A Fair Share for All: TCP-Inspired Adaptation Logic for QoE Fairness Among Heterogeneous HTTP Adaptive Video Streaming Clients

Michael Seufert, Nikolas Wehner, and Pedro Casas

***Abstract*—This paper presents a novel adaptation logic for HTTP adaptive streaming (HAS), which achieves not only a high quality of experience (QoE) but also high QoE fairness among independent and heterogeneous clients. The algorithm forces video clients to adapt the requested quality level based on the current network conditions and their individual bit rate requirements, such that the overall quality levels selected by all currently active streaming clients are fairly distributed, i.e., they do not diverge too much. The design of the algorithm is inspired by the well-known transmission control protocol (TCP) congestion control, and drives heterogeneous clients to independently converge on similar quality levels without the need for communicating with each other and/or with a centralized controller in the network. By defining quality levels with equal visual quality, and preparing video representations accordingly, the quality level fairness is extended to QoE fairness. In this paper, the design of the TCP-inspired adaptation logic (TCPAL) is described and a simulative performance evaluation is conducted to compare the QoE and QoE fairness of the proposed algorithm with other HAS adaptation logics. TCPAL is evaluated both in scenarios with stable and fluctuating streaming capacity, and the impact of its parameters is explored. The results suggest that TCPAL performs on par with other HAS adaptation logics in terms of QoE and QoE fairness for low link capacities, but significantly improves the QoE fairness for increased link capacity. Moreover, the fairness achieved by TCPAL does not degrade in situations with fluctuating streaming capacity.**

I. Introduction

VIDEO services on the Internet have evolved from offering pure downloads of video files to progressive downloads and streaming, which both describe the concurrent download and playback of media files. Today, streaming services are the most popular and most demanding applications of the Internet due to the high number of requests, high bit rates of the video content and strict real-time requirements of the video playback. Still, the delivered streaming service has to meet the expectations of the end users. To understand and eventually improve Internet services like video streaming, application providers and Internet service providers use the concept of Quality of Experience (QoE) to quantify the subjective experience and satisfaction of their customers with the network and the service. For video streaming, the most severe QoE degradations were the waiting time until the start of the playback (initial delay) and interruptions of the playback (stalling) [1]–[3].

The currently prevailing streaming technology – HTTP Adaptive Streaming (HAS) – allows to mitigate these QoE degradations by offering the possibility to adapt the video bit rate to the network conditions. The goal is to ensure a smooth streaming when end users face throughput fluctuations, e.g., in mobile networks. HAS utilizes standard Web protocols (mostly HTTP over TCP, recently also over QUIC) to promote a simple service implementation and high availability. It is implemented in many commercial solutions and was standardized as MPEG Dynamic Adaptive Streaming over HTTP (DASH) [4].

To enable adaptation of the video streaming to the current network conditions, the HAS server stores the video content encoded in different representations, i.e., in different bit rates. The representations are split into segments (also referred to as chunks) and corresponding segments of different representations contain the same frames, such that the bit rate can be seamlessly switched at each segment boundary. Typically, all segments contain a fixed amount (i.e., 1 to 15 s) of video playback time and can either be extracted at runtime from the single representation file or are stored as separate files on the server. To change the video bit rate, the transmitted video has to be altered, e.g., in terms of resolution, frame rate, or compression, which changes the visual quality of the segment, thereby, introducing an additional impact on QoE [1], [5]–[7].

The adaptation logic at the client is an algorithm that selects which segments to download next from the list of available segments. These decisions usually take into account segment characteristics (e.g., bit rate), current playout statistics (e.g., buffer fill level), and current network conditions (e.g., bandwidth measurements or estimations). By adapting the video bit rate appropriately, the algorithm aims to maximize the QoE for the given network conditions by minimizing initial delay, avoiding stalling, and playing out the video in a high visual quality. Consequently, it is the adaptation logic, which

(Corresponding author: Michael Seufert.)

M. Seufert was with the Digital Insight Lab, Center for Digital Safety and Security, AIT Austrian Institute of Technology GmbH, 1210 Vienna, Austria. He is now with the Chair of Communication Networks, Institute of Computer Science, University of Würzburg, 97074 Würzburg, Germany (e-mail: michael.seufert@uni-wuerzburg.de).

N. Wehner and P. Casas are with the Digital Insight Lab, Center for Digital Safety and Security, AIT Austrian Institute of Technology GmbH, 1210 Vienna, Austria (e-mail: nikolas.wehner@ait.ac.at; pedro.casas@ait.ac.at).

Digital Object Identifier 10.1109/TNSM.2019.2910380

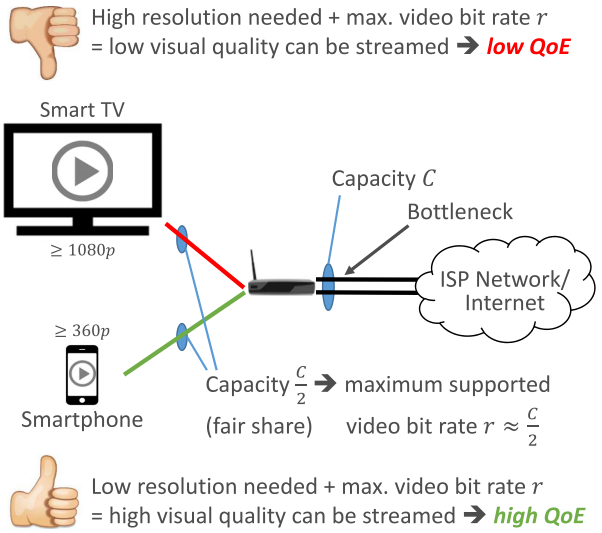

Fig. 1. QoE unfairness as a result of fair capacity allocation.

predominantly influences the network demands of the video streaming and the resulting QoE of the end user.

Network providers want the resulting QoE of the end users in their networks to be high, and they also want that all users have a similar experience, which can be quantified in terms of QoE fairness [8]. When multiple HAS flows are in a network and share a bottleneck link, the link capacity will be equally distributed to all flows by the fairness of the TCP protocol, and thus, also the bit rates of the videos will be similar. However, as the requirements for each streaming flow might be different, e.g., in terms of content complexity or device characteristics, the resulting QoE of the end users might not be fair at all. Figure 1 presents such a situation. At the bottleneck link, the streaming capacity C is equally shared by the two streaming clients, one smart TV and one smartphone. Both streaming clients are allocated the same streaming capacity, which is the fair share $\frac{C}{2}$. Thus, they are limited to request segments with the same maximum supported video bit rate r, which is typically also roughly around $\frac{C}{2}$, although they have heterogeneous requirements in terms of video resolution. The smartphone has a small display and is satisfied streaming a video in low resolution only. Thus, a small bit rate is sufficient to achieve a high visual quality (little compression) and high QoE. However, streaming the same video on a smart TV with the same bit rate will result in a low visual quality (much compression) due to the higher resolution needed for the large TV display, and consequently, a low QoE will result. This shows that, while the capacity allocation and the resulting video bit rate distribution in the network are fair, the QoE distribution is not fair at all. Existing HAS adaptation logics were not able to cope with the problem of QoE fairness for competing HAS flows, such that collaborative and centralized solutions in the network were needed [9]–[12].

This paper presents a novel HAS adaptation logic, which reaches high QoE as well as QoE fairness on shared links with multiple HAS flows without requiring communication between the streaming applications and/or a centralized controller. It is inspired by the well-known Transmission Control Protocol (TCP) congestion control, and drives heterogeneous clients to independently converge on similar quality levels, instead of similar bit rates. When equal quality levels correspond to equal visual quality, which can be easily realized during the content preparation by the video provider, the QoE fairness in the network will be high, i.e., all HAS flows will achieve a similar QoE by using the proposed adaptation logic.

This paper extends previous work [13] by a more precise problem formulation and motivation for needing a QoE-fair HAS adaptation logic. Moreover, the transfer of the concepts behind TCP fairness to achieve QoE fairness is elaborated in more detail. For the performance evaluation of the proposed adaptation logic, additional simulation runs have been generated to improve the validity of the presented results for stable streaming capacities. Furthermore, its performance is compared in four scenarios with fluctuating streaming capacities. Finally, the impact of the parameters of the proposed algorithm is explored.

The remainder of the paper is structured as follows. Section II describes related works on adaptation logics for HAS and QoE fairness. Moreover, it recaps the basic principles of TCP, especially focusing on its congestion control to achieve fairness. Section III presents the design considerations for the novel HAS adaptation logic and shows the implementation of the algorithm in detail. A performance evaluation of the proposed algorithm is conducted based on the simulation framework presented in Section IV. Section V presents the performance of the proposed algorithm, which is compared to existing HAS adaptation logics in terms of QoE and QoE fairness. Finally, Section VI revisits the parameters of the proposed algorithm and explores their impact, and Section VII concludes.

II. Related Work

The Quality of Experience (QoE) of HTTP adaptive streaming (HAS) is a widely investigated research field. The most important research results were reported in [1], while more recent publications confirmed the findings that initial delay, stalling, and quality adaptation are the most dominant QoE factors. While the initial delay has only a small impact on the QoE, stalling, i.e., the interruption of the video playback due to buffer depletion, is considered the worst form of QoE degradation [2], [3]. Moreover, the played out video quality and the time on each quality layer strongly impact the QoE [5]–[7]. These QoE factors will be considered during the performance evaluation of the proposed algorithm.

As subjective QoE studies, which are typically used to assess the QoE, are expensive and time-consuming, QoE models are needed, which can predict the QoE based on objective criteria, e.g., the QoE factors. Recently, such an objective QoE model was standardized as P.1203 [14], whose mode 3 (i.e., complete information is available) will also be used in this paper to obtain QoE scores for HAS sessions. The output of the P.1203 metric is a predicted mean opinion score (MOS) on a 5-point absolute category rating (ACR) scale, ranging from 1 (bad) to 5 (excellent) [15]. The predicted score is shaped by information about the audio and video encoding, as well as by application-layer parameters like the number of stallings, the

length of the stalling events, the initial delay, and the number of quality switches. More details on the implementation of the standard can be found in [14].

