Qual User Exp (2017) 2:7
https://doi.org/10.1007/s41233-017-0010-9

RESEARCH ARTICLE

Quantitative comparison of application–network interaction: a case study of adaptive video streaming

Susanna Schwarzmann[1] · **Thomas Zinner**[1] · **Ognjen Dobrijevic**[2]

Received: 31 December 2016 / Published online: 22 July 2017

Abstract Managing quality of experience (QoE) is now widely accepted as a critical objective for multimedia applications and the supporting communication systems. In general, QoE management encompasses: (1) monitoring of the key influence factors and QoE indicators, and (2) deciding on the appropriate control actions as specified by the management goal. Many multimedia applications, e.g., video streaming and audio conferencing, are able to adjust their operational parameters so as to react to variations in the network performance. However, such an adaptation feature is mostly based on a local client view of the network conditions, which may lead to an unfair allocation of network resources among heterogeneous clients and, thus, an unfair QoE distribution. In order to tackle this issue, there is the call for a cooperation between the applications and the underlying network, which includes application–network interaction (App-Net) in terms of: (1) exchanging information on the monitored QoE indicators, and (2) coordinating the QoE control actions. Various App-Net mechanisms focusing on specific use cases and applications have been proposed to date. This paper gives an overview of App-Net mechanisms and proposes a generic App-Net model that provides the means to realize a coordinated QoE-centric management. Based on the App-Net model, we develop an evaluation methodology to compare three App-Net mechanisms for managing QoE of HTTP adaptive streaming (HAS) against a baseline HAS service. The aim of this quantitative comparison is to explore the trade-offs between QoE gains and the complexity of App-Net implementation, with respect to the number of monitoring and control messages, achieved video quality, and QoE fairness among heterogeneous clients. Our ultimate goal is to set up reproducible experiments that facilitate a holistic evaluation of different App-Net mechanisms.

Keywords Quality of experience (QoE) · QoE-centric management · Monitoring and control · Application–network interaction · Quantitative comparison · HTTP adaptive streaming (HAS) · Video quality · QoE fairness

✉ Thomas Zinner
zinner@informatik.uni-wuerzburg.de

Susanna Schwarzmann
susanna.schwarzmann@informatik.uni-wuerzburg.de

Ognjen Dobrijevic
ognjen.dobrijevic@fer.hr

[1] Institute of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany

[2] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia

Introduction and motivation

The concept of quality of experience (QoE) models the notion of service quality as subjectively perceived by end-users [1, 2] and takes into account the effect of different influence factors (IFs) that are introduced by elements of the service delivery chain, such as end-user equipment, application servers, and network infrastructure, but also that of the end-user and context-related factors [2]. Managing QoE is now widely accepted as a critical objective for multimedia applications (e.g., for video streaming or audio conferencing) and the supporting communication systems. This objective has been boosted, in particular, by the emergence of popular content and service providers like Google, Netflix, Spotify, and Amazon. However, the Internet has not been designed for rigorous resource demands imposed by the multimedia

services, while network mechanisms for a differential treatment of multimedia traffic have not been implemented on a wide scale.

As over-provisioning network resources is often economically unfavorable or technologically limited, e.g., for wireless networks, general solutions to cope with the current communication network limitations propose to adjust configuration and operational parameters of the multimedia applications so as to react to variations in the network performance. Many multimedia applications are able to dynamically adjust their behavior, with mechanisms ranging from fundamental TCP and UDP protocols to novel technologies such as HTTP adaptive streaming (HAS) [3]. However, such an adaptation feature is mostly based on a local client/server view of the network conditions, which may lead to an unfair distribution of the available resources among contending traffic flows and, hence, negatively impact QoE. Even if a fair resource share is achieved, heterogeneous clients may still receive different QoE levels [4]. For instance, a smartphone device and a laptop computer may be assigned an equal network bandwidth for video streaming, although a continuous video reproduction at the smartphone would demand less bandwidth.

The general QoE management cycle includes two phases: (1) monitoring of the key QoE influence factors (QoE-IFs), and (2) deciding on the appropriate control actions that adhere to the management objective. Since there are various QoE-IFs, monitoring strategies include design choices on, e.g., which quality indicators to measure, what are the measurement points, and how often are the measurements carried out. Measurements can be taken on different levels of the protocol stack, resulting in different granularity and types of the retrieved information (e.g., per-packet inspection or flow monitoring). However, we omit such details in this work and consider network assessment of the associated QoE-IFs, which will be referred to as *network monitoring*, and user-device or application assessment of the corresponding QoE-IFs, i.e. *application monitoring* (c.f. Fig. 1). Control actions, on the other hand, can be employed so as to adapt network behavior, application behavior, or both. For the network level, mechanisms such as bandwidth allocation (e.g., [4, 5]), routing reconfiguration (e.g., [6, 7]), and traffic prioritization (e.g., [8]) can be applied, which we refer to as *network control*. At the application level, for example, a client can request from a server another media content resolution and bitrate (e.g., for video) in response to bandwidth fluctuations, or adapt its playout delay (e.g., for audio) so as to alleviate consequences of network jitter, which will collectively be referred to as *application control*.

