ASAP: Adaptive Stall-Aware Pacing for Improved DASH Video Experience in Cellular Networks

AHMED H. ZAHRAN*, Dept. of Computer Science,
University College Cork, Ireland
JASON J. QUINLAN, Dept. of Computer Science,
University College Cork, Ireland
K. K. RAMAKRISHNAN, Dept. of Computer Science and Engineering,
University of California, Riverside
CORMAC J. SREENAN, Dept. of Computer Science,
University College Cork, Ireland

The dramatic growth of video traffic represents a practical challenge for cellular network operators in providing a consistent streaming Quality of Experience (QoE) to their users. Satisfying this objective has so-far proved elusive, due to the inherent characteristics of wireless networks and varying channel conditions as well as variability in the video bitrate that can degrade streaming performance. In this paper, we propose stall aware pacing as a novel MPEG DASH video traffic management solution that reduces playback stalls and seeks to maintain a consistent QoE for cellular users, even those with diverse channel conditions. These goals are achieved by leveraging both network and client state information to optimize the pacing of individual video flows. We evaluate the performance of two versions of stall aware pacing techniques extensively, including stall aware pacing (SAP) and adaptive stall aware pacing (ASAP), using real video content and clients, operating over a simulated LTE network. We implement state-of-the-art client adaptation and traffic management strategies for direct comparisons with SAP and ASAP. Our results, using a heavily loaded base station, show that SAP reduces the number of stalls and the average stall duration per session by up to 95%. Additionally, SAP ensures that clients with good channel conditions do not dominate available wireless resources, evidenced by a reduction of up to 40% in the standard deviation of the QoE metric across clients. We also show that ASAP achieves additional performance gains by adaptively pacing video streams based on the application buffer state.

CCS Concepts: • **Information systems → Multimedia streaming**; • **Networks** → *Network performance modeling*; *Network simulations*;

Additional Key Words and Phrases: Adaptive bitrate video streaming, DASH, QoE, separable programming

*Also with Electronics and Electrical Communication Engineering Dept., Cairo University, Egypt.

Authors' addresses: Ahmed H. Zahran, Dept. of Computer Science,
University College Cork, Ireland, a.zahran@cs.ucc.ie; Jason J. Quinlan, Dept. of Computer Science,
University College Cork, Ireland, j.quinlan@cs.ucc.ie; K. K. Ramakrishnan, Dept. of Computer Science and Engineering,
University of California, Riverside, kk@cs.ucr.edu; Cormac J. Sreenan, Dept. of Computer Science,
University College Cork, Ireland, cjs@cs.ucc.ie.

ACM Reference Format:
Ahmed H. Zahran, Jason J. Quinlan, K. K. Ramakrishnan, and Cormac J. Sreenan. 2018. ASAP: Adaptive Stall-Aware Pacing for Improved DASH Video Experience in Cellular Networks. *ACM Trans. Multimedia Comput. Commun. Appl.* 9, 4, Article 6 (June 2018), 23 pages

1 INTRODUCTION

Video streaming over cellular networks is growing rapidly, and is expected to reach 75% of all cellular traffic by 2020[1]. This rapid growth has network operators playing catch-up trying to ensure that all of their customers are given a consistent Quality of Experience (QoE). In cellular networks, offering quality assurances is especially challenging due to the complexities of the wireless transmission medium. In a recent study by Conviva[2], a majority of users identified video stalls as the most irritating factor while streaming. Hence, it is surprising that approximately 50% of video sessions encounter stalls[3]. There is a crucial need to develop a deep understanding of the cause of video stalls, and to design techniques to minimize stalls.

Video streaming today involves a client player fetching segments from a server over HTTP. The most popular standard is Dynamic Adaptive Streaming over HTTP (DASH). The client continuously adapts the selected video quality based on prevailing network conditions. A well-known issue with DASH is when multiple video users compete to share a network link, with several studies [1, 14, 16] demonstrating that quality instability and stalls are very common. Such issues are greatly exacerbated in wireless networks. To illustrate, when multiple users share a cell site, there are at least three interacting control loops involved in the streaming process. The DASH client exercises control on the video quality control loop with the server; the streaming server and client are involved in the underlying TCP congestion control loop; and the base station scheduler controls the resource allocation and scheduling over the downlink air interface control loop to the cellular device. Each of these control loops operates independently and at different timescales, which often leads to degraded streaming performance [4, 8].

Techniques proposed to improve streaming performance consider different approaches including end-to-end [2, 6, 7, 12, 16, 20], network-based [4, 8, 14, 22, 28, 31], and hybrid solutions [3, 11, 22, 25]. In end-to-end solutions, changing the client adaptation logic and/or the server delivery behavior are the common design elements. In network-based solutions, traffic shaping, trans-rating, and/or trans-coding are examples of the functions considered. These approaches maintain the independence between network operators and content providers, but they may lead to sub-optimal performance because of this. For example, a streaming client may trade video quality by adopting an overly conservative estimator for the available network throughput to reduce stalls. Similarly, network-based solutions usually consider only network-state in their decisions and application state is overlooked. Hence, a video client may still suffer from performance degradation, such as video stalls. Hybrid solutions assume interaction between end-nodes and network agents. Additionally, these solutions may assume integrated control loops for both the end-to-end level and the network level by having a 'bird's eye-view' of the end-end system. But this comes at the cost of communication overhead. The MPEG Server and Network Assisted DASH (SAND) standard provides such a framework, with a DASH-Aware Network Element (DANE) as a network agent.

In delivering video over a wired link, the goal is to allocate resources in a fair manner so that similar[4] clients can have equivalent QoE [23]. The nature of wireless access implies that the achievable data rate is a function of the channel condition at each client. In addition, each client's

[1]CISCO Visual Networking Index. http://goo.gl/jFB2L7. Last accessed Apr 25 2017.
[2]Conviva QoE report. https://goo.gl/lwI2At. Last accessed: Apr 25 2017.
[3]mux.com blog. https://goo.gl/SS674Q Last accessed: Apr 25 2017.
[4]The similarity here considers factors such as device capability and other traffic to the user.

channel condition can vary considerably over short time periods due to fading and other factors. Thus, it must be recognized that achieving equivalent QoE across all clients, even if they are similar, may be an unreasonable objective. Our experiments indicate that such a situation naturally arises when video users with persistently different channel conditions share a single base station leading to a large QoE gap. In such situations, users with poorer channel condition are not allocated sufficient resource, leading to degraded QoE due to video stalls, while clients with good channel conditions always seek to maximize their own QoE. Thus, steps must be taken to manage wireless resources to ensure that clients with poorer channel conditions do not suffer unnecessarily by redistributing channel resources aimed at reducing the extremes of QoE across different clients. Specifically, it is desirable for the network to re-allocate resources to support clients on a short term basis, without introducing substantial unfairness. Our approach is to guide this redistribution process while balancing multiple considerations, including that of avoiding stalls.

In this research, we set out to see if we can eliminate stalls (or come close to it) for cellular users through the use of a network agent that is aware of a limited amount of client state, and can judiciously manage resources at the bottleneck wireless link. We further design a solution that can manage the link resources such that the Quality of Experience across all clients is well-balanced, thus promoting fairness. In [36], we propose Stall Aware Pacing (SAP) as a novel network-based solution to improve the streaming performance in cellular systems. SAP indirectly integrates the client-server quality control loop and the base station downlink scheduling process to reduce stalls when a group of users share a congested cellular downlink air interface. SAP achieves this goal by optimizing the delivery rate of individual packets of a flow, based on both application and network states. In particular, we target resources to clients for whom a stall is imminent. These clients are often, but not always, those at the cell edge. In this paper, we propose ASAP as an adaptive version of SAP that adjusts the traffic pacing based on the application buffer state in all the clients. When all the clients have sufficient media (video data) in their buffer, ASAP increases the amount of the traffic forwarded towards the access network, allowing the base station scheduler to improve the utilization of the available system resources. Additionally, we present the separable formulation of our optimization model and show its scalability as the number of users increases. Furthermore, we evaluate the streaming performance in the case of mobile users.

Our performance evaluation based on real video sessions (H.264 video streamed from a server to actual clients running the GPAC video player) over a simulated LTE network shows that both SAP and ASAP improve the stall performance for different DASH client adaptation algorithms in a wide variety of operating conditions in comparison to state-of-the-art techniques. This evaluation setup that combines real nodes over an emulated network enabled us to compare different solutions in a controllable environment. Moreover, SAP and ASAP achieve this with only a modest reduction in the average delivered video rate. Our contributions can be summarized as follows:

- We developed SAP and ASAP as novel network-based optimized pacing solutions whose design captures the typical user perception for image quality while factoring in the impact of stalls. Additionally, their design provides for differential user treatment to accommodate inherent design features such as device capability or user priority.
- We present a collaborative and a non-collaborative version of SAP and ASAP. In the former, the state of the client buffer is relayed by the client to the network-based pacing module while in the latter an in-network algorithm operates on an estimate of the buffer-level at the client.
- We evaluate SAP and ASAP using a laboratory testbed with video clients using the GPAC player to play streaming video content. The content is real H.264 video streamed from a server. Our emulation-simulation hybrid environment simulates the LTE cellular network in a wide variety of scenarios, including different user topologies and fading conditions.

