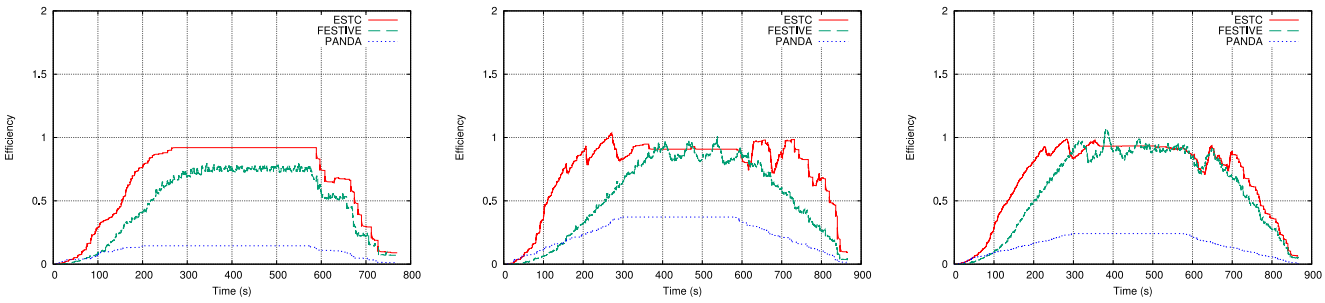


Fig. 12. Comparison of the overall stability, fairness and efficiency metrics between ESTC, FESTIVE and PANDA.

goes empty ($B = 0$). However, there is a difference in managing the buffer between the three solutions where FESTIVE maintains the buffer occupancy around a defined reference buffer , PANDA tries to keep the buffer around the minimum buffer threshold while ESTC keeps the buffer as full as possible (i.e., close to β_{max}).

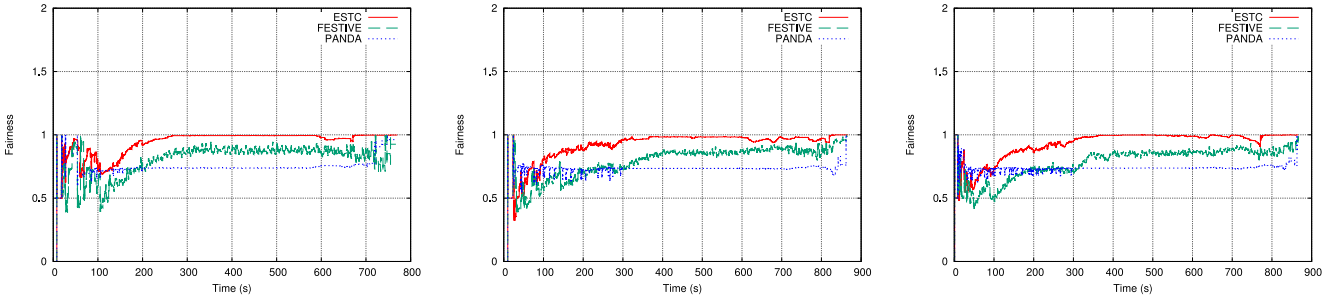
5) Convergence Time: Figs. 4(a) and 4(b) shows that new clients converge quickly to the fair share in case of the proposed solution than in case of FESTIVE. For example, Client 1 achieves l_{10} at $t = 80s$ while in FESTIVE it is achieved at $t = 156s$. This convergence is smooth regardless whether the bitrates are increased (e.g., for new connected clients playing lower bitrate) or decreased (e.g., for old connected clients playing higher bitrate). Furthermore, the quality level of the old connected clients is affected only if necessary (e.g., the arrival of Client 3 did not affect the quality level of Client 1). Contrary to the gradual rate level shifting strategy employed by both ESTC and FESTIVE, we see from 4(c) that PANDA uses aggressive representation level switching (e.g., Client 1 changes the level from l_0 to l_4, Client 2 goes from level l_0 to l_2), reaching quickly the target level but it might badly affect the user's QoE. Fig. 12, summarizes the overall values of the different metrics for the three solutions. It can be noticed that ESTC helps improving the stability by 43%, efficiency by 23% and the fairness by 21% and that is in comparison to FESTIVE. Furthermore, the proposed solution converges quicker and smoothly to the fair allocation compared to FESTIVE as it can be seen from figures 4(a) and 4(b).