QoE factors are not only directly influenced by the network conditions, but also by the adaptation logic of the HAS client. It can be considered as a mechanism for application management, which aims to maximize the user's QoE by requesting appropriate quality levels in order to minimize initial delay, avoid stalling, and download the video with a high visual quality. Network providers do not only want to maximize the QoE of each end user, but also want to achieve a high QoE fairness in their networks. Therefore, they might apply network management or collaborative application and network management. However, as these solutions are not practical and require business agreements between service providers and network providers, the goal of this work is to reach QoE fairness of independent and heterogeneous HAS clients only by application management. Therefore, a new adaptation logic for HAS clients is proposed, which is inspired by TCP fairness and aims to both maximize QoE and QoE fairness.

A. HAS Adaptation Logics

Literature proposes several HAS adaptation logics that focus on different aspects. Some of them are considered in this paper, and will be shortly described in the following. BufferBased [16] solely utilizes the buffer level to decide the next quality level. By dividing the buffer level into several segments and assigning different actions, e.g., increasing/decreasing the quality level, the adaptation logic maps the current buffer level to the segments, and performs the corresponding action. In contrast, ELASTIC [17] applies linear feedback control theory in order to avoid the ON-OFF traffic pattern that occurs when clients estimate their bandwidth shares falsely due to other temporarily inactive clients. KLUDCP [18] selects the quality level based on the disparity between the current buffer level and the desired buffer level, a bandwidth estimate, and the corresponding bit rates. Finally, TRDA [19] is designed for stalling prevention and reduction of the frequency of quality adaptations. Similar to ELASTIC, it is based on previous bandwidth estimates, but introduces a fast start phase, which facilitates a trade-off between the maximization of the video quality and the initial delay.

Reference [20] uses a simulation framework based on the Wi-Fi model of NS-3 to compare several HAS adaptation logics, namely, TRDA/Tobasco, Conventional, PANDA [21], and FESTIVE [22]. They found that TRDA showed the best results with respect to the stalling ratio, the number of quality changes, and the average buffer level, but suffered from the lowest mean video quality. The best video quality was obtained by FESTIVE, which in turn suffered from frequent quality switches and, for too many clients, stalling. Reference [23] compares ten different adaptation logics and considers QoE metrics, as well as network utilization, stability of the adaptation, and the mean bit rates requested by the algorithms. The results revealed no dominating adaptation logic, but even simple adaptation logics performed reasonably well.

B. QoE Fairness

QoE fairness [8] refers to a fair distribution of the QoE scores of all end users in a network. This means that the QoE scores should be very similar and not diverge too much. Reference [8] found that the prevailing fairness measure, Jain's fairness index [24], was not particularly well suited to quantify QoE fairness. The reason is that Jain's fairness index requires ratio scales with a defined zero point, but QoE is rather measured on interval scales, e.g., the 5-point ACR scale, which is typically used to obtain MOS values.

Reference [8] defined a new measure for QoE fairness F with additional scale and metric independence for computing the fairness of QoE scores. Thereby, the fairness measure F is computed as a linear transformation of the standard deviation of the QoE scores to the interval [0;1]. This is achieved by setting the standard deviation σ in relation to the maximum possible standard deviation σ_{max}, which is the difference between the highest possible QoE score H and the lowest possible QoE score L : $F = 1 - \frac{\sigma}{\sigma_{max}} = 1 - \frac{2\sigma}{H-L}$.

Consequently, $F = 1$ indicates that $\sigma = 0$ and all QoE scores are the same, i.e., a perfect QoE fairness. In contrast, $F = 0$ represents the most unfair situation, in which 50% of the users have a QoE score of H and 50% have a QoE score of L. In this paper, F is used to compute the fairness of the selected quality levels and the resulting QoE score.

C. QoE Fairness With Additional Network Management

As current HAS adaptation logics of independent and heterogeneous clients cannot reach QoE fairness in the network, the research community tackled this issue with the help of network management. Thereby, the network conditions are influenced during the video streaming depending on the current QoE, e.g., by prioritization or dedicated bandwidth allocation for video flows [25], [26]. However, this requires the estimation of video requirements and the monitoring of QoE in the network, which has become a difficult and cumbersome task since video traffic is encrypted. Another approach is combined network and application management based on the collaboration and communication between the HAS clients and the network, e.g., by using a centralized controller in the network. In the following, several works are presented.

Reference [27] uses SDN to dynamically allocate the network resources for each client based on its expected QoE. Reference [28] modifies the HAS index files, i.e., the list of available segments, in the network. Reference [29] proposes bandwidth reservation for HAS flows and signals clients which quality levels to request. Reference [9] tries to achieve QoE fairness for multiple clients by using information about network conditions. Reference [10] presents a collaborative traffic management system, which considers QoE fairness in case of encrypted HAS. Recently, SAND standardization efforts were started to exchange information measured at servers and network elements to improve the delivery, and to send quality-related information to clients to improve the reception [11], [12].

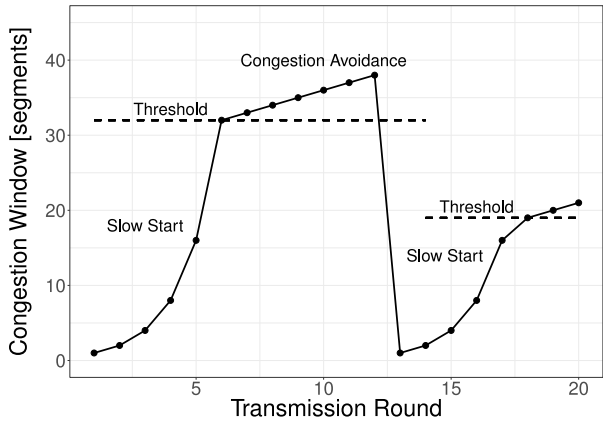

Fig. 2. TCP congestion control.

D. TCP Fairness

The Transmission Control Protocol (TCP) [30] is one of the most widely used protocols on the Internet. TCP is connection-oriented and ensures a guaranteed transmission of undisturbed data. It provides a principle of fair bandwidth shares among multiple clients over a shared network link. This is achieved by utilizing flow control and congestion control. As TCP flow control is rather focused on avoiding to overload the receiver, it was irrelevant for the design of the adaptation logic, and thus, is not discussed here.

The basic TCP congestion control [31] is depicted in Figure 2. It is composed of an exponential growth of the congestion window, the so called slow start phase, followed by a linear growth of the congestion window, the congestion avoidance phase. The congestion window indicates the number of unacknowledged TCP segments, which the sender can send within one transmission round. The duration of a transmission round is characterized by the round trip time. During the slow start phase, the congestion window is increased by one for each received acknowledgment, resulting in an exponential growth of the window, as the number of segments doubles each round.

A threshold specifies the end of the slow start phase and hence the beginning of the congestion avoidance phase. In the congestion avoidance phase, the congestion window is increased by one per transmission round. When a segment is lost, congestion occurs. As a consequence, standard TCP Tahoe decreases the congestion window size to 1 and sets the slow start threshold to half of the previously active congestion window. This is followed by a new slow start phase. Several variations of standard TCP exist, which modify this basic procedure. For example, in TCP Reno the congestion window size is not reduced to 1 on loss, but only to half of the previous window size and a congestion avoidance phase directly follows instead of a slow start phase (fast recovery).

As all clients can detect congestion rather simultaneously by observing their own loss, and react similarly by reducing the own sending rate, this can be considered an altruistic behavior of clients. It happens that the bandwidth of the shared link can be fairly shared among all clients. The proposed HAS adaptation logic was inspired by the TCP congestion control, and uses similar design considerations to achieve a fair quality level for all HAS clients. Note that practical problems might arise with HAS flows in a shared TCP network due to the on-/off-phases of the download [1], [32]. However, these issues are not considered in this work, as it only focuses on investigating the potential of a TCP-inspired adaptation logic.

III. TCP-Inspired Adaptation Logic

This section introduces the novel HAS adaptation logic, which reaches QoE fairness on shared links with multiple independent and heterogeneous HAS clients.

A. Design Considerations

A prerequisite for achieving similar QoE on heterogeneous HAS clients is that each client can request the same quality levels, which is not the case when they can only request different bit rate levels. Typically, different videos with the same bit rate have a different visual quality, e.g., due to different content complexity, resolution, or compression. Therefore, quality levels are introduced. Without loss of generality, the quality levels are ordered from 1 (lowest quality) to l_{max} (highest quality) and the visual quality contained in different segments of the same quality level should be the same. In practice, this means that instead of encoding the different representations of a HAS video file with certain (target) bit rate levels, the target for encoding should be a metric for the visual quality (e.g., SSIM or VQM), such that different segments of the same quality level have the same visual quality. Video providers also have to ensure that the quality levels are consistent for different contents or resolutions.

The main concept for reaching QoE fairness among independent and heterogeneous HAS clients is based on the idea that all clients need to obtain the same information from observing the network. This information can then be used by each client together with information about its individual characteristics to maximize its own QoE in a synchronous and consistent, yet altruistic way. In this case, the term "altruistic" means that each client is aware that the network is shared and does not egoistically increase its own quality level in situations when other clients can not. In the following, the four main design considerations are described that are used to obtain "altruistic" behavior.

1) Limitation of Buffer Filling Rate: One problem of competing heterogeneous clients is that clients with small bit rates can download segments and fill the buffer more quickly than clients with large bit rates. Consequently, with most adaptation logics, they can much earlier attempt to increase the quality level, which can result in an unfair QoE distribution. In order to keep the clients and their buffer more synchronized, a limitation of the buffer filling rate is introduced. After a desired buffer level b_d is reached, the buffer will be kept at a constant level in terms of buffered segments. Therefore, after a segment is downloaded, the clients have to go idle for the contained playtime of a segment τ minus the download time of the segment T, i.e., $t_i = \tau - T$. Thus, a HAS client with sufficiently filled buffer will request segments regularly in intervals

of τ, which results in a buffer filling rate of 1 that is equal to the playback rate (unit rate constraint). Thereby, the buffer is effectively kept constant at the desired buffer level b_d, which also avoids excessive amounts of unnecessarily downloaded data in case of video abortion.