• We show that SAP reduces the number of stalls and the average stall duration by up to 95%, leading to a dramatic reduction in the stall QoE penalty, by up to 85%. We show that SAP reduces the variability in the client QoE by up to 40%, being especially effective when clients operate with different network conditions. Finally, we conclude that the extra requirement for access to client state information in collaborative SAP does not provide a noticeable benefit over non-collaborative SAP.

• We show that ASAP improves stall and quality performance leading to increasing the overall system QoE by up to 5%.

The rest of this paper is organized as follows. Background and related work is presented in Section 2 followed by the design of SAP and ASAP in Section 3. We then present our performance evaluation setup and results in Section 4. Conclusions and future work are presented in Section 5.

2 BACKGROUND AND RELATED WORK

Adaptive streaming over HTTP, recently standardized as DASH, is becoming the dominant technique for transmitting video due to its ability to traverse firewalls and the abundance of HTTP infrastructure. With DASH, the video is split into multiple segments and each segment is encoded into different representations varying in their qualities. DASH video clients may change the video quality at segment boundaries in response to variations in operating conditions. Client adaptation strategies span different approaches including buffer-based, rate-based, and hybrid approaches [19].

Huang et al. [15] propose a buffer-based strategy by which the buffer-level is mapped to the selected video quality. Jiang et al. [16] propose FESTIVE as a rate based heuristic with randomized scheduling of segment requests and stateful adaptive rate update strategy. In [6], De Cicco et al. propose ELASTIC in which segment quality selection is based on a proportional integral controller. In [20], Li et al. propose PANDA that employs self-traffic network probing to establish an accurate estimate of the available network throughput. These adaptation strategies are a few representative examples and the reader is referred to [19], and the references therein, for more extended coverage of different adaptation strategies. The design of these strategies usually include design elements to avoid stalls, such as maintaining a high value for the playout buffer occupancy and/or conservative rate estimators. However, achieving the best streaming performance in the highly variable operating conditions of a cellular network is difficult to achieve while solely relying on client adaptation.

A few server-based techniques have been proposed to improve streaming performance. In [7], De Cicco et al. propose a closed-loop controller implemented at both the client and server to take adaptation decisions and regulate packet transmissions from the server. In [12], Ghobadi et al. propose Trickle as a server-side solution that enforces an upper-bound on the TCP congestion window. This bound is adjusted according to the streaming rate and round-trip time. However, such solutions are agnostic to the underlying operating conditions and are not designed to achieve the best performance in highly variable bandwidth conditions observed in cellular networks, which is the focus of our work.

A number of network-based solutions proposed to improve network resource sharing among multiple video clients are more directly related to our work. In [14], Houdaille and Gouache show that video rate shaping at a WiFi home gateway reduces the number of quality changes and oscillations for both Microsoft's Smooth Streaming and Apple's HLS. In [28], Pu et al. employ a proxy at the edge of network core. This proxy implements split TCP, rate-dependent packet prioritization, and video transrating based on optimizing a system objective function integrating user rate-utility, smooth rate switching, and buffer-level tracking. [35] takes advantage of mobile-CDN, as part of the network operator infrastructure, to implement an application-level fair scheduler for video rate control. The proposed scheduler integrates user, device, network, and TCP information in its decisions. In [8], a target rate for every data and video stream is communicated to an underlying minimum rate proportional scheduler at the base station. This target rate is estimated using an adaptive guaranteed bit rate algorithm that is shown to match the optimal solution of maximizing the total user utility under resource constraints. In [4], Chen et al. propose the AVIS scheduling framework that throttles each stream to a rate in a specific range. The minimum rate in the range is identified by solving an optimization problem that maximizes the total user utility in the wireless system assuming limited resources. The maximum rate is determined based on the allocated minimum rate and the next higher video rate. A crucial point of distinction for our SAP solution is that we adopt a QoE-driven utility that integrates stall probability in its decision, and the ability to incorporate device heterogeneity as a factor. Furthermore, the aforementioned network-based solutions do not allow one to benefit from state updates from clients, thus distinguishing SAP, as it can operate whether or not such collaboration is available.

Collaborative solutions assume some level of interaction between clients and network elements. In [18], the authors investigate the impact of assisting DASH clients by hinting the recommended quality rate in a large scale experiment. The authors show that such assitance leads to reducing stalls and quality switches. In [22], Mok et al. propose using a network proxy in the content provider network to assist the client in measuring the available bandwidth by monitoring packet round trip times. The client integrates this estimate in its QoE-driven adaptation policy. In [31], the authors show that maximizing the minimum buffer-level for multiple non-adaptive video clients in a wireless system is an NP-hard problem. With the client providing the buffer-level at every epoch, they propose a greedy scheduler for non-adaptive video and show its optimality if the wireless medium remains stable between decision epochs.

In [3], Bouten et al. propose an in-network QoE-driven quality selection based on both network and client information. The wired network resources are monitored using a packet sampling approach to accurately forecast future available resources in wired networks. The network then solves an optimization problem that maximizes a total video QoE metric that includes received quality, the number of rate switches and stalls, subject to resource constraints. The result is passed directly to the client which has been altered to use it in selecting the quality of the next segment. Georgopoulos et al. [11] propose a framework to identify individual user quality for fair resource allocation among a group of users. The estimated rate is then communicated directly back to the customized client using the northbound interface of the software defined network controller. In a collaborative setting, Cofano et al. [5] compared the performance of different network control strategies that employ combinations of optimized bandwidth reservation and hinting of the quality rate using a centralized SDN controller. The optimization framework is solely based on a rate utility function that overlooks other relevant QoE-aspects such as stalls and switching. In [17], Kleinrouweler et. al. show, using a simple SDN-based network controller, that assisting video clients by hinting a quality rate can significantly improve the switching performance of video clients. In contrast, SAP uses the solution of its optimization to *indirectly* impact the client behavior, not requiring that the clients be altered to accept directions from the network. Thus, SAP also maintains the independence of control functions of both network and content providers, which from a practical viewpoint is advantageous for companies operating in this highly competitive business sector.

3 SAP

3.1 SAP Overview

SAP is a stall-aware network element that manages a group of video flows and seeks to minimize stalls, while improving the QoE of video clients sharing a cellular base station (BS). SAP achieves

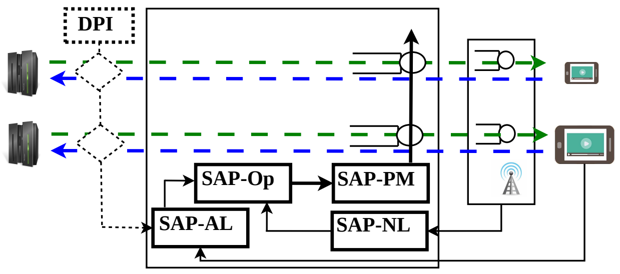


Fig. 1. SAP Architecture, comprising the application state logger (SAP-AL), network state logger (SAP-NL), SAP pacing optimizer (SAP-Op), and SAP pacing manager (SAP-PM)

this goal by pacing the delivery rate of video packets while considering both client application and network state. Network state is captured by the amount of resources dedicated to clients receiving video traffic, the number of clients and the channel quality information for individual clients. Application state information includes video encoding information and the current video buffer-level at the client. Figure 1 shows the SAP components.

SAP may operate in either a collaborative or non-collaborative mode. In collaborative mode, the streaming client provides the application logger the current application state information. In the non-collaborative mode, SAP has to adopt different techniques (e.g., deep packet inspection) to obtain or estimate the application state information. SAP's network state logger interacts with network entities to collect relevant network state information including the number of DASH clients, their corresponding channel quality information, and the amount of resources dedicated for them at the BS. The pacing optimizer integrates the application and network state information to determine the best delivery rate for each individual video flow. SAP's pacing manager controls the delivery rate for each flow based on the rate calculated by the optimizer.

3.2 Design of the SAP Pacing Optimizer

Typically, the adaptation algorithm of DASH determines the quality of the next segment it is going to request. The time between consecutive requests is therefore of the order of a segment duration, which is typically between 4-10 seconds for cellular networks. On the other hand, the cellular BS scheduler executes every few milliseconds, and in LTE it allocates resources every two milliseconds. Note that the base station maintains individual user queues, as shown in Fig. 1, which are serviced by the scheduler according to the adopted resource allocation policy. Hence, implementing a fully assisted solution, e.g., [5, 17], that performs rate hinting and/or rate control would require changing the BS scheduler to enforce such traffic management policies. Considering the complexities involved in changing the scheduler design, SAP adopts a different design that *indirectly* impacts the client quality control and scheduler resource allocation decisions.

SAP's optimization executes periodically at a frequency in between this BS scheduler frequency and the client application adaptation frequency. We expect a typical period between 250 ms and 1 second as being suitable for performing this pacing optimization. Such period would enable SAP to perform several rate adjustments per segment to orchestrate the interaction between different streams without incurring excessive unnecessary processing overhead. Additionally, SAP may be executed in response to a predefined set of triggers, such as client departure, client arrival, change in network resources and/or similar related events. Alternatively, SAP's optimizer could also be invoked in a hybrid combination of both periodic and explicit trigger events.