6) Performance Results With Large Number of Clients: To further validate our proposed solution, we conducted the same simulations using larger number of clients having different bandwidth capacities and random starting times. Indeed, we conducted the simulations using 20, 50 and 100 clients and the relevant results are depicted in Figs. 13(a), 13(b) and 13(c) for the efficiency, and Figs. 14(a), 14(b) and 14(c) for the fairness metric. For a 20 clients scenario, the clients start randomly between 0 and 200 seconds and compete for 100Mbps of the shared bandwidth, while the starting time interval when running 50 clients is from 0 to 300 seconds. For a 100 clients scenario, the clients compete for 300Mbps of the shared bandwidth and start randomly within the time interval 0 and 300 seconds. For 20 clients, Fig. 13(a) shows that ESTC outperforms both FESTIVE and PANDA at all times, while being more stable (i.e., the red straight line after all clients connect at t = 280s). The ascending and descending portions of the curve correspond respectively to the connecting and leaving periods of different clients. We can also see that ESTC converges quicker to the fair share compared to FESTIVE and PANDA, and ESTC's clients are more stable than FESTIVE's clients (i.e., the straight red line for ESTC compared to the oscillating green line for FESTIVE). In Fig. 14(a), we can vividly see that ESTC is more fair (i.e., the fairness values at steady state are close to 1) in terms of giving different clients, with different bandwidth capacities, their fair share. As to the efficiency and fairness results with 50 and 100 clients depicted in Figs 13(b), 13(c), 14(b) and 14(c) respectively, we can see that ESTC outperforms FESTIVE in most of the time for the



(a) 20 CLIENTS. (b) 50 CLIENTS. (c) 100 CLIENTS.

Fig. 13. The efficiency comparison between ESTC, FESTIVE and PANDA when running larger number of clients.



(a) 20 CLIENTS. (b) 50 CLIENTS. (c) 100 CLIENTS.

Fig. 14. The fairness comparison between ESTC, FESTIVE and PANDA when running larger number of clients.

efficiency, while the ESTC's fairness is better than FESTIVE's at all times. Both ESTC and FESTIVE outperform PANDA in terms of efficiency and fairness. The overall observations do not change even if FESTIVE presents occasionally better performance in terms of efficiency.

VII. Conclusion

A successful HTTP adaptive video streaming solution should ensure the following objectives: high utilization of network resources, stability, fairness, and a short time of convergence to the fair share while avoiding buffering. In this vein, this paper proposed a DASH-based solution that achieves these objectives through close collaboration between clients and servers. Effectively, in the proposed solution, the control of each of the target objectives (i.e., efficiency, fairness and stability) is delegated to the appropriate side (client or server), depending on the available information at each side. At the client side, the available bandwidth, the buffer occupancy and the history of the played bitrates are known, which makes it the best candidate to control the bandwidth utilization, stability and managing buffer. Similarly, the fairness is well controlled at the server side since the number of connected clients with their current played bitrates and the bottleneck link capacity are known. The simulation results showed that the proposed approach improves the bandwidth utilization, fairness and stability while keeping the buffer as full as possible. Also, the obtained results showed that the clients converge quicker and smoothly to the fair share in case of both increasing and decreasing the bitrate. As future research work, the authors envisage implementing the proposed solution in a real-life environment, considering the case of multiple servers as part of a large scale content delivery network.

References

[1] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014, doi: 10.1109/TNET.2013.2291681.

[2] Z. Li *et al.*, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, Apr. 2014.

[3] T. Stockhammer, "Dynamic adaptive streaming over HTTP: Standards and design principles," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst. (MMSys)*, San Jose, CA, USA, 2011, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943572

[4] S. Chen, Z. Yuan, and G.-M. Muntean, "An energy-aware routing algorithm for quality-oriented wireless video delivery," *IEEE Trans. Broadcast.*, vol. 62, no. 1, pp. 55–68, Mar. 2016.

[5] M. Michalos, S. Kessanidis, and S. Nalmpantis, "Dynamic adaptive streaming over HTTP," *J. Eng. Sci. Technol. Rev.*, vol. 5, no. 2, pp. 30–34, 2012.