The QoE management objective can be defined in a different manner as improving or optimizing QoE, e.g., to provide a fair assignment of QoE levels among end-users.

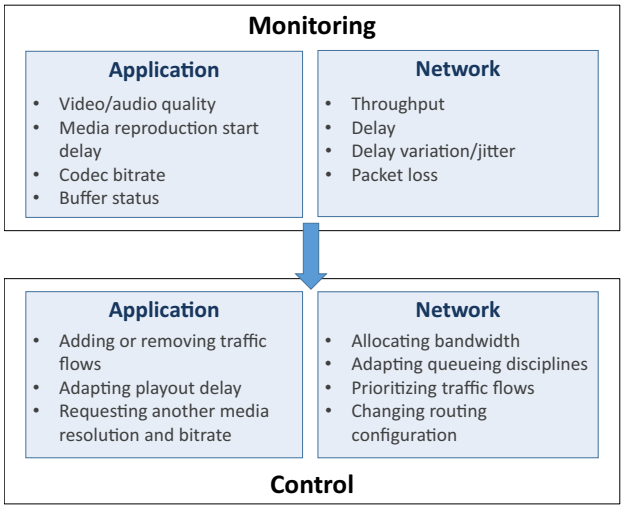


Fig. 1 Illustration of the QoE management aspects that consist of monitoring and control of application and network

In order to achieve such an optimization objective, an explicit exchange of information on the monitored QoE-IFs and of control instructions between the applications and the supporting network is required. The latter resembles typical cross-layer design approaches, which focus on the information exchange. So as to extend this notion in a way to include the solutions that aim at establishing a cooperative control interplay between applications and the network, we introduce the term Application–Network Interaction, or App-Net for short. With this in mind, several App-Net solutions also propose a coordinated approach for the QoE management (e.g., [5, 8]), which assumes a logically centralized entity collecting the application and network monitoring information, and then deciding on control actions between these two levels. In this work, we refer to such an entity as the *policy manager* (PM).

As the App-Net solutions propose different design choices, algorithms, and actions for QoE management, they influence QoE-IFs in different ways and to different extents. This fact calls for a comparison of the strategies to estimate their performance. However, the current works that present App-Net strategies discuss the benefits and impacts on QoE-IFs for diverse use cases and different quality metrics. Furthermore, different testing environments which are applied and the dissimilarity of the conducted experiments impair the comparability. In order to reveal and compare the performance of different App-Net approaches, a benchmarking is necessary to gain insights on how these approaches impact QoE in different scenarios. To the best of our knowledge, such holistic evaluations do not exist yet. In this work, we aim at comparing different App-Net approaches and estimating their complexity. While the goal is not to conduct a QoE study, but, instead, we discuss the impact of the interaction approaches

on various IFs for QoE. Our contribution is, hence, threefold.

1. We facilitate reproducible experiments by setting up a controlled environment and implementing three App-Net approaches based on the ideas from state-of-the-art work. To achieve reproducibility and comparability, the complete source code of this work is made publicly available.[1]
2. We use this infrastructure to evaluate and compare the chosen App-Net mechanisms in different scenarios. With respect to the considered QoE metrics and scenarios, we go beyond evaluations in the original works in order to reveal pros and cons of the App-Net mechanisms with respect to several QoE-IFs.
3. We investigate the number of exchanged interaction messages as a first approximation of the complexity for a specific App-Net strategy.

Building on our previous work [9], in which we identified a generic App-Net model and provided an initial evaluation of two App-Net mechanisms for enhancing HAS QoE, this paper:

1. extends the overview of existing App-Net solutions to include additional state-of-the-art results and an insight into the QoE-IFs considered by each of them,
2. expands the App-Net model so as to provide a more in-depth discussion on possible interaction strategies, and
3. provides a quantitative comparison of three App-Net mechanisms for improving HAS QoE against a baseline HAS service.