3.2.1 System Model. We consider a base station (BS) serving U DASH users that each have a different channel quality identified by its spectral efficiency per resource unit (RU) $\gamma_u kbps/RU$, where $u \in \{1..U\}$. We assume that the BS capacity is divided into a group of allocatable resource units managed by the BS scheduler. In LTE, a resource block group (RBG) at an eNodeB represents the allocatable resource unit. The size of a RBG would vary depending on the BS bandwidth. For example, the RBG is a single resource block for a BS with 1.4MHz cell bandwidth while 2-resource blocks form a single RBG in a BS with bandwidth of 3MHz or 5Mhz. As the BS bandwidth increases, RBG are formed from larger resource block clusters to simplify the scheduler design. We assume that C RUs are dedicated for DASH users. Generally, C would vary dynamically depending on different factors, including the volume of non-video traffic that may be sharing the BS with DASH clients. We assume the presence of a bandwidth slicer that distributes the cell bandwidth among different traffic classes [4], e.g., DASH video, non-DASH video, and background traffic. In compliance with DASH, we consider each video is split into S segments with each segment corresponding to a duration of τ seconds. The videos are encoded into Q quality representations whose average rates are denoted r_q, where $q \in \{1..Q\}$.

3.2.2 SAP QoE-Oriented Design. The SAP pacing optimizer integrates stalls and video quality as key aspects affecting the QoE performance of video delivery over cellular systems. Specifically, SAP maximizes a video quality utility metric and minimizes a stall penalty that is a significant factor for user QoE. SAP's rate changes are presented to the client adaptation logic in a manner similar to changes in the channel condition. Our evaluation confirms that SAP does not adversely impact the end-client switching performance and in many cases reduces the number of quality switches.

SAP captures the visual quality using a tunable concave quality utility metric inspired by the video quality metric (VQM) [27]. For the same content encoded at different rates, VQM varies between 0 to 1, with higher quality having a lower value. Further, it is well understood that as the encoding rate increases, the marginal improvement in quality reduces. Therefore, we propose an exponential video quality utility measure, denoted by $\Upsilon_u(x_u)$, expressed as

$$\Upsilon_u(x_u) = (1 - e^{-\rho_u x_u / r_q}), \tag{1}$$

where x_u is the rate that would be allocated to user u and ρ_u is a tunable parameter that can be set according to the device capability or based on operator requirements. Note that in Eq. (1), higher rates have larger utility. Also, $\Upsilon_u(x_u)$ would have a larger marginal utility for low video rates in comparison to higher ones. With users viewing video on a range of devices with varying capabilities, the video rate utility would correspondingly vary across heterogeneous devices. One option to account for this heterogeneity is to tune ρ_u by the operator to differentiate the quality of service provided to users with different priorities. Given the exponential utility function is upper-bounded by 1, ρ_u can be estimated as

$$\rho_u = -\frac{r_q}{\overline{r_u}} \log(\epsilon), \tag{2}$$

where $\overline{r_u}$ represents the maximum rate assigned to user u due to device capability or assigned user priority. With ϵ being a small fraction, this design implies that $\Upsilon_u(\overline{r_u}) = 1 - \epsilon$ and that any increase in the streaming rate beyond $\overline{r_u}$ for user u would lead to an insignificant change in the user utility.

SAP seeks to capture the stall impairment by estimating the probability of buffer depletion at the client, i.e., video stalls. A high probability value implies frequent stalls and/or longer stall duration. User u would typically stall if the download time of the requested segment, denoted as d_u, is larger than the segment deadline D_u. D_u can be estimated as the time remaining until segment playout. Since segments are typically downloaded sequentially, the next segment deadline equals the buffer-level at the time of sending the segment request. The download time d_u depends on the downloaded segment size and the delivery rate of this segment to the client. The downloaded segment quality is independently selected by the client according to its adaptation policy based on the application state information. Assuming that the wireless access link is the bottleneck, the delivery rate of the segment would mainly depend on the resource scheduling at the BS, which of course only considers network state information when it is allocating resources to its users.

In essence, SAP indirectly controls the BS scheduling, by managing the flow of packets toward the BS, considering both network and application states. This is performed at an intermediate timescale between the small BS scheduling period and the much longer quality adaptation time scale. Hence, when the SAP pacing manager is executed, users can be classified into one of two states

• new-request state in which the user has recently finished downloading a segment and is requesting a new one, or

• mid-request state in which the user is still downloading a previously requested segment.

With the first class, SAP determines the stall probability of user u, denoted as π_u, as

$$\pi_u \quad = \quad Prob(d_u > D_u) = Prob\left(\frac{S_{\vartheta_u}}{x_u} > D_u\right),$$

where x_u represents the rate at which user u's packets would be delivered to the BS and S_{ϑ_u} represents a segment size random variable conditioned on the selected quality level $\vartheta_u \in \{1..Q\}$ corresponding to the selected x_u. It is worth noting that for the new-request state, x_u represents the quality rate of the new segment that should ideally be requested by the client from SAP's point of view. This quality identifier ϑ_u is maintained by SAP as part of the flow state until it is updated with the next segment request. Hence, if the system state made SAP to select a high quality rate at the beginning of the segment, this decision would be supported until the next segment request, even when network state changes.

If the user is in the mid-request state, the SAP pacing manager calculates the stall probability using the conditional residual segment size distribution, denoted as $F_{S_{\vartheta_u}}(.)$, and is expressed as

$$\pi_u = \frac{1 - F_{S_{\vartheta_u}}(b_u + D_u x_u)}{1 - F_{S_{S_{\vartheta_u}}}(b_u)}, \tag{3}$$

where b_u represents the total transmitted bytes for the currently downloaded segment of user u. Note that segment download deadline D_u is reduced by the time elapsed since the last execution of the SAP pacing optimization. Clearly, there is a need to monitor both downlink and uplink. The downlink is monitored to determine the number of bytes transmitted per segment for each active user. The uplink is monitored to identify new segment requests.

3.2.3 SAP Optimization Program. SAP maximizes the total quality utility minus the total stall penalty for all users. This design is inspired by the desire to provide an improvement in the QoE performance of video delivery on the cellular network, and particularly to seek consistent QoE for users, even those with diverse wireless channel conditions. Our SAP pacing manager optimization program is expressed as

$$\max_{x_i} \sum_{u=1}^{U} (\Upsilon_u(x_u) - \beta\pi_u)$$

such that

$$\sum_u \quad x_u/\gamma_u < \zeta \quad C \tag{4}$$

$$x_u \quad \in \quad \{\hat{r}_1, \ldots, \hat{r}_Q\} \tag{5}$$

Table 1. Weibull distribution parameters for segment sizes

quality	1	2	3	4	5	6	7	8	9	10
Shape	2.65	2.7	2.72	2.84	2.9	3.18	3.07	3.21	3.15	3.20
Scale	132	210	313	419	586	973	1309	1666	2140	2388

where β represents non-negative weights for the switching and stall penalties, respectively. $\hat{r}_i$ represents the scaled version of the video encoding rate, and ζ represents a scaling factor to avoid resource underutilization at the BS. We scaled the encoding rates r_i to $\hat{r}_i$ to compensate for the overhead of lower layers such that the application goodput would match the target encoding rate. Additionally, we employed the scaling factor ζ to avoid resource underutilization, resulting from performing SAP rate control using the reported wide-band spectral efficiency while the scheduling performed at base station is based on individual resource unit spectral efficiency.

The integration of quality and temporal components in the SAP optimization assist streaming clients to avoid stalls, irrespective of the cause of stalls. Stalls may happen for different reasons including significant changes in the user channel condition, sudden changes in network load (e.g., arrival of new users), and/or large increase in the video's bandwidth demand due to the inherent variable bitrate of compressed video. SAP accommodates many of these scenarios by dynamically re-adjusting the resource allocation and thus controlling the packet delivery process to the BS scheduler. By throttling a specific user, the BS is implicitly forced to serve the traffic of other users.

The operation of the SAP pace optimizer depends on the availability of a set of conditional distributions for segment sizes using different encoding rates for a given segment duration. These can be easily determined by fitting the segment size data to a suitable distribution. A set of such distributions are prepared in advance by the network operator for a wide range of content, with segment size data made available using one of several techniques. One option is to fetch byte-range MPD files if they are available; these provide segment sizes for each encoding rate. Or segment sizes can be obtained by iteratively downloading segments at several fixed qualities from the content provider and recording the segment sizes. Alternatively, if there was cooperation with the content provider, segment size data (or indeed distributions) could be provided to the network operator directly. In this work, we obtain segment size data from the full iVID dataset [30] and fit this data to a Weibull distribution using fitdistplus package in R.

Table 1 shows the scale and shape parameters of the fitted Weibull distributions for different quality levels of all movies in the publicly available iVID dataset. The segment duration for these is 4 seconds.

The SAP pacing optimization program can be classified as a non-linear discrete optimization problem. Solving such problems is usually time consuming and not necessarily feasible in real-time. However, we ensure real-time operation by taking advantage of the problem structure. Since all nonlinear terms are functions of a single optimization variable, the program can be formulated as a *separable programming* problem that can be solved at a speed similar to linear programs [10, 13].