[6] C. Liu, I. Bouazizi, and M. Gabbouj, "Parallel adaptive HTTP media streaming," in *Proc. 20th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Maui, HI, USA, Jul. 2011, pp. 1–6.

[7] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst. (MMSys)*, San Jose, CA, USA, 2011, pp. 169–174. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943575

[8] C. Müller, S. Lederer, and C. Timmerer, "A proxy effect analysis and fair adaptation algorithm for multiple competing dynamic adaptive streaming over HTTP clients." in *Proc. VCIP*, San Diego, CA, USA, 2012, pp. 1–6.

[9] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware dash system," in *Proc. 3rd Multimedia Syst. Conf. (MMSys)*, Chapel Hill, NC, USA, 2012, pp. 11–22. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155558

[10] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Process. Image Commun.*, vol. 27, no. 4, pp. 271–287, 2012.

[11] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *Proc. 22nd Int. Workshop Netw. Oper. Syst. Support Digital Audio Video (NOSSDAV)*, Toronto, ON, Canada, 2012, pp. 9–14. [Online]. Available: http://doi.acm.org/10.1145/2229087.2229092

[12] A. Nadembega, A. Hafid, and T. Taleb, "An integrated predictive mobile-oriented bandwidth-reservation framework to support mobile multimedia streaming," *IEEE Trans. Wireless Commun.*, vol. 13, no. 12, pp. 6863–6875, Dec. 2014.

[13] T. Taleb, A. Hafid, and A. Nadembega, "Mobility-aware streaming rate recommendation system," in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, Kathmandu, Nepal, Dec. 2011, pp. 1–5.

[14] K. Kashibuchi, T. Taleb, A. Jamalipour, Y. Nemoto, and N. Kato, "Prioritization-based layered multicast for fixed/mobile networks with fast convergence and inter-session fairness," *J. Commun. Softw. Syst.*, vol. 2, no. 2, pp. 89–98, 2006.

[15] A. Hava, Y. Ghamri-Doudane, G. M. Muntean, and J. Murphy, "Increasing user perceived quality by selective load balancing of video traffic in wireless networks," *IEEE Trans. Broadcast.*, vol. 61, no. 2, pp. 238–250, Jun. 2015.

[16] C. Xu, Z. Li, J. Li, H. Zhang, and G.-M. Muntean, "Cross-layer fairness-driven concurrent multipath video delivery over heterogeneous wireless networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 7, pp. 1175–1189, Jul. 2015.

[17] A. Bokani, M. Hassan, and S. Kanhere, "HTTP-based adaptive streaming for mobile clients using Markov decision process," in *Proc. 20th Int. Packet Video Workshop*, San Jose, CA, USA, Dec. 2013, pp. 1–8.

[18] L. De Cicco, S. Mascolo, and V. Palmisano, "Feedback control for adaptive live video streaming," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst. (MMSys)*, San Jose, CA, USA, 2011, pp. 145–156. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943573

[19] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proc. 23rd ACM Workshop Netw. Oper. Syst. Support Digital Audio Video (NOSSDAV)*, Oslo, Norway, 2013, pp. 19–24. [Online]. Available: http://doi.acm.org/10.1145/2460782.2460786

[20] T. Taleb and K. Hashimoto, "MS2 : A new real-time multi-source mobile-streaming architecture," *IEEE Trans. Broadcast.*, vol. 57, no. 3, pp. 662–673, Sep. 2011.

[21] D. O. Mau, T. Taleb, and M. Chen, "MM3C: Multi-source mobile streaming in cache-enabled content-centric networks," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[22] E. Thomas *et al.*, "Applications and deployments of server and network assisted DASH (SAND)," in *Proc. IET Conf.*, Jan. 2016, p. 22. [Online]. Available: http://digital-library.theiet.org/content/conferences/10.1049/ibc.2016.0022

[23] *Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 5: Server and Network Assisted DASH (SAND)*, ISO/IEC Standard 23009-5:2017, 2017. [Online]. Available: https://www.iso.org/standard/69079.html

[24] D. Jarnikov and T. Özçelebi, "Client intelligence for adaptive streaming solutions," *Signal Process. Image Commun.*, vol. 26, no. 7, pp. 378–389, 2011.