If the buffer is below the desired level b_d, the buffer has to be increased. Therefore, the buffer filling rate can be increased to 2. This is achieved by allowing HAS clients to go idle only for $t_i = \frac{\tau}{2} - T$ after a segment download, which causes them to request segments in intervals of $\frac{\tau}{2}$. As the buffer filling rate is double the playout rate, the buffer will increase linearly with rate 1. If the video is initially loading or stalling, the buffer has to be increased as fast as possible to minimize the initial delay or stalling time. Therefore, in this case, the HAS clients do not need to go idle after a segment download, but can request segments back to back until the playback can be started.

2) Capacity Estimation and Upper Bounds on Quality Level: After the timing of segment requests is more synchronized among all clients, the requested quality levels have to be synchronized next. Therefore, a common information about the network is needed, which can be observed by each client individually, and can be used to come to an agreement on the quality level. Such information is the maximum capacity $\hat{C}_{max}$, which is observed by the client. For each of k active HAS clients, its observed $\hat{C}_{max}$ should be similar and approximate $C_{max} = \frac{C^*}{n}$, which is the fair bandwidth share of the link capacity C^* as a result of TCP fairness, when n flows (k video flows and $n-k$ other flows) share the link. After each segment download, a capacity estimate $\hat{C} = \frac{s}{T}$ can be computed by dividing the size s of the requested segment by the download time T, and the observed maximum capacity $\hat{C}_{max}$ can be updated. All adaptation decisions will be based on the observed maximum capacity. Therefore, a fictitious reference video v' is considered with a maximum needed bit rate r'_{max} equal to the maximum capacity, i.e., $r'_{max} = \hat{C}_{max}$. This reference video would just be able to maintain its buffer when its throughput is equal to the maximum capacity, i.e., the download time of a segment is equal to the playtime contained in a segment. To account for bit rate variations, the maximum needed bandwidth is computed as the mean bit rate of all segments of the highest quality level $\bar{r}_{max}$ plus α-times the corresponding standard deviation $\sigma_{r_{max}}$, i.e., $r_{max} = \bar{r}_{max} + \alpha \cdot \sigma_{r_{max}}, \alpha \geq 0$.

If the throughput of a video decreases to a share $0 < q \leq 1$ of the maximum capacity, i.e., $\hat{C} = q \cdot \hat{C}_{max}$, the reference video would have to decrease its quality level to $q \cdot l_{max} \coloneqq l_u$, and thus, to be fair, also all other videos should reduce their quality levels to l_u. The computation of l_u utilizes a parameter $\beta \leq 1$ to relax the normalization with the individually observed $\hat{C}_{max}$, i.e., $l_u = \lceil \frac{\hat{C}}{\beta \cdot \hat{C}_{max}} \cdot l_{max} \rceil$. This means that the maximum quality level can still be requested when $\hat{C} \geq \beta \cdot \hat{C}_{max}$. Thus, by obtaining the same (or similar) $\hat{C}_{max}$ and $\hat{C}$, competing clients can individually find the same (or a similar) upper bound l_u for the quality level of requested segments based on the current network conditions.

3) Slow Start and Congestion Avoidance Phase: Similar to TCP, the quality level of HAS clients should be increased fast until the currently supported quality level is reached, while at the same time fairness among the HAS clients should be maintained. Therefore, the proposed algorithm was designed on the idea of transferring the concepts behind TCP fairness to achieve QoE fairness. As a first step, parameters of the TCP congestion control had to be mapped to the adaptation of quality levels in HAS, which is depicted in Table I.

TABLE I
MAPPING OF TCP VARIABLES AND HAS VARIABLES FOR TRANSFERRING TCP CONGESTION CONTROL INTO TCPAL SLOW START AND CONGESTION AVOIDANCE PHASE

TCP Variables	⟷	HAS Variables
Transmission round	⟷	Segment request
Round trip time	⟷	Segment download time T + idle time t_i
Congestion window	⟷	Quality level of next segment l_n
Slow start threshold	⟷	Half of maximum quality level l_u
Congestion	⟷	Decreasing buffer B

An exponential growth of the current quality level l_c was implemented similar to the TCP slow start phase. A slow start phase is entered always after the playback of the video started, i.e., after initial delay or stalling. It allows to double the requested quality level after each downloaded segment, i.e., the new quality level $l_n = 2 \cdot l_c$. After the current segment has exceeded half of the currently allowed maximum level l_u, and thus, another doubling is not possible, the congestion avoidance phase follows.

The congestion avoidance phase approaches and aims to reach the maximum supported quality level. In this phase, the quality level can be increased by 1 after each γ segments. Therefore, each client remembers its last quality change t_{lc}, and if more time than $\gamma \cdot \tau$ has passed, the new quality level $l_n = l_c + 1$ can be requested. Note that the upper bound l_u overrides increases of the quality levels that would have been triggered by the slow start or congestion avoidance phase.

4) Congestion Detection and Compensation of Bit Rate Requirements: A "congestion" with the current quality level l_c occurs when the download time T of a segment is larger than the contained playtime τ, and thus, the buffer decreases. In this case, l_c should be decreased in order to avoid further depletion of the buffer, which could eventually cause stalling. To consistently act among all HAS clients, they will compute the same new quality level $l_n = \delta \cdot l_c$, being a fixed share $\delta < 1$ of the current quality level on which the congestion was detected. Moreover, a hard threshold for the buffer level b_l is introduced, which also triggers a congestion. If the buffer drops below the low threshold b_l, the buffer should increase, and thus, the quality level has to be decreased.

As videos with larger bit rates are more likely to face congestion, the segment download times T have to be compensated according to the bit rate requirements. Therefore, again the fictitious reference video v' is considered. It is reminded that to maintain its buffer, it should be allowed to fully use the download time $T' = \tau$ to download a segment of the highest quality level under the maximum network conditions $\hat{C}_{max}$. To have a comparable and fair congestion threshold, a video with a smaller maximum needed bit rate r_{max} should have to download the video with a proportionally shorter download time. Thus, the bit rate compensation can be done via

the compensated download time T_c, which can be computed from the actual download time T as $T_c = T \cdot \frac{\hat{C}_{max}}{r_{max}}$ and compared to the segment playtime τ to detect congestion. Again $r_{max} = \bar{r}_{max} + \alpha \cdot \sigma_{r_{max}}$ is the individually needed maximum bit rate of each HAS client.

As congestion is always an indicator that the requested quality level cannot be maintained, the assumptions on the network conditions have to be reevaluated. To account for new clients, which reduce the fair bandwidth share $C_{max} = \frac{C^*}{n}$, the assumed maximum capacity $\hat{C}_{max}$ is halved. This is a conservative reaction, as for each additional client (*n* increases by 1), the new C_{max} is at least half of the previous C_{max}. Still, $\hat{C}_{max}$ can be updated with subsequent capacity estimations.

B. Algorithm Description

The algorithm is called TCP-inspired Adaptation Logic (TCPAL) and is shown in Algorithm 1. TCPAL has ten predefined parameters on which all HAS clients have to agree, namely, the segment playtime τ, maximum quality level l_{max}, the initial delay threshold b_i, the stalling threshold b_s, the low buffer threshold b_l, the desired buffer level b_d, the considered bit rate variation α, the normalization relaxation β, the linear growth regulation γ, and the congestion decrease factor δ. Additionally, each HAS client has to maintain and accordingly update the maximum needed bit rate r_{max}, the current quality level l_c, the current buffer level B, the time since the last quality change t_{lc}, the currently observed maximum capacity $\hat{C}_{max}$, and the Boolean variables isInitialDelay, isStalling, and slowStart. These client variables are initialized as shown in "Client Vars" and they are updated either by the algorithm when indicated by ":=" (in contrast to "=", which represents an allocation to a temporal variable), or externally (reduction of B during playback, $B == 0$ (playback is stalling) $\Rightarrow$ isStalling := True), or never (r_{max}).

After a video request, the HAS clients will immediately download the first segment in the lowest quality level, i.e., $l_c = 1, t_i = 0$, and subsequently call TCPAL whenever a segment is completely downloaded. Thereby, the only changing inputs are the size of the last downloaded segment *s* and the download time of the last segment *T*. TCPAL will output the quality level of the next segment l_n and the idle time t_i, i.e., the time until the next segment with quality level l_n can be requested. The presented design considerations can be found in Algorithm 1 as follows: The limitation of the buffer filling rate is implemented in Lines 2-15. The capacity estimation and upper bound to the available quality levels can be seen in Lines 16-18. The congestion detection in Lines 19-20 is followed by the corresponding consequences of congestion in Lines 21-23. The slow start phase corresponds to Lines 25-28, and the congestion avoidance phase to Lines 29-35.