The separable model of SAP can be expressed as

$$\max \sum_{u=1}^{U} \sum_{q=1}^{Q} (\Upsilon_{uq} - \beta \pi_{uq}) \theta_{uq} \tag{6}$$

such that

$$\sum_{q=1}^{Q} \theta_{uq} = 1 \ \ \forall u \in \{1, .., U\}\,, \tag{7}$$

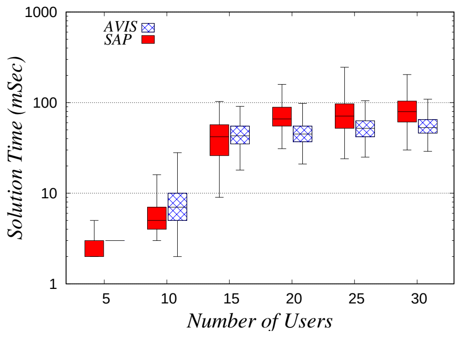


Fig. 2. Solution time vs number of clients

$$\sum_u \frac{\sum_{q=1}^{Q} \hat{r}_q \theta_{uq}}{\gamma_u} < \zeta C, \tag{8}$$

$$\theta_{uq} \in \{0, 1\} \tag{9}$$

where Υ_{uq} and π_{uq} represent the quality utility and stall penalty for user u when assigned video quality rate $\hat{r}_q$, respectively.

It is worth noting that separable modeling can be applied to other relevant streaming solutions that share the common program structure, i.e., non-linear terms are only function of a single optimization variable. For example, AVIS [4] also enjoys the same characteristic and it can be reformulated as a separable program. Hence, one can achieve both optimized operation in real-time.

Figure 2 shows the box-plot the solution time of the separable implementation of both SAP and AVIS versus the number of optimized user session. This figure is generated based on the solution of 500 randomly generated program instances using the same video specifications in the iVID data set [30]. Additionally, cellular network relevant parameters (e.g., spectral efficiency, BS bandwidth, etc.) matches those defined in standard LTE networks. The results shown were obtained using a laptop with an Intel Core i7-4810MQ (2.8GHz) processor and 8GB RAM. It is worth noting that we scaled the base station bandwidth as the number of users increases in order to ensure the program feasibility, i.e., ensuring the presence of adequate resources to serve existing clients.

The figure illustrates that the program can be solved in few milliseconds for small scale programs. For example, the solution time of programs with six users is less than 6msec. Additionally, it shows that the solution time remains bounded to tens of milliseconds as the number of optimized sessions increases. To illustrate, the solution time of SAP remains less than 110 milliseconds for the case of 30 users, which is foreseen as a sufficient bound on the number of video clients that are simultaneously streaming video in the same cell. Hence, the separable implementation of SAP offers an optimal resource management approach whose solution time is similar to unoptimized heuristics.

3.3 Collaborative vs. Non-collaborative SAP

The previous section detailed the design of SAP and how it makes use of network and application states. SAP can operate in one of two modes: collaborative or non-collaborative. In collaborative SAP, the application state logger acts as a DANE interface that receives status messages from the client. This provides SAP with encoding information (rates and segment duration) and the client's current buffer-level. Note that the segment duration is used only for identifying the conditional segment size distribution to apply. Encoding parameters would be provided at the beginning of the session, while the buffer-level would be reported on a regular basis during the session. In the non-collaborative case, we assume that the network operator would implement additional in-network functions to identify the required information for SAP. We now explore the non-collaborative case and consider two possibilities:

- ***Non-encrypted client-CDN communication.*** In this case, the operator would be able to extract encoding parameters, such as encoding rates and segment duration, from the DASH description (MPD) file. To estimate the buffer-level, the SAP application state logger would need to rely on deep packet inspection. Generally, the uplink mainly carries HTTP GET requests for video segments and TCP ACK packets. By tracking HTTP requests, SAP can estimate a conservative value for the client application buffer as detailed below.
- ***Encrypted client-CDN communication.*** In this case, SAP has to rely on *a priori* information about video traffic. General guidelines for encoding rates used by different video content providers are publicly available. Hence, SAP may rely on these public rates as alternatives for r_i. Note that the SAP pacing optimizer may use arbitrary discrete rates for r_i with inter-rate gaps following the public guidelines. A typical rate ratio between consecutive rates is 1.5. But, we believe that the minimum rate is the most critical factor. Choosing the right value for the minimum rate is important, especially for users with poor link conditions: choosing too small a value may over-throttle the flow to the user, resulting in excessively poor QoE; on the other hand, choosing too large a value may result in inefficient utilization of the available resources. For our work in this encrypted scenario, we assume a technique to estimate the segment duration by observing the segment size of the first few segments. This technique may assume a small segment duration (e.g., 4 sec) until a more accurate estimate of the segment duration is calculated. Similar to the non-encrypted case, this segment duration would be used to estimate the buffer-level as presented below.

3.3.1 SAP Buffer-level Estimator. The buffer-level estimation is needed in the non-collaborative scenario to identify the delivery deadline of segments. The buffer-level, denoted as b, can be estimated as the difference between the received and played video durations, denoted as D_r and D_p respectively. Hence, the buffer-level can be calculated as

$$b = D_r - D_p. \tag{10}$$

By knowing the segment duration and the number of received segments, the received media duration can be directly estimated as their product. The number of received segments can be determined by monitoring the HTTP GET requests on the user uplink, denoted as g_u for user u. The received video duration may be estimated as $(g_u - 1)\tau$ seconds. It is important to note that this approach would efficiently work in both non-encrypted and encrypted cases, provided that the client requests segments in a sequential manner. However, if the client opts to abandon segments while they are being downloaded, for example to request lower quality segments to avoid stalls, then g_u should correspond to sequential segment requests. Note that in such mechanisms today the client is required to close the TCP connection to trigger abandonment of the requested segment and establish a new connection to request the new segment. Hence, this exchange can be captured to maintain an accurate estimate of g_u.

The playout video duration can be estimated as the difference between the current wall clock, denoted as t_c, and the time at which playout starts, denoted as t_p. The initial buffering depends on the client implementation, but an initial buffering duration between 6 and 10 seconds is recommended to reduce rebuffering events in mobile systems [34]. Our estimator assumes that the client would always start after downloading the first segment. The validity of this assumption is based on the observation that larger segment durations are used for cellular clients. It was shown in [24] that using a large segment duration in mobile networks improves the user QoE and bandwidth utilization. Hence, the estimated buffer-level would be close to the actual buffer-level. However, it would be a conservative estimate for the buffer-level if the client actually has a larger initial buffer. Additionally, this estimate would deviate from the actual buffer if the user intervenes with the session, e.g., pause the playout. Note that in this case, the client would continue to download the media until the buffer saturates and then would stop requesting segments until the playout is resumed. In this case, SAP is agnostic to user action and would consider all clients as actively playing out their entire buffer.

Since we depend on the real-clock to capture the play-out duration of the stream, the duration of an interruption should be used to rectify the buffer-level estimate. In our estimator, stalls are captured by negative buffer estimates. To rectify the buffer-level estimate, we shift our playout reference time to the instant at which the client resumes playout after a stall. We assume that the client would continue the playout once it receives a segment. Additionally, we reset the number of received segments to 1 (i.e., $g_u = 2$). Hence, we can consider t_p as the time at which playout starts or resume after a stall. Similarly, g_u can be considered the number of stall-free received segments since the client started or since the last stall.

3.4 Adaptive SAP (ASAP)

ASAP represents a dynamic SAP that aims to ensure stall immunity for video clients at risk of stalls while avoiding unnecessarily impacting other streams. By design, SAP introduces a controlled queue for individual streams before the typical client buffers at the cellular BS, as shown in Figure 1. By adjusting the packet delivery rate towards BS queues, SAP impacts the behavior of the cellular scheduling. Note that the BS scheduler will only allocate resources for clients with buffered packets at the BS. Hence, SAP improves the stall immunity by pacing video packets to allow clients at risk to be allocated resources by the BS scheduler. These clients are usually suffering from deteriorating network conditions that lead to their playout buffer depletion. Hence, these clients needs more resources to cope with the network condition to avoid stalls. These resources are obtained from the common resource pool shared among all clients leading to reduced packet delivery rates for other clients. While this support is necessary when there is a risk of stalls, it may not be necessary as the buffered media in the application buffer of all clients is sufficiently high. In such scenario, more data can be safely delivered towards the cellular BS queues in order to allow those users with good channel condition to benefit from their high spectral efficiency. Hence, the average system throughput and user QoE can be improved.

ASAP is proposed to achieve both stall immunity and improved video quality by adjusting ζ based on the minimum buffer level among *all* active video clients. A higher ζ implies that SAP operates using a relaxed resource constraint. By delivering more traffic to the access network, the control of SAP over the BS scheduler behavior is reduced. On the contrary, using a smaller ζ improves the chance of resource allocation to users facing the risk of stalls. Hence, ASAP adaptively sets ζ as expressed below

$$\zeta = \begin{cases} \zeta_l & , B_{min} \leq B_l \\ \zeta_h & , B_{min} \geq B_h \\ \zeta_l + \frac{(B_{min}-B_l)(\zeta_h-\zeta_l)}{(B_h-B_l)} & , otherwise \end{cases}$$

where B_{min} represents the minimum buffer level among all the video clients, B_l and B_h represent lower and upper bounds on B_{min}, ζ_l and ζ_h represents a lower and upper bounds for ζ.