[25] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over HTTP," in *Proc. 19th Int. Packet Video Workshop (PV)*, Munich, Germany, May 2012, pp. 173–178.

[26] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Process. Image Commun.*, vol. 27, no. 4, pp. 288–311, 2012.

[27] A. Bradai, T. Ahmed, R. Boutaba, and R. Ahmed, "Efficient content delivery scheme for layered video streaming in large-scale networks," *J. Netw. Comput. Appl.*, vol. 45, pp. 1–14, Oct. 2014.

[28] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proc. 8th Int. Conf. Emerg. Netw. Exper. Technol. (CoNEXT)*, Nice, France, 2012, pp. 109–120. [Online]. Available: http://doi.acm.org/10.1145/2413176.2413190

[29] S. Lee, S. Rho, and J. H. Park, "Multimedia contents adaptation by modality conversion with user preference in wireless network," *J. Netw. Comput. Appl.*, vol. 37, pp. 25–32, Jan. 2014.

[30] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. ACM Conf. SIGCOMM*, Chicago, IL, USA, 2014, pp. 187–198. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626296

[31] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for HTTP video streaming," in *Proc. ACM SIGCOMM Workshop Future Human-Centric Multimedia Netw. (FhMN)*, Hong Kong, 2013, pp. 9–14. [Online]. Available: http://doi.acm.org/10.1145/2491172.2491179

[32] S. Zhao, Z. Li, D. Medhi, P. Lai, and S. Liu, "Study of user QoE improvement for dynamic adaptive streaming over HTTP (MPEG-DASH)," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, Santa Clara, CA, USA, 2017, pp. 566–570.

[33] S. Wei and V. Swaminathan, "Cost effective video streaming using server push over HTTP 2.0," in *Proc. IEEE 16th Int. Workshop Multimedia Signal Process. (MMSP)*, Jakarta, Indonesia, 2014, pp. 1–5.

[34] G. Tian and Y. Liu, "On adaptive HTTP streaming to mobile devices," in *Proc. 20th Int. Packet Video Workshop (PV)*, San Jose, CA, USA, 2013, pp. 1–8.

[35] S. Wei, V. Swaminathan, and M. Xiao, "Power efficient mobile video streaming using HTTP/2 server push," in *Proc. IEEE 17th Int. Workshop Multimedia Signal Process. (MMSP)*, Xiamen, China, 2015, pp. 1–6.

[36] J. van der Hooft *et al.*, "HTTP/2-based adaptive streaming of HEVC video over 4G/lte networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.

[37] J. van der Hooft *et al.*, "An HTTP/2 push-based approach for low-latency live streaming with super-short segments," *J. Netw. Syst. Manag.*, pp. 1–28, Mar. 2017. [Online]. Available: https://doi.org/10.1007/s10922-017-9407-2, doi: 10.1007/s10922-017-9407-2.

[38] B. Ciubotaru, G. Ghinea, and G.-M. Muntean, "Subjective assessment of region of interest-aware adaptive multimedia streaming quality," *IEEE Trans. Broadcast.*, vol. 60, no. 1, pp. 50–60, Mar. 2014.

[39] V. Paxson, "End-to-end Internet packet dynamics," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, pp. 277–292, Jun. 1999, doi: 10.1109/90.779192.

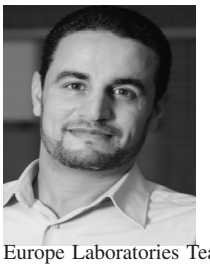
[40] Z. Yuan, G. Ghinea, and G.-M. Muntean, "Beyond multimedia adaptation: Quality of experience-aware multi-sensorial media delivery," *IEEE Trans. Multimedia*, vol. 17, no. 1, pp. 104–117, Jan. 2015.

[41] M. Jain and C. Dovrolis, "End-to-end estimation of the available bandwidth variation range," in *Proc. ACM SIGMETRICS Int. Conf. Measur. Modeling Comput. Syst.*, Banff, AB, Canada, 2005, pp. 265–276. [Online]. Available: http://doi.acm.org/10.1145/1064212.1064242

[42] R. Jain, D.-M. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," Eastern Res. Lab., Digit. Equip. Corporat., Hudson, MA, USA, Rep. DEC-TR-301, 1984.