Algorithm 1: TCPAL

Parameters: $\tau, l_{max}, b_i, b_s, b_l, b_d, \alpha, \beta, \gamma, \delta$
Client Vars: $r_{max} := \bar{r}_{max} + \alpha \cdot \sigma_{r_{max}}, l_c := 1, B := 0,$ $\hat{C}_{max} := 0, t_{lc} := t_{NOW}$, isInitialDelay := True, isStalling := True, slowStart := True
Input: s, T
Output: l_n, t_i

1 $B := B + \tau$;
2 **if** *isStalling == True* **then**
3 $\quad t_i = 0$;
4 $\quad$ slowStart := True;
5 $\quad$ **if** *(isInitialDelay == True and* $B \geq b_i$*) or (isInitialDelay == False and* $B \geq b_s$*)* **then**
6 $\quad\quad$ playVideo();
7 $\quad\quad$ isInitialDelay := False;
8 $\quad\quad$ isStalling := False;
9 **else**
10 $\quad$ **if** $B \geq b_d$ **then**
11 $\quad\quad t_i = \max(\tau - T, 0)$;
12 $\quad$ **else**
13 $\quad\quad t_i = \max(\frac{\tau}{2} - T, 0)$;
14 $\quad$ **end**
15 **end**
16 $\hat{C} = \frac{s}{T}$;
17 $\hat{C}_{max} := \max(\hat{C}, \hat{C}_{max})$;
18 $l_u = \max(\min(\lceil \frac{\hat{C}}{\beta \cdot \hat{C}_{max}} \cdot l_{max} \rceil, l_{max}), 1)$;
19 $T_c = T \cdot \frac{\hat{C}_{max}}{r_{max}}$;
20 **if** $T > \tau$ *or* $T_c > \tau$ *or* $B - t_i < b_l$ **then**
21 $\quad l_n = \max(\min(\delta \cdot l_c, l_u), 1)$;
22 $\quad \hat{C}_{max} := \frac{\hat{C}_{max}}{2}$;
23 $\quad$ slowStart := False;
24 **else**
25 $\quad$ **if** *slowStart* **then**
26 $\quad\quad l_n = \min(2 \cdot l_c, l_u)$;
27 $\quad\quad$ **if** $l_n > \frac{l_u}{2}$ **then**
28 $\quad\quad\quad$ slowStart := False;
29 $\quad$ **else**
30 $\quad\quad$ **if** $t_{NOW} - t_{lc} > \gamma \cdot \tau$ **then**
31 $\quad\quad\quad l_n = \min(l_c + 1, l_u)$;
32 $\quad\quad$ **else**
33 $\quad\quad\quad l_n = \min(l_c, l_u)$;
34 $\quad\quad$ **end**
35 $\quad$ **end**
36 **end**
37 **if** l_n *!=* l_c **then**
38 $\quad t_{lc} := t_{NOW}$;
39 $l_c := l_n$;
40 **return** l_n, t_i

IV. Simulation Framework

This section describes the simulation framework for the performance evaluation of the proposed adaptation logic. While a generic network simulation tool could potentially also have been used for the performance evaluation, a custom simulation framework was developed to have full control over all aspects of the simulated system, i.e., the bottleneck link and the streaming clients together with the adaptation logic.

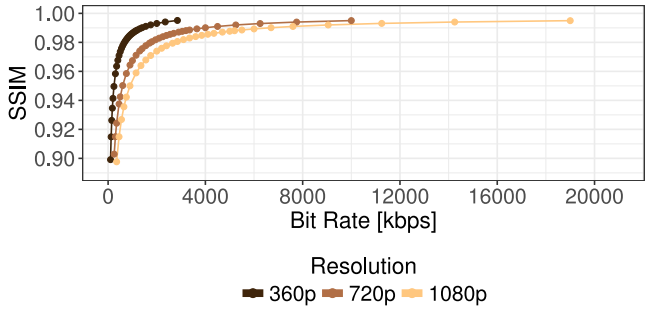

Fig. 3. Target SSIM values for quality metrics and corresponding bit rates for different resolutions.

In [26], a Java discrete-event simulation for video streaming and Web page downloads over a bottleneck link with fixed capacity was presented. In this work, the simulation was extended to provide HAS capabilities, to integrate different HAS adaptation logics, and to allow fluctuating link capacities.

The simulation allocates the available link capacity as follows. Each of the active application flows receives an equal share of the current capacity, i.e., the normal TCP-fair share of bandwidth. Whenever a client enters or leaves the system, or the download of a client becomes idle or resumes, the available capacity is redistributed, such that every downloading client is allocated the updated fair bandwidth share. The simulation keeps track of the download and playback of all videos, which can be used to compute several key performance indicators, e.g., the initial delay, stalling, and played out video qualities. Each simulation run is aborted after 550 s and all statistics are stored in log files, which are later used for the evaluations.

For a more realistic simulation, real video chunks were integrated into the simulation. Therefore, the video "Big Buck Bunny" [33] was split into segments with a playback length of four seconds and each segment was encoded with constant bit rate encoding with FFmpeg. In order to obtain quality levels with an equal visual quality, the well-known full-reference metric SSIM (structural similarity) [34] was utilized. 32 target SSIM values were selected as quality levels, ranging from 0.900 (lowest quality level) to 0.995 (highest quality level). Since there are currently no tools that are capable of encoding videos based on a passed target SSIM, the encoding process was performed manually using the method of trial and error, i.e., the encoding bit rate of the segment was varied until the calculated mean SSIM matched the target SSIM. To account for different requirements of HAS clients, e.g., in terms of display characteristics, the video was encoded in three different resolutions (360p, 720p, 1080p), and each client was assigned one of the resolutions at the start of a simulation run.

The extracted video representations are depicted in Figure 3, where the y-axis represents the target SSIM values, the x-axis describes the required encoding bit rate in order to obtain the corresponding SSIM value, and each resolution is colored differently. The figure shows that the HD video content requires a much higher bit rate for a similar SSIM compared to the other video sequences.

V. Performance Evaluation

In the following, the performance of the proposed TCP-inspired adaptation logic (TCPAL) is investigated in detail. Therefore, the parameters were set to $\tau = 4$s, $l_{max} = 32$, $b_i = 12$s, $b_s = 4$s, $b_l = 8$s, $b_d = 16$s, $\alpha = 1, \beta = 0.9, \gamma = 2$, and $\delta = 0.75$. Note that the impact of these initial choices for the TCPAL parameters α, β, γ, and δ is revisited in Section VI.

A. Results for Stable Streaming Capacity

First, the performance of TCPAL is evaluated in scenarios with a stable streaming capacity, i.e., the bottleneck link capacity is constant and the link is only used by HAS clients.

Figure 4 studies the behavior of HAS clients using TCPAL qualitatively in exemplary simulation runs with link capacity 7 Mbps. In Figure 4a, there is only one HAS client on the link, streaming a resolution of 720p. The x-axis depicts the time of the simulation run, while the y-axis shows the requested quality level. In the beginning, the slow start phase can be seen, in which the quality level increases exponentially. The congestion avoidance phase follows, in which the quality level is increased linearly until congestion is detected and the quality level drops to 75% (δ) of the previous level. It can be seen that TCPAL repeatedly approaches the maximum supported quality level, which is 30 given the link capacity of 7 Mbps (see Figure 3), and then drops due to congestion.

Figure 4b shows a situation, in which six HAS clients start simultaneously to request the 720p video. It can be seen that all clients act quite synchronous during both slow start and congestion avoidance phase. It can be seen that the maximum supported quality levels are much smaller, because the link capacity is equally shared among the six HAS clients. Figure 4c shows the same situation, but now the six clients request different video resolutions. Two clients request 360p, two clients 720p, and two clients 1080p. This leads to heterogeneous bit rate requirements in the network. It can be seen that the clients are at different quality levels at the end of the slow start phase. However, during the congestion avoidance phase, the clients bring their quality levels to a similar range.

Finally, Figure 4d shows the most challenging scenario, in which six diverse HAS clients ($2 \times$ 360p, $2 \times$ 720p, $2 \times$ 1080p) start to request their videos at a uniformly random time in the first 100 s. Thus, the HAS clients are out of sync in this scenario, and the available bandwidth of each client will change when a new HAS client starts to request his video. Nevertheless, it can be seen that the quality levels requested by TCPAL are still quite similar. Later, this scenario will be further investigated to compare the performance of TCPAL to other adaptation logics in terms of QoE and QoE fairness.

To understand the shortcomings of other HAS adaptation logics, Figure 5 shows the behavior of other HAS adaptation logics in exemplary simulation runs of the same scenario of Figure 4d. BufferBased (Figure 5a) increases the quality level quite synchronous for all HAS clients but very slowly. Also TRDA (Figure 5d) shows a linear increase, but as it increases one quality level per segment, high quality levels can be reached much faster. In contrast, ELASTIC (Figure 5b) and KLUDCP (Figure 5c) almost immediately jump to very high

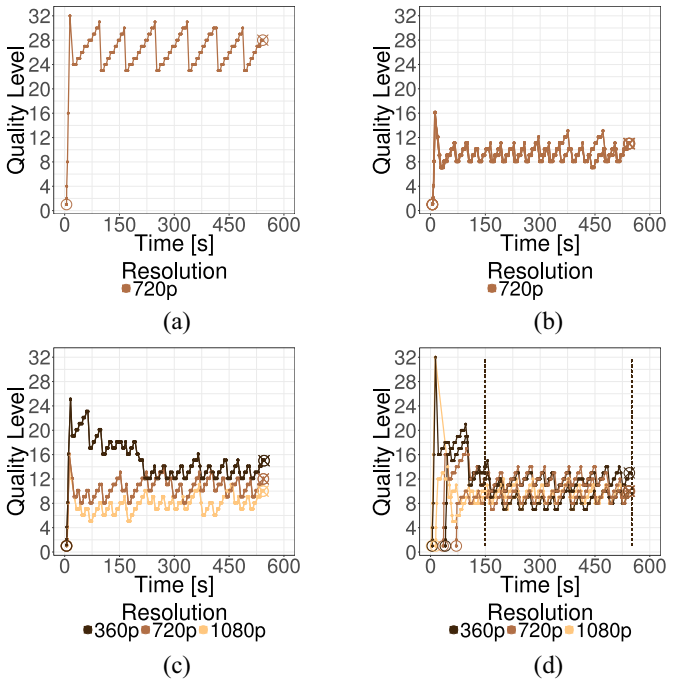

Fig. 4. Performance of TCPAL in different scenarios with link capacity 7 Mbps. (a) Single HAS client with 720p. (b) Six HAS clients with 720p and simultaneous start. (c) Six HAS (2 × 360p, 2 × 720p, 2 × 1080p) clients with simultaneous start. (d) Six diverse HAS (2 × 360p, 2 × 720p, 2 × 1080p) clients with randomized start.