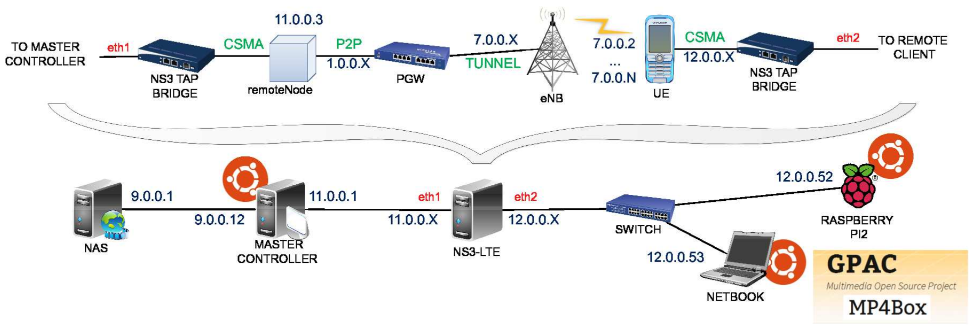


Fig. 3. Evaluation Testbed

4 PERFORMANCE EVALUATION

4.1 Evaluation Setup

Our evaluation testbed is designed based on an empirical methodology that integrates the use of realistic video content and player software, operating over a simulated cellular network, thus promoting realism and enabling controlled repeatability of experiments across a range of configurations. Fig. 3 shows our evaluation testbed, which is based on iVID D-LiTE testbed [29]. In the following subsections, we present the key elements of our testbed. A full implementation details is presented in [36] and its Appendix.

4.1.1 Streaming Clients. In our testbed, DASH clients are running over a version of Ubuntu installed in Raspberry Pi 2 and standard net-books. The clients use GPAC 0.5.2-DEV-rev985[5] that we extend with well-known/recent adaptation algorithms such as BBA2 [15], FESTIVE [16], conventional (CONV) [16, 20], and ARBITER [37]. BBA2 represents the class of buffer-based algorithms in which the quality selection mainly depends on the current buffer-level. FESTIVE represents the class of conservative rate-based algorithms that are designed to operate well in scenarios with shared bottlenecks. CONV represents another rate-based strategy that streams video while trying to maintain the quality at a stable level as much as possible. ARBITER represents the class of hybrid algorithms that integrate both application and network state in the quality selection decision. The parameters of different streaming algorithms are set to the default values reported in the papers cited. In all the evaluation scenarios, a client is configured to perform 8 seconds (two segments) of initial buffering and 4 seconds (one segment) of rebuffering after any stall [32].

4.1.2 LTE Network Setup. The LTE network is implemented using the LTE-EPC network simulator (LENA) module[6] in ns3[7]. In our setup, external nodes are connected to simulated nodes using the ns-3 TAP mechanism, which uses a special net device called a TapBridge. In order to connect LTE UEs to the external streaming clients, a second carrier sense multiple access (CSMA) net-device (12.0.0.x network) is added to the UE, as LTE-net-devices are not compatible with the ns3-TAP bridge. On the other side of the LTE network, the LTE packet gateway (PGW) is connected to the network attached storage (NAS) server through a master node whose functionality is explained below. We have modified the code of both the LENA scheduler and routing modules in ns3 as described below.

[5]https://gpac.wp.mines-telecom.fr/
[6]LENA module. https://goo.gl/6D1Wfq. Last accessed: Dec 8, 2016.
[7]https://www.nsnam.org/

For proper routing of downlink packets, we added static routes for the 12.0.0.x network at the remote node. Additionally, we changed the implementation of ipv4-list-rout-ing.cc to mangle the destination address of downlink packets before entering and after exiting LENA devices to allow the packets to travel through the tunnel between LTE PGW and UEs. The destination address is changed from 12.0.0.x to 7.0.0.x before being forwarded to the PGW. Additionally, the destination addresses of downlink video packets received at an LTE netdevice (7.0.0.x network) are changed back to the actual client 12.0.0.x address and are then forwarded to the ns3 12.0.0.x TapBridge. For uplink traffic, the packets follow the default gateway towards the PGW and are then forwarded based on static routes that are installed at the LTE PGW and remote host node for the NAS network (9.0.0.x) to be forwarded to the appropriate device. Each physical client node (Raspberry PI or netbook) is configured with a default gateway which is the corresponding CSMA netdevice of the connected LTE UE. The packets then proceed to the LTE PGW as the default gateway for LTE UEs.

In LENA, we consider the log distance path-loss channel model [9] for the link between eNodeB and UEs. This pathloss is overloaded with fading traces generated using a tool provided with LENA. The default LENA configuration parameters are used for both eNodeB and pathloss model. All our evaluations are conducted with the proportional fairness (PF) scheduler at the eNodeB. We have modified the PF scheduler implementation only so that user channel quality can be reported to the traffic manager being evaluated.

4.1.3 Master Controller. The master controller node is responsible for orchestrating the evaluation and performing traffic management functions during video sessions. More specifically, this node configures the network and GPAC clients before an evaluation run starts. Note that network configurations include parameters such as eNodeB bandwidth, eNodeB scheduler, and fading model, while the client configuration parameters include the adaptation algorithm and streamed video specifications. At the end of the run, it also collects performance logs from the clients for post processing.

During the evaluation, the master controller performs traffic management functions. In our evaluation, we compare the performance of three different traffic controllers including no traffic control (noTC), AVIS [4], SAP, and ASAP. AVIS focuses on fair user allocation and reduction of quality switches, but overlooks stalls in its design. Both AVIS and SAP optimization programs are implemented using the Lindo solver[8]. The AVIS implementation is based on the continuous version presented in [4] and both ASAP and SAP implementations are based on the separable programming model. In our testbed, the AVIS parameters are set to the default values presented in [4]. The stall weight β is set to 100 and the default utility saturation parameter ρ_u of all clients is set to 3Mbps for both SAP and ASAP. 3Mbps is the maximum rate for HD resolution in the iVID video dataset.

ASAP, SAP and AVIS are periodically executed every 250 ms and the outcome of the optimization program is used to throttle individual user queues using the traffic control command in Linux, the operating system of the master controller node in our testbed. The master node maintains a communication channel with the eNodeB to obtain user channel quality information (CQI) required by both AVIS, ASAP, and SAP. Similarly, it maintains communication channels with streaming clients to obtain relevant application information for the collaborative scenarios. In the non-collaborative scenario, the buffer-level is estimated as presented in Section 3.3.1.

4.1.4 Evaluation Scenarios. In our evaluation, we consider a group of video users sharing a highly loaded LTE eNodeB to capture the impact of streaming in a limited-resource environment. Our performance evaluation shows that different traffic management solutions have similar performance in lightly loaded scenarios. Hence, we focus more on the more relevant highly loaded systems.

[8]http://www.lindo.com/

The eNodeB bandwidth is 1.4 MHz and has a transmission power of 30.2dBm. This eNodeB has 6 resource units split into 6 allocatable resource block groups (RBG) in the downlink. Each RBG can support a PHY rate between 16 Kbps and 712 Kbps depending on the user reported channel quality indicator value. In our evaluation, we consider two different user topologies in highly loaded cells as detailed below.

We consider 6 DASH clients streaming 5-minute, 4-second segment, videos over a single LTE eNodeB. In every session, the clients are introduced to the network separated by a 1 second time gap. Each user streams a different five-minute video from the iVID dataset [30][9], whose videos are encoded with 4-second segments at the following rates {235, 375, 560, 750, 1050, 1750, 2350, 3000, 3850, 4300} Kbps. Note that the base station is only used by the video users. In a more general setup, other traffic may share the eNodeB and video users would be allocated a slice of the total base station resources.

In the following, we first compare the performance of SAP to other traffic manager consider static and mobile scenarios. We then present the performance enhancement achieved by ASAP.

4.2 Performance metrics

Our performance metrics include the average received data rate per session (r_{av}), the average number of stalls per session (n_{st}), the average stall duration per session (t_{st}), the average number of switches per session (n_{sw}), the average switching level (l_{sw}), and a combined QoE metric (x_q) [26], which was originally developed in [21] and [33]. The QoE metric is expressed as [26]

$$x_q = max(0, 0.17 + 5.67\frac{q_{av}}{q_Q} - 6.72 * \frac{q_{std}}{q_Q} - 4.95\varphi),$$

where q_{av} and q_{std} represents the average and standard deviation of the received quality level, and φ represents the stall penalty and is expressed as

$$\varphi = 0.875 * max(0, 1 + ln(f_{st})/6) + 0.008333 * min(t_{st}, 15),$$

where f_{st} represents the frequency of stalls and t_{st} is the average stall duration per session. The results shown represents the average of each metric obtained across 15 runs.

4.3 Performance Results

4.3.1 Scenario 1: Collaborative Static Clients with Diverse Link Conditions . In this scenario, we consider equally separated users with the nearest and farthest users being at 25m and 375m, respectively, from the base station. Fig. 4 illustrates the distribution of the achievable rate per resource unit for each user. Note that these rates are determined by mapping the reported CQI value to the eNodeB as defined in the LTE standards. The figure shows that the closest client to the eNodeB can achieve the highest PHY rate per RU (712 Kbps) almost all the time, while the farthest client's achievable rate is less than 200Kbps for 60% of the time.