[43] *Network Simulator 3*. Accessed: Aug. 29, 2016. [Online]. Available: http://www.nsnam.org/

Oussama El Marai received the Engineering degree in computer science from the University of Science and Technology Houari Boumediene, Algiers, Algeria, in 2005, and the master's degree from Ecole nationale Superieure d'Informatique, in 2009, where he is currently pursuing the Ph.D. degree. He is currently a Lecturer with Taibah University, Medina, Saudi Arabia. His research interests include adaptive multimedia delivery, video QoE optimization, and network QoS.



Tarik Taleb (S'05–M'05–SM'10) received the B.E. degree (with distinction) in information engineering, and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, in 2001, 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Finland. Prior to his current academic position, he was a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. He was then leading the NEC Europe Laboratories Team researching on research and development projects on carrier cloud platforms, an important vision of 5G systems. Before joining NEC and till 2009, he was an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, Japan, in a laboratory fully funded by KDDI. From 2005 to 2006, he was a Research Fellow with Intelligent Cosmos Research Institute, Sendai, Japan. He is an IEEE Communications Society (ComSoc) Distinguished Lecturer.

His research interests lie in the field of architectural enhancements to mobile core networks (particularly 3GPP's), mobile cloud networking, network function virtualization, software defined networking, mobile multimedia streaming, intervehicular communications, and social media networking. He was an recipient of the Best Workshop Award by the IEEE ComSoC for the successful event "IEEE Workshop on Telecommunications Standards: from Research to Standards." Based on the success of this workshop, he has also founded and has been the Steering Committee Chair of the IEEE Conference on Standards for Communications and Networking. He has been also directly engaged in the development and standardization of the evolved packet system as a member of 3GPP's System Architecture Working Group. He is a member of the IEEE Communications Society Standardization Program Development Board.

Prof. Taleb is the General Chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference to be held in Marrakech, Morocco. He is/was on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the *IEEE Wireless Communications Magazine*, the IEEE JOURNAL ON INTERNET OF THINGS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and a number of Wiley journals. Till 2016, he served as the Chair of the Wireless Communications Technical Committee, the largest in the IEEE ComSoC. He also served as the Vice Chair of the Satellite and Space Communications Technical Committee of the IEEE ComSoc from 2006 to 2010. He has been on the Technical Program Committee of different IEEE conferences, including Globecom, ICC, and WCNC, and chaired some of their symposia.

He was a (co)recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize in 2017, the 2009 IEEE ComSoc Asia-Pacific Best Young Researcher Award in 2009, the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation in 2008, the 2007 Funai Foundation Science Promotion Award in 2007, the 2006 IEEE Computer Society Japan Chapter Young Author Award in 2006, the Niwa Yasujirou Memorial Award in 2005, the Young Researcher's Encouragement Award from the Japan chapter of the IEEE Vehicular Technology Society in 2003, and the best paper awards at prestigious conferences for some of his research works.

Mohamed Menacer is currently an Assistant Professor with the College of Computer Science and Engineering, Taibah University, Medina, Saudi Arabia, where he is a Founding Member and the Head of research units with the IT Research Center for the Holy Quran and its Sciences (NOOR) and the Chair of the Scientific Committee of the 2013 Taibah University International Conference on Advances in Information Technology for the Holy Quran and its Sciences.

He received the M.Phil. degree from Hull University, U.K., in 1989, and the Ph.D. degree from Nottingham University, U.K., in 1994. He was a Research Associate with De Montfort University, U.K., and Nottingham University, U.K., from 1995 to 2002. He was with Rolls-Royce, U.K., and Dunlop, U.K., from 1999 to 2004. He has published over 40 scientific paper and books and filled for several patents, two awarded so far.

Mouloud Koudil received the Engineer diploma, M.Sc., and Ph.D. degrees in computer science from Ecole nationale Superieure d'informatique, Algiers, Algeria, in 1987, 1991, and 2002, respectively, where he has been a Full-Time Professor since 1991. He has been leading the Codesign Research Group, System Design Methodology Laboratory (LMCS), since 2001. His research interests include embedded systems, networks on chips, wireless sensor networks, and Internet of Things.