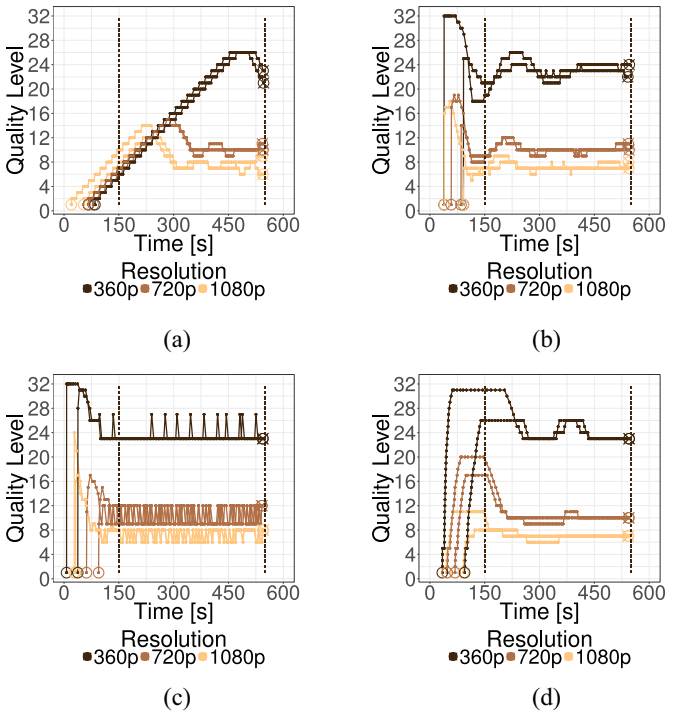

Fig. 5. Performance of other HAS adaptation logics for six HAS (2 × 360p, 2 × 720p, 2 × 1080p) clients with randomized start and link capacity 7 Mbps (see Figure 4d). (a) BufferBased. (b) ELASTIC. (c) KLUDCP. (d) TRDA.

quality levels at the start of the video. While the behavior of each adaptation logic is different (e.g., high oscillations of the quality level for KLUDCP, but inert adaptation for ELASTIC or TRDA), all adaptation logics converge to different quality levels for different resolutions. This means, they cannot achieve fair adaptation for heterogeneous clients with different bit rate requirements. The quality level, and thus, the QoE of clients with high video bit rates (e.g., 1080p) is much lower than for clients with low bit rate requirements (e.g., 360p).

In the following, the quantitative performance evaluation are presented. Therefore, the reference scenario of Figure 4d was used, i.e., six clients (2 × 360p, 2 × 720p, 2 × 1080p) can randomly (uniformly distributed) start a video in the first 100 s of a simulation run. Three different link capacities are used, namely, the already presented capacity of 7 Mbps, for which the six clients have to find appropriate intermediate quality levels, a low link capacity of 2.4 Mbps, and a high link capcity of 70 Mbps. In case the six clients share a link capacity of 2.4 Mbps, the clients can only support low quality levels and need to avoid stalling. In case of 70 Mbps, all six clients should be able to reach very high quality levels. Note that fluctuations of the link capacity are not investigated. However, the increasing number of HAS clients and their oscillating download and idle phases change the fair bandwidth share of all active HAS client, which can be considered a kind of fluctuation. 50 simulation runs were conducted for each adaptation logic and link capacity. Each simulation run is aborted after 550 s, and the performance evaluation (except for initial delay) considers only the steady state phase from 150-550 s (dashed lines in Figures 4d and 5).

Figure 6 shows the QoE results for the different link capacities on the x-axes. Each bar represents a different adaptation logic, see the legend for a mapping of colors and algorithms. The y-axes indicate the mean of the considered performance metric, i.e., initial delay (Figure 6a), total stalling time (Figure 6b), time-weighted average quality level (TWAQL, Figure 6c), and the predicted MOS from P.1203 (Figure 6d), and the corresponding 95% confidence intervals are shown on top.

Figure 6a shows the results for initial delay, for which a shorter initial delay is better in terms of QoE. It can be seen that TCPAL (yellow) reaches low initial delays comparable to TRDA (orange) for all link capacities. Only BufferBased (black) starts the playback of the streamed video faster. The results for stalling are presented in Figure 6b, where again shorter stalling is to be preferred for achieving a better QoE. TCPAL cannot completely avoid stalling for the lowest link capacity, however, the mean total stalling time is short. For higher link capacities stalling is not an issue for TCPAL.

Figure 6c displays the time-weighted average quality level (TWAQL), i.e., each quality level was weighted according to the time for which it was played out. As a higher visual quality level results in a higher QoE, here, in contrast to initial delay and stalling, a higher TWAQL corresponds to a better QoE. It can be seen that the TCPAL adaptation logic requests on average slightly lower quality levels than other adaptation logics for higher link capacity. This comes by design as TCPAL shows an altruistic behavior and restricts itself from requesting too high quality levels. Additionally, the congestion avoidance phase, in which the maximum supported quality

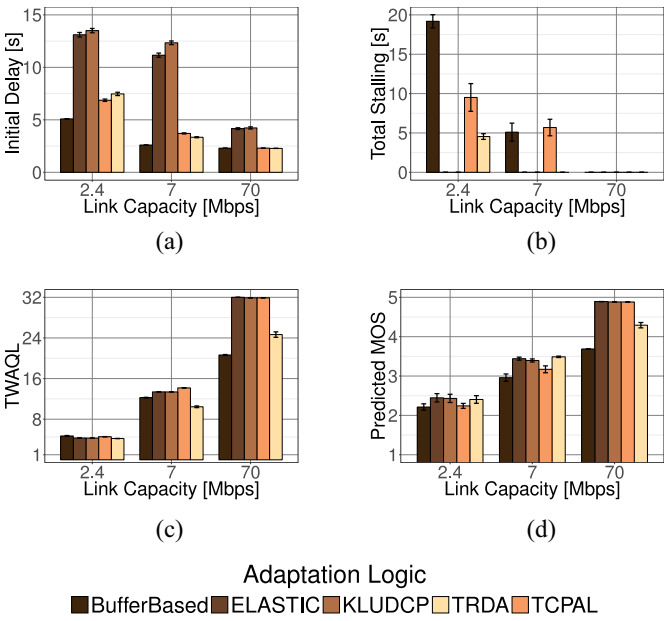

Fig. 6. Results on QoE factors and QoE score (stable streaming capacity). (a) Initial delay. (b) Total stalling time. (c) Time-weighted average quality level (TWAQL). (d) Predicted MOS from P.1203.

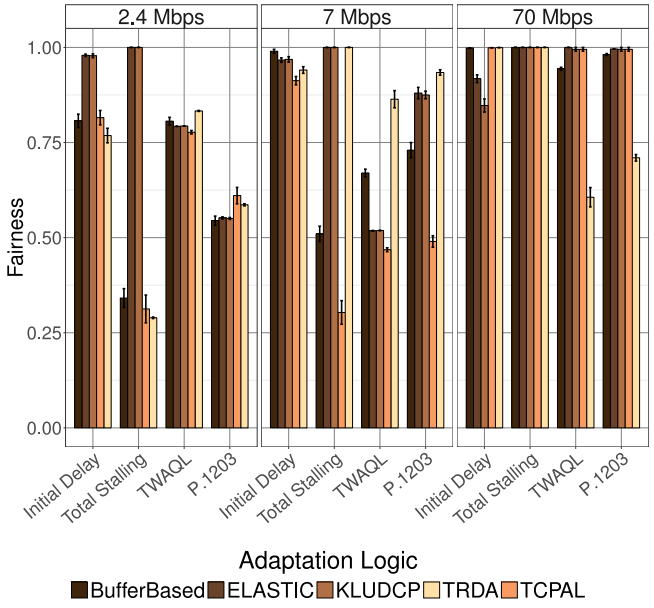

Fig. 7. Results on QoE fairness (stable streaming capacity).

level is approached linearly and the quality level is reduced on congestion, leads to an average quality level slightly below the maximum supported quality level. This propagates to the MOS prediction based on P.1203 in Figure 6d. Here, the output is on the actual MOS scale from 1 (bad) to 5 (excellent), which is the usual metric for describing the QoE, and thus, a higher MOS directly refers to a better QoE. It can be seen that ELASTIC (dark brown) and KLUDCP (light brown) can achieve the highest MOS for all link capacities. Nevertheless, TCPAL is on a par with P.1203 scores of 2.4 for 2.4 Mbps and 3.5 for 7 Mbps. Although it performs slightly worse than ELASTIC, KLUDPC, and TRDA for 70 Mbps, it still achieves a high score of 4.3 on the MOS scale.

Finally, Figure 7 shows the corresponding fairness of the QoE metrics. While Jain's fairness index [24] was used for initial delay and stalling (unlimited metrics on ratio scale), the F metric [8] was used for time-weighted average quality level and the predicted MOS scores. The desirable fairness score is 1, which means a perfect fairness of the underlying metric for all six HAS clients. The figure shows the fairness scores for all QoE metrics of Figure 6 to highlight all aspects of the trade-offs between QoE and QoE fairness. However, when it comes to a final evaluation of the algorithms in terms of QoE and QoE fairness, the final P.1203-based QoE score in Figure 6d and the corresponding P.1203 column in Figure 7 should be considered, as the computation of P.1203 takes the other QoE metrics (initial delay, stalling, visual quality level) into account.

In case of the low link capacity of 2.4 Mbps, clients with high resolutions struggle with stalling and very low quality levels, while clients with low bit rate requirements can maintain a smooth streaming on low quality levels. Thus, the resulting QoE is very different, which can be seen in the low fairness scores. Also for TCPAL, only the congestion detection via the actual download time works, and thus, clients with low bit rate requirements manage to download segments with a higher quality level as can be seen in Figure 8a. As this is the same problem that other HAS adaptation logics face, the fairness scores of TCPAL are on the same level in this scenario.

For 7 Mbps, stalling should be avoided by all adaptation logics. Still, the fairness of the other adaptation logics is not very high as the requested quality levels highly diverge for different bit rate requirements (see Figure 5). In this scenario, TCPAL manages in all simulation runs to reach a significantly higher fairness in terms of time-weighted average quality level (TWAQL) and predicted MOS (P.1203). Together with the comparably high overall QoE, this shows that the design considerations of TCPAL were useful and allow a good and fair streaming of heterogeneous and independent HAS clients.