The rest of this paper is structured as follows. "Generic App-Net model" gives an overview of App-Net solutions that focus on improving QoE and describes the generic App-Net model. Based on this model, "Evaluation methodology" explains the App-Net comparison methodology and the evaluation testbed that is used to gather experimental results. The quantitative comparison results are presented and analyzed in "Evaluation results", followed by the conclusions and a future work overview.

Generic App-Net model

In the current highly competitive market, end-users can decide between a large number of service providers. It is, hence, an imperative for those providers to deliver their services in a way that offers end-users a rich experience, which is especially critical for real-time multimedia services. To reach a "good" end-user QoE, different IFs that are related to end-users, systems, applications/services, and context need to be taken into account [2]. With respect to end-users, the factors may encompass demographic information, user preferences and expectations. System-wise factors include typical features and performance of the supporting communication systems, such as network throughput, delay, jitter and packet loss, as well as server and user device characteristics in terms of, e.g., processing power, memory, screen size, and presentation features. Parameters referring to applications/services, on the other hand, comprise supported service features, available protocol support, media encoding, resolution and sample rate, but also content-specific information (e.g., content type). Factors describing the situation in which a service is being delivered relate to, e.g., user activity and mobility, time of usage, user location, and service cost.

Recent years witness a plethora of research results that focus on issues in managing QoE for real-time multimedia services [10, 11], such as video streaming and audio conferencing. The associated mechanisms differ in QoE-IFs that they consider, as well as level of support for QoE monitoring and control. Since a main prerequisite for improving or optimizing QoE is the level of support for application/network monitoring and control, our survey of the existing App-Net mechanisms is extended beyond our previous work [9] and uses the following classification:

1. Monitoring support:
   (a) *Application monitoring* (AMN)—QoE-IFs related to applications/services and servers/user devices are assessed at the application level.
   (b) *Network monitoring* (NMN)—QoE-IFs corresponding to the underlying communication network are evaluated at the network level.
2. Control support:
   (a) *Application control* (ACT)—an application-level entity is informed on application-level or network-level QoE-IFs, and holds QoE control.
   (b) *Network control* (NCT)—a network entity is reported on application-level and/or network-level QoE indicators, and runs the QoE control process.
   (3) *PM-coordinated control* (PCT)—application-level and network-level QoE-IFs are conveyed to a PM, which coordinates control operations among applications and the network.

App-Net mechanisms for improving QoE

One of the most dominantly used multimedia services over the Internet is video streaming, which is commonly built upon the HAS technology and its widely accepted standard,

[1] https://github.com/lsinfo3/App-Net.git.

Dynamic Adaptive Streaming over HTTP (DASH) [12]. DASH partitions video and audio content into short segments, of a few seconds in length, and stores them on a server. Each of the segments is produced in various representations that provide, e.g., different encoding bitrate and resolution so as to facilitate adaptation of video quality to varying network bandwidth. These segment representations are accompanied by a media presentation description (MPD) file, which contains all the meta-information on media content. General DASH operation consists of a client fetching the MPD file from a server that hosts the requested media content, and then choosing which segment representations to retrieve and play out. When deciding on a segment representation to download, a DASH client takes state of its buffer and estimated network throughput into account.

Adzic et al. [13] address DASH streaming in a mobile, bandwidth-limited environment. They present an ACT-based mechanism that prevents a client from switching to a higher bitrate segment and, thus, consuming additional bandwidth, if significant QoE gains cannot be achieved. While network bandwidth is assessed at the application level, common application-related IFs for DASH, such as encoding type and bitrate, video resolution and frame rate, are collected via AMN, but also additional, content-related factors regarding spatial and temporal characteristics. In [14], a QoE-driven management framework for Voice over IP (VoIP) services in mobile broadband networks is described. This framework builds upon application-level inspection of network conditions that are measured via packet delays, and then having communication end-points adapt audio encoding or packetization parameters when the performance is degraded.

QoE-aware DASH [15] is an ACT approach that adds a transparent proxy on the path between a DASH client and server to perform estimation of the available network bandwidth. This way, DASH clients can obtain and use more precise insight into network conditions when choosing which segment representations to download, in addition to typical application-related IFs such as encoding type and bitrate. A proposal for improving QoE for Skype VoIP services is presented in [16]. It discusses the strategies of adjusting forward error correction (FEC) coding scheme and injecting redundant voice traffic with respect to different voice encoders, so as to conform to network packet loss and packet loss burstiness.