Figure 5 illustrates the impact of stall weight on the streaming performance metrics for SAP-50, SAP-100, and SAP-200, where X in SAP-X denotes the stall weight. This figure illustrates that as the stall weight β increases from 50 to 100 a noticeable improvement in the stall metrics is achieved. For example, the average number of stalls, n_{st}, dropped by 63% and the average stall duration, t_{st}, also dropped by 82%. Such drops lead to a reduction of 65% in the stall penalty φ. This improvement is accompanied by a minimal impact only 2% in the average video quality rate. Note that increasing the stall weight implies that SAP would be more sensitive to lower buffer levels and hence, would allocate more resource to users at risk of stalls leading to a drop in the average streaming rate. SAP-100 also shows a noticeable improvement in the switching performance in comparison to

[9]iVID Dataset. https://goo.gl/BP6LWR. Last accessed: Dec 8, 2016.

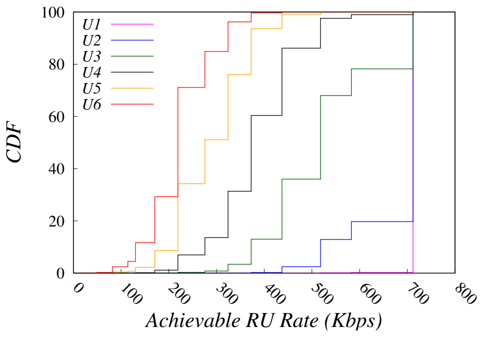


Fig. 4. Achievable rate per resource unit (RU) per user in scenario 1

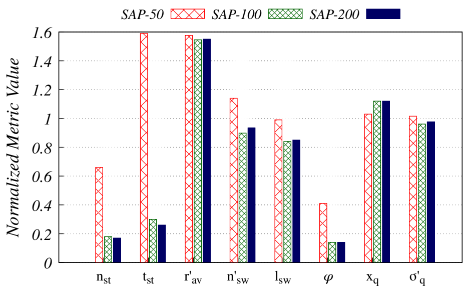
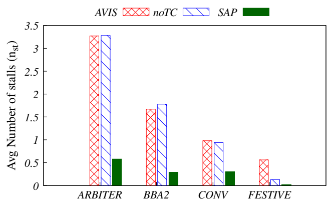


Fig. 5. Normalized streaming performance metrics for various stall weight β. $r'_{av} = r_{av}/235$, $\sigma'_q = \sigma_q/30$, and $n'_{sw} = n_{sw}/5$.

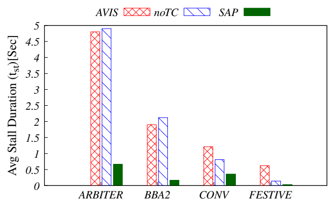
SAP-50. These performance metrics lead to approximately 9% improvement in the average QoE metric x_q. Increasing the stall weight beyond 100 does not lead to a significant improvement in the overall system performance. For example, both SAP-200 and SAP-100 achieve the same stall penalty and the same QoE. In the rest of this document, we show results for SAP-100. Hence, we drop the SAP-X notation.

Fig. 6 shows the average of each of the performance metrics for different streaming algorithms. SAP significantly improves the stall performance for all algorithms in comparison to both noTC and AVIS. The existing diversity in the channel condition enabled SAP to achieve a significant reduction in both the number of stalls, n_{st}, and stall duration, t_{st}. In conjunction with BBA2, SAP reduces both n_{st} and t_{st} by 84% and 94% in comparison to noTC. SAP also reduces both n_{st} and t_{st} of FESTIVE by 95%. Additionally, we note a significant reduction of 83% and 86% for n_{st} and t_{st} when SAP is used with ARBITER. Similar performance gains are achieved by SAP in comparison to AVIS. In fact, AVIS shows stall performance that is similar to or worse than noTC. The improvement in stall performance by SAP thus reduces the stall QoE penalty φ by 50%, 89%, 66%, and 40% for BBA2, FESTIVE, CONV, and ARBITER, respectively. In this scenario with users having diverse link conditions, the user farthest from the eNodeB encounters most of these stalls.

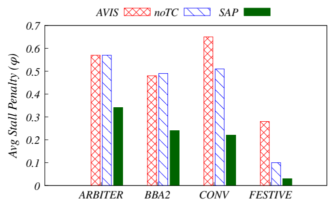
The improved stall performance for SAP is due to its resource management strategy, that not only ensures a minimum rate for every client, but also dynamically paces individual stream packets to protect clients from stalling. Fig. 7 illustrates the resource shares allocated by SAP and AVIS for individual streams. Clearly, Fig. 7a and Fig. 7b illustrates that in SAP more resources are allocated to protect the farthest client who is more exposed to stalls. It is also interesting to observe that



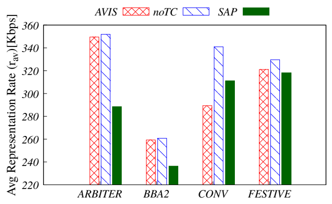
(a) Avg. number of stalls



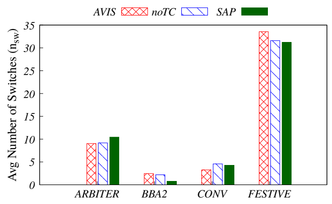
(b) Avg. stall Duration



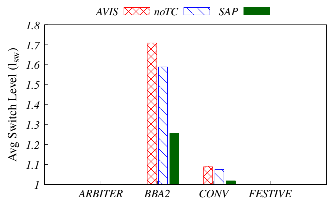
(c) Avg. stall QoE penalty



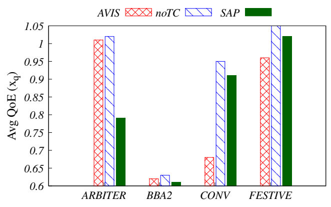
(d) Avg. quality rate



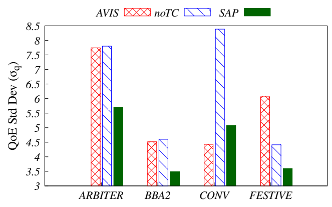
(e) Avg. number of switches



(f) Avg. switch level

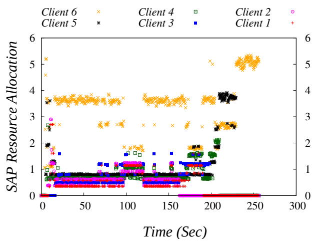


(g) Avg. QoE metric

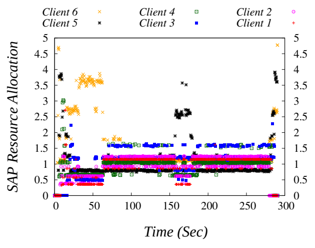


(h) QoE std. dev.

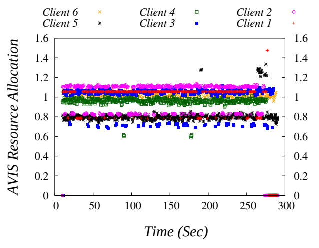
Fig. 6. Performance metrics for scenario 1



(a) BBA2-SAP



(b) FESTIVE-SAP



(c) FESTIVE-AVIS

Fig. 7. Comparing SAP and AVIS resource allocation with different algorithms

with BBA2, SAP is continuously supporting this user by allocating more resources over the entire session duration, in comparison to only the initial part with FESTIVE. Note that BBA2 employs a large buffer (240 sec) and its clients are continuously competing for the network resources. On the contrary, FESTIVE employs a much smaller buffer (30 Sec). Hence, FESTIVE clients with good channel conditions are able to fill their buffer and consequently delay their segment requests. Thus, clients with poorer channel conditions are offered more transmission opportunities by the eNodeB and SAP stall avoidance would kick in at a much lower frequency in comparison to BBA2. On the other hand, Fig. 7c shows that while AVIS tends to avoid frequent rate changes it does not provide the same level of stall protection to video clients.

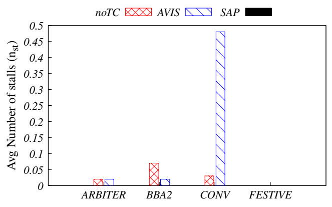
Fig. 6d shows that introducing a network traffic controller, e.g., AVIS or SAP, results in reducing the average representation rate r_{av}. This reduction is expected, as pacing the traffic would slow the delivery rate of packets, leading to lower rate estimates for rate-based strategies or lower buffer-levels in buffer-based strategies. Fig. 6e and Fig. 6f show that SAP significantly reduced the average number of switches n_{sw} and average switching level l_{sw} of BBA2 by 64% and 60%, respectively. Excluding BBA2, the switching performance of the remaining algorithms is not significantly affected by the traffic pacing/control functions.