For the high link capacity of 70 Mbps, initial delays are short, no stalling occurs and a constant high quality level can be maintained by the other adaptation logics. By design, the quality level of TCPAL oscillates in the congestion avoidance phase due to the strict definition of congestion, which can be seen in Figure 8b. This should not negatively affect the QoE fairness, however, it can be seen that the requested quality levels additionally diverge. The reason for this problem is that the download times of segments are very small, and thus, the observed maximum link capacity might be different for different HAS clients depending on how many other clients are downloading or idling, see the on-/off-problem of HAS, e.g., [1], [32]. This leads to self-restrictions of only some clients based on congestion avoidance. Thus, the requested quality levels are different, which results in reduced fairness for time-weighted average quality level (TWAQL), and also for predicted MOS (P.1203). To improve the performance of TCPAL in this case, the robustness of the estimation of the maximum link capacity would have to be improved. However, for different networks, different approaches might have to be implemented, see [35], [36]. Moreover, similar to [37], TCPAL

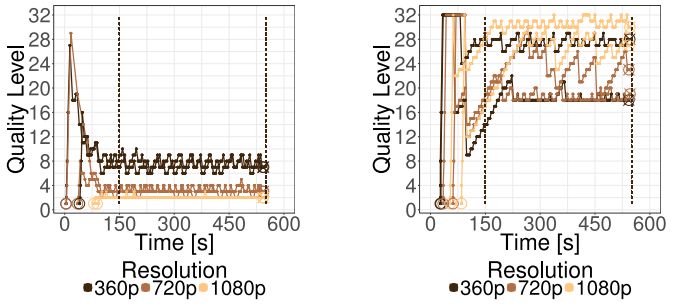

Fig. 8. Performance of TCPAL for six HAS (2 × 360p, 2 × 720p, 2 × 1080p) clients with randomized start and smaller or larger link capacity (see Figure 4d). (a) 2.4 Mbps. (b) 70 Mbps.

parameters could be dynamically adjusted according to the network conditions to improve its performance, e.g., based on machine learning.

B. Results for Fluctuating Streaming Capacity

Next, the performance of TCPAL is evaluated in scenarios with fluctuating streaming capacity. This fluctuation can be caused by varying channel conditions on the mobile link or by cross-traffic, i.e., traffic of other applications on the same bottleneck link.

The same simulation setup is used for the performance evaluation, but the framework was adjusted to allow for possible changes in the link capacity every 1 s. Four fluctuation patterns are investigated, which include independent fluctuations. The patterns result in a highly volatile streaming capacity and can be considered as worst-case scenarios. The four fluctuation patterns are based on the same mean streaming capacities $\bar{C}$ of 2.4 Mbps, 7 Mbps, and 70 Mbps, and the standard deviation was controlled to $\frac{1}{2}\bar{C}$, i.e., half of the mean capacity:

- Alternating (ALT): The ALT pattern switches the link capacity every second from $\frac{1}{2}\bar{C}$ to $\frac{3}{2}\bar{C}$, and vice versa.
- Uniform (UNI): The UNI pattern selects every second a link capacity uniformly, i.e., with probability $\frac{1}{3}$, from three values, namely, $(1-\frac{\sqrt{6}}{4})\cdot\bar{C}$, $\bar{C}$, and $(1+\frac{\sqrt{6}}{4})\cdot\bar{C}$.
- Normal (NOR): The NOR pattern draws every second a normally distributed fluctuation with mean 0 and standard deviation $\frac{1}{2}\bar{C}$, and adds this fluctuation to $\bar{C}$.
- Exponential (EXP): The EXP pattern draws every second an exponentially distributed fluctuation with mean $\frac{1}{2}\bar{C}$, and adds this fluctuation to $\frac{1}{2}\bar{C}$.

Note again, that for all four patterns, the mean link capacity is $\bar{C}$ and the standard deviation is $\frac{1}{2}\bar{C}$.

In the following, the results for the four fluctuation patterns are compared to the results for the stable streaming capacity, i.e., the constant link capacity (CON). This time 20 simulation runs were conducted for each streaming capacity and fluctuation pattern. Figure 9 shows bar plots for the QoE metrics. Each bar represents a different fluctuation pattern, see the legend for a mapping of colors and fluctuation patterns. Again, the y-axes indicate the mean of the considered performance metric, i.e., initial delay (Figure 9a), total stalling time (Figure 9b), time-weighted average quality level (TWAQL, Figure 9c), and

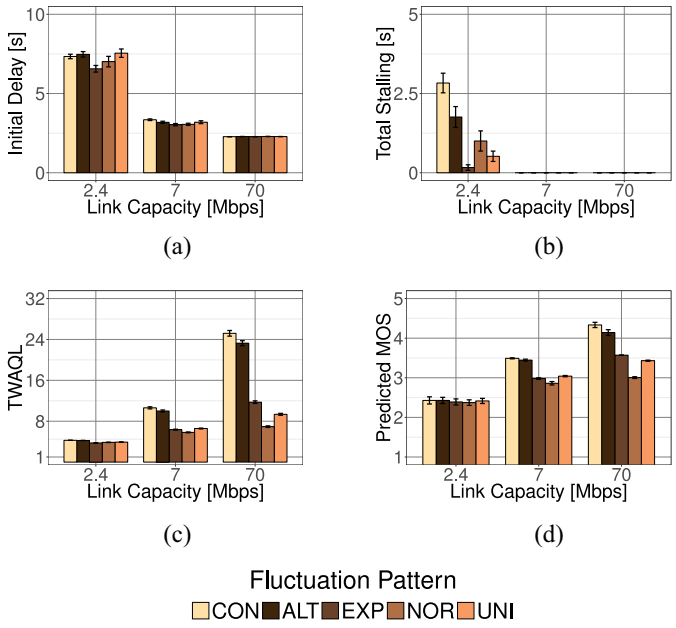

Fig. 9. Results on QoE factors and QoE score (fluctuating streaming capacity). (a) Initial delay. (b) Total stalling time. (c) Time-weighted average quality level (TWAQL). (d) Predicted MOS from P.1203.

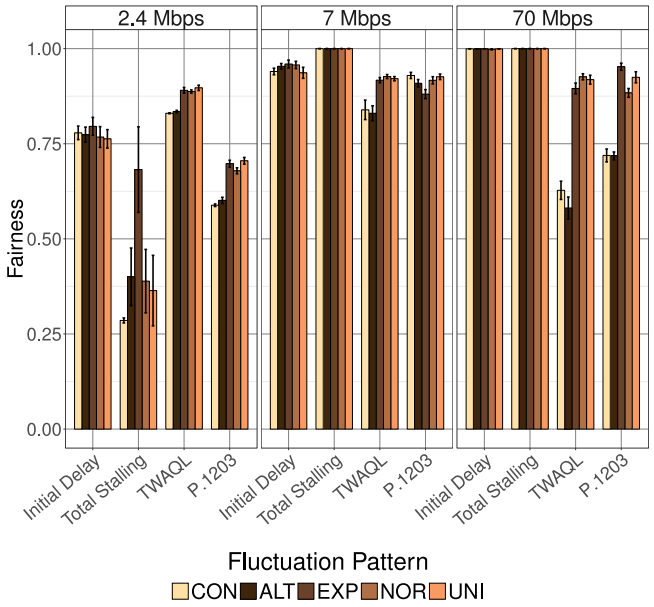

Fig. 10. Results on QoE fairness (fluctuating streamingcapacity).

the predicted MOS from P.1203 (Figure 9d), and the corresponding 95% confidence intervals are shown on top. Note that the yellow bar (CON) is equivalent to the yellow bar of Figure 6, which represents the performance of TCPAL under stable streaming conditions, and thus, can be used as a reference.

For initial delay, it can be seen in Figure 9a that the fluctuation of the streaming capacity does not have a huge impact. Instead, for all three mean streaming capacities (2.4 Mbps, 7 Mbps, and 70 Mbps), the initial delay is very similar to the corresponding initial delay for stable streaming capacity (CON). Even with fluctuating streaming capacity, TCPAL is

able to avoid stalling in the 7 Mbps and 70 Mbps conditions, see Figure 9b. Stalling can only be observed for 2.4 Mbps. Here, the total stalling time is reduced compared to stable streaming conditions, especially for exponential fluctuation (EXP), which has a total stalling time close to 0. Note that in 2.4 Mbps condition, the average quality levels (see Figure 9c) are very similar for all fluctuation patterns. Thus, the reason for the reduced stalling is that, although the mean streaming capacity is low, phases with high streaming capacity can occur, which help to quickly fill the buffer and avoid stalling. This effect is most exposed for EXP, while the other fluctuation patterns show more moderate capacity fluctuation.

Figure 9c shows the results for the TWAQL metric, i.e., the time-weighted average quality level of played out segments. It can be seen that fluctuations of the streaming capacity cause TCPAL to reduce the quality level compared to stable streaming capacity. The reason is that phases with low streaming capacity in the fluctuation patterns will increase the download times of segments. This causes congestion in TCPAL more often, which results in a reduction of the requested quality levels. As described above, the TWAQL is still very similar to CON for 2.4 Mbps, but it diverges more and more for higher capacities. In case of 70 Mbps, exponential (EXP), normal (NOR), and uniform (UNI) fluctuation even reduce the TWAQL metric to half of CON or less. In contrast, alternating fluctuation (ALT) results in only a slight reduction of the quality level, even with the high mean capacity of 70 Mbps. The reduced quality levels eventually lead to a reduced predicted MOS score, as expected, which can be seen in Figure 9d. Here, the worst fluctuation patterns reach a P.1203, which is around 3 for 7 Mbps (down from 3.5 for CON) and for 70 Mbps (down from 4.3 for CON). For 2.4 Mbps, the quality levels were similar to CON and stalling was only slightly reduced from a low CON baseline. Thus, in this scenario, the P.1203 scores stay similar to CON for all fluctuation patterns, having a score around 2.4.