Bouten et al. [17] address the issue of QoE optimization for multiple HAS clients and introduce intermediate proxies that detect the maximum segment quality levels the clients are permitted to retrieve, subject to the measured outbound proxy bandwidth. For this, a proxy runs a rate adaptation algorithm when a HAS client needs to establish or terminate a connection with the server and then the proxy directs each client which segment representation to download. An ACT-centric cross-layer framework for real-time over-the-top (OTT) multimedia applications, e.g., Skype conferencing, is described in [18]. The framework utilizes a simple NMN to detect congestion in mobile broadband networks, which represents a system-wise QoE-IF. This congestion information is then delivered to a Skype application, which decides on whether to invoke encoding-rate adaptation or network-level intra-flow packet prioritization.

Houdaille and Gouache [19] propose a bandwidth manager located in home gateway that allocates a fair share of network resources to each HAS session. The NCT manager monitors the available network bandwidth and then performs traffic shaping for HAS clients, which reduces the number of media content adaptations at the application level. NCT-centric mechanisms that specifically target YouTube are presented in [6, 20]. In [6], a software-defined networking (SDN) controller is notified on the current buffer level and media flow bandwidth demand, which are then used to choose less congested links for media flows.

Nam et al. adopt the SDN concept to increase QoE for HAS services [7]. The central part of their PCT-based solution is an SDN application, which calculates a new route when network congestion takes place. Monitoring is performed periodically by clients, which report application-level metrics such as buffer status, and by an SDN controller, which reports on network performance (e.g., jitter). A similar SDN-centric proposal that includes NCT with QoE-aware network paths is presented in [21], addressing different service categories (e.g., audio conversation and video streaming).

The QoE Fairness Framework [4] is a PCT approach for a fair QoE distribution among heterogeneous DASH clients. It realizes App-Net by monitoring network bandwidth, collecting user device features and MPDs, and reporting them to a PM. The PM then calculates appropriate media bitrate for each client and directs the clients which media representation to choose. Another PCT proposal for DASH services, which reduces video freezes (or stallings), is presented in [8]. There, network devices are regularly polled for traffic statistics, while clients report on application metrics such as buffer state. If a client buffer runs empty, a PM prioritizes specific media segments in the network and instructs the associated client to request the matching media quality.

NOVA, short for Network Optimization for Video Adaptation, is developed by Joseph et al. [22]. It implements a cross-layer joint optimization of resource allocation and quality adaptation. NMN entities regularly send network state updates to a base station, while AMN notifications from clients are only sent to the base station if

video experiences a re-buffering event. A NOVA algorithm then computes bandwidth slices for each DASH client so that several QoE-IFs are optimized, the respective network controller performs the bandwidth slice allocation, while rate adaption is performed by the clients independently. Cofano et al. [5] describe several NCT strategies that target QoE fairness among different clients. The bandwidth reservation strategy uses application-level information so as to assign media flows to dedicated queues on network interfaces, while a hybrid strategy combines bandwidth reservation with media bitrate calculation (similarly to [4]).

Server and Network Assisted DASH (SAND) [23] is an ongoing standardization effort to specify a DASH messaging framework, with the goal to enhance multi-client QoE. The framework introduces a DASH-aware network element (DANE), which is capable of identifying and parsing MPD files, but also of modifying video segments. The SAND architecture proposes the following communication interfaces: client-to-DANE, DANE-to-DANE, and DANE-to-client. This allows the exchange of different message types and, thus, the interaction between DASH clients, media content servers, and the network devices. Exchanged information may include the intent of a client to request a video from a specific server, client capabilities with respect to connectivity, or information on the currently cached video segments at a particular server. SAND does not specify any adaptation logic, but provides a basis for the App-Net mechanisms by introducing DANEs and standardized communication interfaces.

More generic App-Net approaches are investigated in [24–26]. Ferguson et al. propose the participatory networking paradigm [24], which enables end-users, user equipment and applications to interact with the network. For instance, end-users can trigger application reconfiguration based on the network state or postpone initiation of, e.g., a conference call until sufficient network resources are available. OpenADN, or Open Application Delivery Networking [25, 26], introduces an abstraction layer between applications and the network, so as to enable the mapping of policies across various multimedia services to network behavior. OpenADN is based on the SDN principles, enabling, e.g., a network operator to customize message routing by chaining different middleboxes in the path.

Building the generic App-Net model

The approaches presented previously employ several distinct functions to realize App-Net, but all exhibit a common control loop. First, relevant monitoring information on applications, network, or both is collected. Then, control actions are computed and enforced, at network-level, application-level, or both. Neglecting implementation specific details of these App-Net functions, we group them into the following categories: application or network monitoring, and application or network control. The latter categories are used to identify fundamental building blocks and derive an abstract App-Net model (Fig. 2).