Fig. 6g shows that the QoE metric x_q drops when SAP is used. This reduction is due to SAP reshuffling resources to help suffering clients - which are those close to the cell edge. Such reshuffling reduces x_q for users close to the eNodeB and increases x_q for distant users.We emphasize that SAP provides improved fairness, by reducing the variance in QoE as shown in Fig. 6h. The latter figure shows that SAP reduces σ_q by 40%, 40%, 24%, and 26% in comparison to noTC for BBA2, FESTIVE, CONV, and ARBITER, respectively.

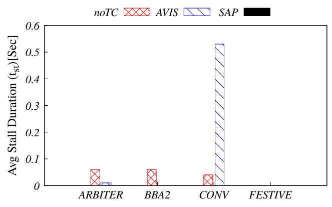
4.3.2 Scenario 2: mobile collaborative clients. In this scenario, we consider six mobile users moving in a bounded box around the base station. The box side is determined such that the maximum distance between any client and the BS is 400m. The users are initially located equally spaced along the cell diagonal and are allowed to randomly move according to a Gauss-Markov mobility model with an average speed of 1 m/s and a memory factor equal to 0.8. When the user reaches the simulated boundary, it moves back into the cell. Typically, the user who is handed off to another cell would see similar network conditions in the new cell that he has been handed off to in a uniformly loaded network.

Figure 8 plots the performance metrics for mobile users using different algorithms. The figure shows that user mobility improves the user streaming performance in different ways. First, the users have fewer stalls and higher quality rate in comparison to the static user scenario. Additionally, the figure shows that SAP successfully eliminates the stalls encountered by the user for all the evaluated streaming algorithms. Figure 8d also shows that, for all streaming strategies, SAP improves the average video quality rate r_{av} in comparison to both noTC and AVIS. This improvement reaches 4-5% in comparison to noTC. Figure 8e shows that SAP has insignificant impact on the average number of switches. Additionally, the impact of SAP on the average switching level differs among the tested algorithms as shown in Figure 8f. We believe that the switching behavior is generally dominated by the design of the quality adaptation strategy. The average QoE is boosted by SAP for all adaption strategies due to stall elimination and increased representation rates. However, this improvement is accompanied by the increase of the standard deviation among streaming clients.

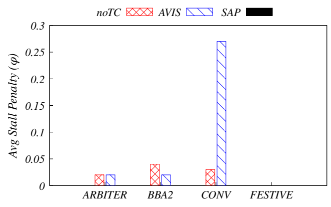
4.3.3 Collaborative vs. Non-collaborative SAP. In this section, we investigate the performance of SAP in non-collaborative mode in which the buffer-level is estimated by SAP in comparison to being collaboratively provided by the client. In the non-collaborative mode, we consider that encoding information is available to SAP, but the instantaneous buffer-level has to be estimated by SAP, using the estimation technique presented in Section 3.3.1. Fig. 9 shows the streaming performance metrics averaged over all sessions. Note that we normalized some of the metrics to fit all of them in one figure, and the normalization factors are indicated in the caption. This figure shows that both collaborative and non-collaborative SAP achieve similar performance metrics for all the adaptation algorithms. Additionally, Fig. 10 plots the correlation between individual client performance metrics for both collaborative and non-collaborative scenarios. The figure shows a high correlation for all metrics for all adaptation strategies except for the FESTIVE stall metric. The low correlation for the FESTIVE stalls is due to a minor variation in the stall distribution



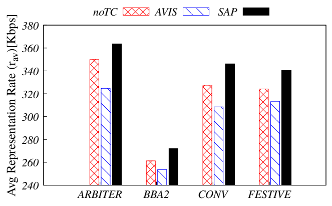
(a) Avg. number of stalls



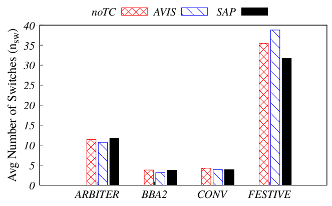
(b) Avg. stall Duration



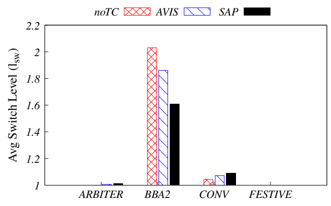
(c) Avg. stall QoE penalty



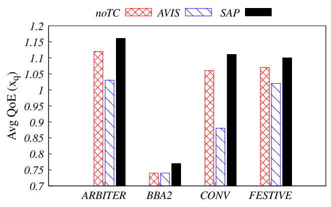
(d) Avg. quality rate



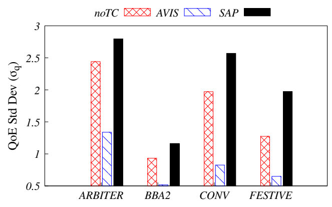
(e) Avg. number of switches



(f) Avg. switch level



(g) Avg. QoE metric



(h) QoE std. dev.

Fig. 8. Performance metrics for scenario 2

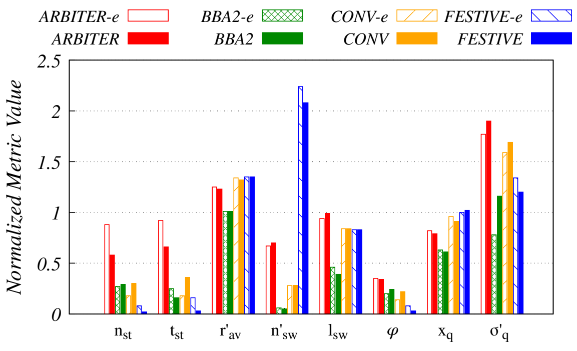


Fig. 9. Normalized streaming performance metrics for collaborative and non-collaborative scenarios. -e denotes the non-collaborative version. $r'_{av} = r_{av}/235$, $\sigma'_q = \sigma_q/3$, and $n'_{sw} = n_{sw}/15$.

across the clients. Only the farthest client encounter stalls in the collaborative scenario while in the non-collaborative scenario the second farthest client also has a couple of stalls. Hence, we conclude that integrating the buffer estimation technique with SAP can be sufficient, thus avoiding any frequent updates from the streaming end client. Consequently, the network does not need to

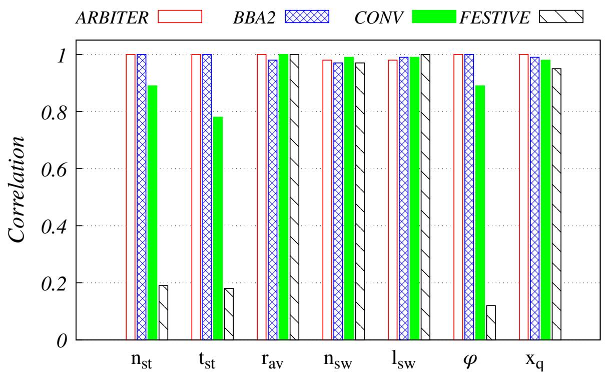


Fig. 10. The correlation between the performance metrics of collaborative and non-collaborative scenarios

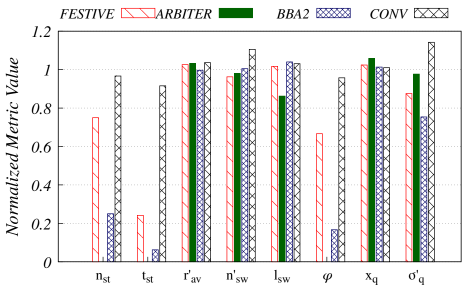


Fig. 11. ASAP performance normalized to corresponding SAP metrics.

have a direct interface with client software or be aware of the local state of the individual client adaptation.

4.3.4 ASAP. We evaluate the performance of ASAP in the static user setup presented in Section 4.3.1. In the shown results, we set ASAP parameters as follows: $\zeta_l = 1$, $\zeta_h = 1.2$, $B_l = 5$, and $B_h = 10$. Figure 11 plots the performance metrics of ASAP normalized to the corresponding SAP metrics for different algorithms. ASAP improves the stall performance across all the algorithms. ASAP reduces the number of stalls by up to 100%. Additionally, ASAP increases the average quality rate for the tested adaptation strategies, by up to 3.6%, except for BBA2, for which a drop of 1% is observed. We believe this drop is the continuous activity of BBA2 due to it large buffer size (240 sec). Hence, clients never manage to fill in their buffer and are continuously competing for resources. Hence, users in worse network conditions may not have the chance to fill in their buffer and hence, ASAP usually operates on low values for ζ. On the other hand, other strategies with smaller buffers fill in their buffers and temporarily do not use network resources allowing other clients to fill in their buffers. Hence, ASAP operates at higher values for ζ_h allowing higher requested streaming rates. ASAP shows insignificant impact on the switching performance of SAP. The overall QoE metric is boosted by 1%-5% for different algorithms.

The implementation of a assisted streaming solution, such as SAP and ASAP, needs to maintain flow state and to act upon this state. This state comprises a network part (e.g., link quality) and an application part (e.g., buffer-level). The storage requirements for such information is quite small and can easily scale with the number of user flows. Note that SAP and ASAP decisions are generated only for the users sharing a single base station. Hence, the solution can be implemented in a scalable manner, with separate instantiations for each base station, and could be implemented even in a distributed manner. Thus, we do not anticipate any scalability issues from a processing perspective. There may be additional requirements depending on the implementation. For example, a collaborative solution implies the implementation of an interface to interact with streaming endpoints, such as a DASH-Aware Network Element (DANE) as a network agent. On the other hand, a non-collaborative solution would require the implementation of a buffer estimator per flow. In both cases, the control signaling load will be limited and can easily scale.