Figure 10 compares the QoE fairness of TCPAL for fluctuating streaming capacity with its QoE fairness for stable streaming capacity. For all mean streaming capacities, the fairness score for initial delay is similar to CON for all fluctuation patterns. For stalling with a mean streaming capacity of 2.4 Mbps, the QoE fairness score increases compared to CON although also the confidence intervals grow. Still, for ALT, NOR, and especially for EXP, the QoE fairness is significantly higher compared to CON. For the other mean streaming capacities no stalling occurred, hence, all QoE fairness scores are equal to 1. When considering TWAQL, an increased QoE fairness can be observed for EXP, NOR, and UNI for all mean streaming capacities. Only with ALT, TCPAL has roughly the same fairness as with CON. Also for P.1203, EXP, NOR, and UNI result in increased QoE fairness compared to CON for 2.4 Mbps and 70 Mbps, while ALT stayed similar to CON. For P.1203 under 7 Mbps, all scores are very similar, only EXP results in a slightly lower QoE fairness.

To sum up, the performance analysis of TCPAL under fluctuating streaming capacity showed that absolute QoE metrics slightly degrade compared to stable streaming capacity, which is an expected result. However, the QoE fairness of TCPAL could stay on the same level, and could even increase for some fluctuation patterns. This shows that TCPAL is able to fulfill its design goals of a QoE-fair adaptation even under fluctuating streaming conditions.

VI. Exploring the Impact of TCPAL Parameters

Finally, the impact of the four parameters of TCPAL α, β, γ, and δ are investigated in detail. In the design process of TCPAL, and the performance evaluation presented above, the investigated parameters were initialized as $\alpha = 1, \beta = 0.9, \gamma = 2$, and $\delta = 0.75$. In the following, the parameters are shortly revisited and the design considerations are explained.

- α: TCPAL considers the maximum needed bandwidth r_{max} of each video. This value is derived from the mean bit rate of all segments of the highest quality level $\bar{r}_{max}$. Due to bit rate variations caused by variable bit rate encoding (VBR), the actual bit rate fluctuates around the mean. Therefore, the maximum needed bandwidth is increased by α-times the standard deviation of the bit rate of the segments at the highest quality level $\sigma_{r_{max}}$, i.e., $r_{max} = \bar{r}_{max} + \alpha \cdot \sigma_{r_{max}}, \alpha \geq 0$. The initial value of $\alpha = 1$, i.e., an increase of the mean bit rate by one standard deviation, was chosen to account for a portion of the VBR fluctuations.
- β: TCPAL is based on $\hat{C}_{max}$, i.e., the individually observed maximum link capacity, and adjusts the upper limit of the quality level l_u based on the ratio of the current observed link capacity $\hat{C}$ and $\hat{C}_{max}$. As a maximum observation might be prone to outliers, the observed $\hat{C}_{max}$ value might be too optimistic and can rarely be reached again by future observations $\hat{C}$. This would prevent the streaming clients from requesting the maximum quality level l_{max}. Therefore, the parameter $\beta \leq 1$ relaxes this condition, i.e., it virtually reduces $\hat{C}_{max}$, such that the maximum quality level can still be requested when $\hat{C} \geq \beta \cdot \hat{C}_{max}$. This results in $l_u = \lceil \frac{\hat{C}}{\beta \cdot \hat{C}_{max}} \cdot l_{max} \rceil$. The initial value was set to $\beta = 0.9$ to only moderately reduce the individually observed maximum link capacity.
- γ: In the congestion avoidance phase, TCPAL probes the maximum sustainable quality level. To keep the clients more synchronized, the quality level can only be increased by 1 after each γ segments. Thus, the parameter γ controls the aggressiveness of the quality level increases in the congestion avoidance phase, i.e., a higher γ causes TCPAL to wait longer before the quality level can increase. The initial value of $\gamma = 2$ results in possible quality level increases every 8 s ($\tau = 4$ s), such that TCPAL does not increase the quality too aggressively in the congestion avoidance phase.
- δ: TCPAL reduces the quality level when congestion is detected or the buffer level is too low. This is controlled by parameter $\delta < 1$, i.e., the new quality level $l_n = \delta \cdot l_c$ is a share of the current quality level l_c. Due to the probing in the congestion avoidance phase, the quality level has to be reduced quite often. Thus, the initial value of $\delta = 0.75$ was chosen to not decrease the quality level too much.

TABLE II
INVESTIGATED VALUES OF TCPAL PARAMETERS

	Lower	Initial	Higher
α	0	1	2
β	0.75	0.9	1
γ	1	2	3
δ	0.5	0.75	0.9

TABLE III
IMPACT OF TCPAL PARAMETERS ON QOE FACTORS

	Initial Delay	Total Stalling	TWAQL	P.1203
↗ α	–	–	–	–
↗ β	–	–	–	–
↗ γ	–	–	–	–
↗ δ	↗	↗ (H)	↗	–

To investigate the impact of changing these initial parameters, the following approach was taken. In addition to the initial value of each parameter, a value below and a value above was selected, see Table II. Thus, in total there were three values for each parameter. Then, a full factorial design with 81 parameter combinations was evaluated for the most challenging scenario with six diverse HAS clients (2 × 360p, 2 × 720p, 2 × 1080p). Each parameter combination was evaluated with ten simulation runs for each of the three different stable streaming capacities, namely, 2.4 Mbps, 7 Mbps, and 70 Mbps. The remaining parameters of the streaming remained as $\tau = 4$s, $l_{max} = 32$, $b_i = 12$s, $b_s = 4$s, $b_l = 8$s, $b_d = 16$s.

The QoE factors and QoE fairness scores for each of the four metrics (initial delay, total stalling time, time-weighted average quality level, and predicted MOS from P.1203) were recorded for each run. The impact of each parameter on the eight metrics was analyzed with four-factor analyses of variance (ANOVAs). Significant differences between the groups were further investigated with a post-hoc analysis based on Tukey's honestly significant difference (HSD) test to obtain the trend of the parameter impact.

The results for the QoE factors are presented in Table III. In the table, an arrow indicates that, according to Tukey's HSD test, there is a significant difference between the means of the QoE factor for the lower and the higher parameter value. Note that these differences could always be observed for all three link capacities. The direction of the arrow shows the trend for an increasing parameter value. Additionally, the letters L and H indicate that with the lower or higher parameter value, the QoE factor is significantly different from the initial parameter value. For the parameters α, β, and γ, no trend can be observed for any QoE factor. However, the parameter δ has an impact. Increasing δ leads to higher initial delay, higher total stalling time, and higher average quality level (TWAQL). Here, the higher δ value of $\delta = 0.9$ results in significantly higher total stalling time compared to the initial parameter value $\delta = 0.75$ with an average increase in total stalling time by around 24 s, which is a worse performance. However, the opposing trends of initial delay/total stalling time (higher is worse) and TWAQL (higher is better) cancel out in terms of resulting QoE. This can be seen from the P.1203 score, which is not affected when δ is changed. To sum up, in terms of the QoE factors, the initial parameter selection could not be significantly improved by lower or higher parameter values.

TABLE IV
IMPACT OF TCPAL PARAMETERS ON QOE FAIRNESS

	Initial Delay	Total Stalling	TWAQL	P.1203
↗ α	–	–	↘ (L)	↘ (L)
↗ β	–	–	–	–
↗ γ	–	–	↗	–
↗ δ	–	↘ (L)	↘ (L,H)	↘ (H)

Table IV shows the corresponding results for the QoE fairness scores. Again, the arrows indicate the trend for significant differences between the low and high parameter value based on Tukey's HSD test, and the letters L and H indicate significant differences from the initial parameter value for the lower or higher parameter value, respectively. Again, all presented results were observed for all three link capacities. It can be seen that increasing α will reduce the fairness of the TWAQL and P.1203 metrics. Thereby, a lower $\alpha = 0$ has a significantly higher fairness compared to the initial parameter value $\alpha = 1$. Nevertheless, the mean difference between the lower parameter value and the initial parameter value is 0.03 for TWAQL fairness and 0.02 for P.1203 fairness, and thus, can be neglected. For β, no differences can be observed, but for γ, there is the trend that increasing γ can increase the TWAQL fairness. However, no significant difference from the initial parameter value was found. Increasing δ results in trends towards reduced fairness scores for total stalling time, TWAQL, and P.1203. Here, the lower value $\delta = 0.5$ results in significantly higher fairness for total stalling time (mean difference 0.30) and TWAQL (mean difference 0.04) compared to $\delta = 0.75$, and the higher value $\delta = 0.9$ results in significantly lower fairness for TWAQL (mean difference 0.02) and P.1203 (mean difference 0.03).

Thus, it can be summarized from the parameter study that the initial parameter values already resulted in a decent performance of TCPAL with respect to the QoE factors and the corresponding QoE fairness of the QoE factors. Still, the fairness in terms of total stalling time could be improved by a large margin (mean difference 0.30) compared to the results presented above by selecting a lower δ. This would effectively also reduce initial delay and total stalling, but also the requested quality levels, while keeping the same overall QoE score in terms of P.1203. Moreover, selecting lower values for α would have a beneficial effect on the fairness in terms of the requested quality level and the P.1203 score with no negative side effects, however, the improvements are only marginal (mean difference 0.02).

VII. CONCLUSION

This paper presented a TCP-inspired adaptation logic (TCPAL) for QoE fairness among independent and heterogeneous clients of HTTP adaptive video streaming. It is based on the idea of transferring the concepts behind TCP fairness

to achieve QoE fairness. Therefore, four design considerations were incorporated to drive heterogeneous HAS clients to converge on similar quality levels instead of similar bit rates.

In particular, TCPAL limits the buffer filling rate in order to keep the clients and their buffer more synchronized. It relies on an estimation of the streaming capacity, which each client can observe individually, and which they can use to agree on the quality level. Moreover, it directly reuses the TCP slow start and congestion avoidance algorithm, which is originally used for adjusting the congestion window, for selecting the next quality level. Finally, it handles a decreasing buffer similar to congestion in TCP, while compensating bit rate requirements of heterogeneous HAS clients to keep them in sync.

The performance evaluation in scenarios with stable streaming capacity showed that TCPAL performed on par with other HAS adaptation logics in terms of QoE for low and medium streaming capacity. Only for high streaming capacity it performs slightly lower than competing HAS adaptation logics, although it still achieves a high QoE score of 4.3 on the MOS scale. This reduced QoE score can be considered the cost of QoE fairness. However, the QoE fairness itself is also lower for high streaming capacity due to the on-/off-problem of HAS, which causes different HAS clients to observe different streaming capacities, and thus, TCPAL selects different quality levels. In this scenario, TCPAL has to be improved by incorporating a more robust capacity estimation. In the other cases, TCPAL works as intended. For low streaming capacity, the achieved QoE fairness is again on a par with other HAS adaptation logics for low streaming capacity. The reason is that in this scenario, all streaming clients struggle to avoid stalling, and thus, there is not much margin left for improving both the QoE and the QoE fairness. However, with medium streaming capacity, the benefits of TCPAL are clearly visible. Here, TCPAL could significantly improve the QoE fairness compared to other HAS adaptation logics. Even with fluctuating streaming capacity, the QoE fairness achieved by TCPAL did not degrade compared to scenarios with stable streaming capacity.

Finally, a parameter study was conducted to explore the impact of TCPAL parameters on its performance. For this, the initially chosen parameter values of TCPAL were compared in a full factorial design to lower and higher values, respectively. It could be observed that the QoE could not be significantly improved with lower or higher parameter values. However, the fairness in terms of total stalling time could be improved by a large margin, and the fairness in terms of the requested quality level and the P.1203 score could be slightly improved with changing the initially chosen parameters. Still, the current performance of TCPAL is already very promising.

REFERENCES

[1] M. Seufert *et al.*, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2015.

[2] D. Ghadiyaram, J. Pan, and A. C. Bovik, "A time-varying subjective quality model for mobile streaming videos with stalling events," in *Proc. SPIE Appl. Digit. Image Process. XXXVIII*, San Diego, CA, USA, 2015, Art. no. 959911.

[3] K. Zeng, H. Yeganeh, and Z. Wang, "Quality-of-experience of streaming video: Interactions between presentation quality and playback stalling," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, 2016, pp. 2405–2409.

[4] *Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats*, ISO/IEC Standard 23009-1:2012, 2012.

[5] M. Seufert, T. Hoßfeld, and C. Sieber, "Impact of intermediate layer on quality of experience of HTTP adaptive streaming," in *Proc. 11th Int. Conf. Netw. Service Manag. (CNSM)*, Barcelona, Spain, 2015, pp. 256–260.

[6] H. T. T. Tran, T. Vu, N. P. Ngoc, and T. C. Thang, "A novel quality model for HTTP adaptive streaming," in *Proc. 6th IEEE Int. Conf. Commun. Electron. (ICCE)*, 2016, pp. 423–428.

[7] F. Wang, Z. Fei, J. Wang, Y. Liu, and Z. Wu, "HAS QoE prediction based on dynamic video features with data mining in LTE network," *Sci. China Inf. Sci.*, vol. 60, no. 4, 2017, Art. no. 042404.

[8] T. Hoßfeld, L. Skorin-Kapov, P. E. Heegaard, and M. Varela, "Definition of QoE fairness in shared systems," *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 184–187, Jan. 2017.

[9] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck, "QoE-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 12, no. 2, pp. 1–28, 2016.

[10] J. Chen, M. Ammar, M. Fayed, and R. Fonseca, "Client-driven network-level QoE fairness for encrypted 'DASH-S'," in *Proc. ACM SIGCOMM Workshop QoE Anal. Manag. Data Commun. Netw. (Internet-QoE)*, Florianopolis, Brazil, 2016, pp. 55–60.

[11] E. Thomas *et al.*, "Applications and deployments of server and network assisted DASH," in *Proc. Int. Broadcast. Conven. (IBC)*, Amsterdam, The Netherlands, 2016, p. 22.

[12] *Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 5: Server and Network Assisted DASH (SAND)*, ISO/IEC Standard 23009-5:2015, 2016.

[13] M. Seufert, N. Wehner, P. Casas, and F. Wamser, "A fair share for all: Novel adaptation logic for QoE fairness of HTTP adaptive video streaming," in *Proc. 14th Int. Conf. Netw. Service Manag. (CNSM)*, Rome, Italy, 2018, pp. 19–27.

[14] "Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport," Int. Telecommun. Union, Geneva, Switzerland, ITU-Recommendation P.1203, 2016. [Online]. Available: https://www.itu.int/rec/T-REC-P.1203/en

[15] "Subjective video quality assessment methods for multimedia applications," Int. Telecommun. Union, Geneva, Switzerland, ITU-Recommendation P.910, 2008.

[16] 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," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, 2015.

[17] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH)," in *Proc. 20th Int. Packet Video Workshop (PV)*, San Jose, CA, USA, 2013, pp. 1–8.

[18] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," in *Proc. 4th Workshop Mobile Video (MoVID)*, Chapel Hill, NC, USA, 2012, pp. 37–42.

[19] 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, 2012, pp. 173–178.

[20] H. Ott, K. Miller, and A. Wolisz, "Simulation framework for HTTP-based adaptive streaming applications," in *Proc. Workshop NS-3 (WNS3)*, Porto, Portugal, 2017, pp. 95–102.

[21] 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.

[22] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, Nice, France, 2012, pp. 97–108.

[23] C. Timmerer, M. Maiero, and B. Rainer, "Which adaptation logic? An objective and subjective performance evaluation of HTTP-based adaptive media streaming systems," *arXiv preprint arXiv:1606.00341*, 2016. [Online]. Available: https://arxiv.org/abs/1606.00341

[24] R. Jain, D.-M. Chiu, and W. R. Hawe, *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System*. Hudson, MA, USA: Eastern Res. Lab., 1984.
[25] R. Houdaille and S. Gouache, "Shaping HTTP adaptive streams for a better user experience," in *Proc. 3rd Annu. ACM Conf. Multimedia Syst. (MMSys)*, Chapel Hill, NC, USA, 2012, pp. 1–9.
[26] F. Wamser *et al.*, "Modeling and performance analysis of application-aware resource management," *Int. J. Netw. Manag.*, vol. 25, no. 4, pp. 223–241, 2015.
[27] A. Bentaleb, A. C. Begen, and R. Zimmermann, "SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking," in *Proc. ACM Multimedia Conf. (MM)*, Amsterdam, The Netherlands, 2016, pp. 1296–1305.
[28] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A mobile edge computing-based architecture for improved adaptive HTTP video delivery," in *Proc. IEEE Conf. Stand. Commun. Netw. (CSCN)*, Berlin, Germany, 2016, pp. 1–6.
[29] G. Cofano *et al.*, "Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming," in *Proc. 7th ACM Int. Conf. Multimedia Syst. (MMSys)*, Klagenfurt, Austria, 2016, Art. no. 3.
[30] J. Postel, "Transmission control protocol," Internet Eng. Task Force, Fremont, CA, USA, RFC 793, 1981. [Online]. Available: https://tools.ietf.org/html/rfc793
[31] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," Internet Eng. Task Force, Fremont, CA, USA, RFC 5681, 2009. [Online]. Available: https://tools.ietf.org/html/rfc5681
[32] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proc. Internet Meas. Conf. (IMC)*, Boston, MA, USA, 2012, pp. 225–238.
[33] Blender Foundation. (2008). *Big Buck Bunny*. [Online]. Available: https://www.bigbuckbunny.org
[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[35] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools," *IEEE Netw.*, vol. 17, no. 6, pp. 27–35, Nov./Dec. 2003.
[36] S. S. Chaudhari and R. C. Biradar, "Survey of bandwidth estimation techniques in communication networks," *Wireless Pers. Commun.*, vol. 83, no. 2, pp. 1425–1476, 2015.
[37] Z. Akhtar *et al.*, "Oboe: Auto-tuning video ABR algorithms to network conditions," in *Proc. Conf. ACM Special Interest Group Data Commun. (SIGCOMM)*, Budapest, Hungary, 2018, pp. 44–58.

Michael Seufert received the bachelor's degree in economathematics and the Diploma and Ph.D. degrees in computer science from the University of Würzburg, Germany, and holds the Erstes Staatsexamen in mathematics, computer science, and education (teaching in secondary schools). From 2012 to 2013, he was with FTW Telecommunication Research Center, Vienna, Austria, researching in the area of user-centered interaction and communication economics. From 2013 to 2017, he was a Researcher with the Chair of Communication Networks, University of Würzburg. From 2018 to 2019, he was a Post-Doctoral Fellow and a Scientist with the AIT Austrian Institute of Technology, Vienna. Since 2019, he has been a Post-Doctoral Fellow and a Scientist with the Chair of Communication Networks, University of Würzburg. His research focuses on QoE of Internet applications, artificial intelligence and machine learning for networks, monitoring and analytics of (encrypted) network traffic, proactive QoE- and socially-aware traffic management solutions, monitoring and orchestration of edge cloud services, group-based communications, as well as performance analysis and modeling of communication systems.

Nikolas Wehner received the master's degree in computer science from the University of Würzburg, Germany. Since 2018, he has been a Research Engineer with the Center for Technology Experience, AIT Austrian Institute of Technology, Vienna, Austria. His research focuses on QoE of Internet applications, big data analytics, network measurements, and performance analysis.

Pedro Casas received the Electrical Engineering degree from the Universidad de la República, Uruguay, in 2005 and the Ph.D. degree in computer science from the Institut Mines-Télécom, Télécom Bretagne, France, in 2010. He is a Scientist in ICT Security and Information Management with the AIT Austrian Institute of Technology, Vienna. He was a Post-Doctoral Research Fellow with LAAS-CNRS Lab, Toulouse, from 2010 to 2011, and a Senior Researcher with Telecommunications Research Center Vienna (FTW) from 2011 to 2015. He was a Project Manager and a Technical Work Leader in different networking-related initiatives, including research projects and commercial solutions. His work focuses on machine-learning and data mining-based approaches for networking, big data analytics and platforms, Internet network measurements, network security and anomaly detection, as well as QoE modeling, assessment, and monitoring. He has published over 135 networking research papers in major international conferences and journals, received 12 awards for his work, including seven best paper awards. He is the General Chair for different conferences, workshops, and leading actions in network measurement and analysis, including the IEEE ComSoc ITC Special Interest Group on Network Measurements and Analytics.