5 CONCLUSIONS

The ability of DASH video streaming clients to operate effectively over cellular networks is poor, due to the inherent variability of both video datarates and the wireless channel quality. In highly loaded systems, these characteristics may lead to streaming issues, such as stalls. Unlike the situation when using wired links, it must be recognized that achieving equivalent QoE across all clients is not a reasonable objective. Fairness is not absolute, but is rather a function of the channel conditions available to each client. Thus, steps must be taken to manage wireless resources to ensure that clients with poorer channel conditions do not suffer unnecessarily, even as clients with good channel conditions seek to maximize their own QoE. This can be achieved by the network redistributing channel resources so as to reduce extremes of QoE across clients. SAP seeks to re-allocate wireless resources in favor of a client that becomes less likely to experience a stall, without significantly degrading the QoE of other users. In this paper, we present SAP as a traffic management solution to improve the streaming performance of a group of DASH video users competing for a base station's resources. SAP integrates both application and network state to optimally pace individual stream delivery. Our extensive experiments show that SAP significantly reduces video stalls encountered by different users in comparison to the state of the art solutions. Additionally, SAP reduces the QoE variability across multiple clients, leading to a more consistent experience for static users. The introduction of adaptive pacing rate in ASAP further enhances the streaming performance by reducing the stalls and improving the user QoE. Our current evaluation setup involves a combination of real-experimentation and simulations operating in real-time. This setup limits the scalability of our evaluation and in the future we plan to explore ways in which we can address this limitation.

ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant No.: 13/IA/1892. This work was supported in part by NSF grant CNS-1619441.

REFERENCES

[1] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis. 2012. What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?. In *Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '12).* Toronto, Canada, 9–14.

[2] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. 2013. Server-based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players. In *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '13).* Oslo, Norway, 19–24.

[3] N. Bouten, R. de O. Schmidt, J. Famaey, S. Latré, A. Pras, and F. De Turck. 2015. QoE-driven in-network optimization for Adaptive Video Streaming based on packet sampling measurements. *Computer Networks* 81 (2015), 96 – 115.

[4] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang. 2013. A Scheduling Framework for Adaptive Video Delivery over Cellular Networks. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom '13).* Miami, Florida, USA, 389–400.

[5] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo. 2016. Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. Article 3, 12 pages.

[6] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. 2013. ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH). In *20th International Packet Video Workshop (PV)*. San Jose, CA USA, 1–8.

[7] L. De Cicco and S. Mascolo. 2014. An Adaptive Video Streaming Control System: Modeling, Validation, and Performance Evaluation. *Networking, IEEE/ACM Transactions on* 22, 2 (April 2014), 526–539.

[8] D. De Vleeschauwer, H. Viswanathan, A. Beck, S. Benno, Gang Li, and R. Miller. 2013. Optimization of HTTP adaptive streaming over mobile cellular networks. In *INFOCOM, 2013 Proceedings IEEE*. Turin, Italy, 898–997.

[9] V. Erceg, L. J. Greenstein, S. Y. Tjandra, S. R. Parkoff, A. Gupta, B. Kulic, A. A. Julius, and R. Bianchi. 1999. An empirically based path loss model for wireless channels in suburban environments. *IEEE Journal on Selected Areas in Communications* 17, 7 (Jul 1999), 1205–1211.

[10] A. Galperin and Z. Waksman. 1981. A separable integer programming problem equivalent to its continual version. *J. Comput. Appl. Math.* 7, 3 (1981), 173 – 179.

[11] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. 2013. Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking (FhMN '13)*. Hong Kong, China, 15–20.

[12] M. Ghobadi, Y. Cheng, A. Jain, and M. Mathis. 2012. Trickle: Rate Limiting YouTube Video Streaming. In *Proceedings of the 2012 USENIX Conference on Annual Technical Conference (USENIX ATC'12)*. Berkeley, CA, USA, 191–196.

[13] D. S. Hochbaum and J. George Shanthikumar. 1990. Convex Separable Optimization is Not Much Harder Than Linear Optimization. *J. ACM* 37, 4 (Oct. 1990), 843–862.

[14] R. Houdaille and S. Gouache. 2012. Shaping HTTP Adaptive Streams for a Better User Experience. In *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*. Chapel Hill, North Carolina, 1–9.

[15] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. 2014. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14)*. Chicago, Illinois, USA, 187–198.

[16] J. Jiang, V. Sekar, and H. Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '12)*. Nice, France, 97–108.

[17] Jan Willem Kleinrouweler, Sergio Cabrero, and Pablo Cesar. 2017. An SDN Architecture for Privacy-Friendly Network-Assisted DASH. *ACM Trans. Multimedia Comput. Commun. Appl.* 13, 3s, Article 44 (June 2017), 22 pages.

[18] Jan Willem Kleinrouweler, Britta Meixner, and Pablo Cesar. 2017. Improving Video Quality in Crowded Networks Using a DANE. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*. ACM, New York, NY, USA, 73–78.

[19] J. Kua, G. Armitage, and P. Branch. 2017. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP. *IEEE Communications Surveys Tutorials* 19, 3 (thirdquarter 2017), 1842–1866.

[20] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. 2014. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *EEE Journal on Selected Areas in Communications* 32, 4 (2014), 719–733.

[21] R.K.P. Mok, E.W.W. Chan, and R.K.C. Chang. 2011. Measuring the quality of experience of HTTP video streaming. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. Dublin Ireland, 485–492.

[22] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang. 2012. QDASH: A QoE-aware DASH System. In *Proceedings of the 3rd Multimedia Systems Conference (MMSys '12)*. Chapel Hill, North Carolina, 11–22.

[23] M. Mu, M. Broadbent, A. Farshad, N. Hart, D. Hutchison, Q. Ni, and N. Race. 2016. A Scalable User Fairness Model for Adaptive Video Streaming Over SDN-Assisted Future Networks. *IEEE Journal on Selected Areas in Communications* 34, 8 (Aug 2016), 2168–2184.

[24] D. M. Nguyen, L. B. Tran, H. T. Le, N. P. Ngoc, and T. C. Thang. 2015. An evaluation of segment duration effects in HTTP adaptive streaming over mobile networks. In *2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*. 248–253.

[25] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2, Article 28 (Oct. 2015), 24 pages.

[26] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2, Article 28 (Oct. 2015), 24 pages.

[27] M.H. Pinson and S. Wolf. 2004. A new standardized method for objectively measuring video quality. *Broadcasting, IEEE Transactions on* 50, 3 (Sept 2004), 312–322.

[28] W. Pu, Z. Zou, and C. W. Chen. 2012. Video adaptation proxy for wireless Dynamic Adaptive Streaming over HTTP. In *Packet Video Workshop (PV), 2012 19th International*. Munich-Garching, Germany, 65–70.

[29] J. J. Quinlan, D. Raca, A. H. Zahran, A. Khalid, K. K. Ramakrishnan, and C. J. Sreenan. 2016. D-LiTE: A platform for evaluating DASH performance over a simulated LTE network. In *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. Rome, Italy, 1–2.

[30] J. J Quinlan, A. H. Zahran, and C. J. Sreenan. 2016. Datasets for AVC (H.264) and HEVC (H.265) for Evaluating Dynamic Adaptive Streaming over HTTP (DASH). In *Proc. of ACM MMsys 2016 (dataset track)*. Klagenfurt, Austria.

[31] A. Seetharam, P. Dutta, V. Arya, J. Kurose, M. Chetlur, and S. Kalyanaraman. 2015. On Managing Quality of Experience of Multiple Video Streams in Wireless Networks. *Mobile Computing, IEEE Transactions on* 14, 3 (March 2015), 619–631.

[32] T. C. Thang, H. T. Le, H. X. Nguyen, A. T. Pham, J. W. Kang, and Y. M. Ro. 2013. Adaptive video streaming over HTTP with dynamic resource estimation. *Journal of Communications and Networks* 15, 6 (Dec 2013), 635–644.

[33] J. De Vriendt, D. De Vleeschauwer, and D. Robinson. 2013. Model for estimating QoE of video delivered using HTTP adaptive streaming. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. Ghent, Belgium, 1288–1293.

[34] Jun Yao, Salil S. Kanhere, Imran Hossain, and Mahbub Hassan. 2011. Empirical Evaluation of HTTP Adaptive Streaming under Vehicular Mobility. In *NETWORKING 2011*, Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 92–105.

[35] F.Z. Yousaf, M. Liebsch, A. Maeder, and S. Schmid. 2013. Mobile CDN enhancements for QoE-improved content delivery in mobile operator networks. *Network, IEEE* 27, 2 (March 2013), 14–21.

[36] Ahmed H. Zahran, Jason J. Quinlan, K. K. Ramakrishnan, and Cormac J. Sreenan. 2017. SAP: Stall-Aware Pacing for Improved DASH Video Experience in Cellular Networks. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 13–26.

[37] A. H. Zahran and C. J. Sreenan. 2016. ARBITER: Adaptive rate-based intelligent HTTP streaming algorithm. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. Seattle, WA, USA, 1–6.