The model consists of four functional blocks, namely Application Monitoring (AMN), Application Control (ACT), Network Monitoring (NMN), and Network Control (NCT). AMN supervises the state of running applications, while NMN captures performance parameters that indicate the network state. Control functions of the model are shared between ACT and NCT—the former features adaptation actions implemented in applications, while the latter does the same for the network level. In order to jointly optimize behavior of applications and the network, and hence be able to achieve a fair QoE distribution, monitoring information about applications and the network may be provided to a logically centralized PM. Utilizing this information, the PM can decide on adaptation actions and instruct ACT/NCT to enforce them, thus realizing a PCT (joint control) approach.

The actual realization of the control loop varies between the App-Net approaches, which differ in monitoring accuracy, the degree of control support, and the implementation domain of the specific monitoring/control function. Such an App-Net cooperation may encompass decisions on whether assessment of key performance indicators and the adjustment of parameters are performed in the network, the applications, or both. Additionally, the App-Net mechanisms differ with regard to the frequency of control actions and technical implementation specifics. Thus, we classify the presented mechanisms with respect to the identified App-Net functional blocks (Table 1), also comparing monitoring accuracy and the frequency of control actions. Furthermore, we provide an insight into QoE-IFs considered by the App-Net mechanisms. Initial (I) refers to monitoring/control being performed only during service establishment, triggered (T) assumes that monitoring or control is initiated by an irregular event, while periodic (P) regards the actions in regular time intervals.

One important aspect that needs to be considered in the App-Net design regards network neutrality. Network neutrality (NN), which can be viewed from a technological, an economical or a legal standpoint, is still heavily debated and may be defined in different ways. It is most commonly referred to as the operation of a communication network that treats all of its traffic equally, independently of traffic source and destination, service/application type or content, etc. [27]. The surveyed papers do not explicitly elaborate on NN, but the approaches described in them violate to some extent the main NN principle when aiming to optimize or improve quality for a particular multimedia service (sometimes at the expense of other services). A thorough

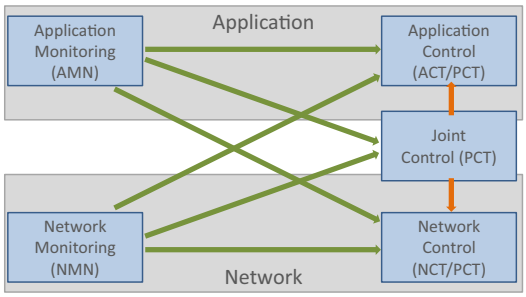


Fig. 2 Key building blocks for realizing application–network interplay

analysis of how the presented App-Net approaches conform to the NN design standards is not included in this paper. However, please note that PCT-based mechanisms which target QoE fairness come closest to fulfilling positive discrimination among end-users in terms of quality. In that sense, it is also interesting to note that some user studies (e.g. by Yiakoumis et al. [28]) have shown that end-users actually want specific services to have a differential treatment in the network.

Evaluation methodology

In this section, we first present a messaging service that is used to realize the information exchange as outlined in the abstract App-Net model. Afterwards, we highlight three PCT-based App-Net mechanisms, which are evaluated in "Evaluation results". Finally, the evaluation testbed and the chosen performance metrics are described. The source code of the messaging service and the implemented App-Net mechanisms will be made publicly available upon acceptance of the manuscript.

Messaging service for the App-Net realization

The interaction between application and network entities requires the exchange of monitoring and control information. In order to enable this exchange and to investigate different approaches on a common architecture, we develop a light-weight publish-subscribe messaging system that is based on ØMQ.[2] The central point of this system is a message broker (ØMQ-Broker), which receives all messages and forwards them to the targeted entities. Figure 3 illustrates the associated information flow. Application and network monitoring data is first delivered to a PM, which then forwards its control decisions to the corresponding application and network control entities via the ØMQ-Broker. Possible scalability issues of such an architecture could be solved by more advanced messaging architectures, if necessary.

Assessment of video quality and video quality fairness

We use the Structural Similarity Metric (SSIM) [29] to assess and compare the quality of transmitted videos. It is a full-reference metric, which means that all uncompressed and distortion-free video frames are used as a reference against the transmitted compressed frames. As luminance, contrast, and structure are considered for computing similarity, this metric reflects well the human perception and is a good approximation for video quality [30, 31]. The SSIM value ranges between 0 and 1, whereby 1 means equality to the uncompressed original content. Hence, we refer to video quality as the displayed quality, not considering other factors like video stalling or initial delay.

We furthermore investigate the fairness in terms of video quality, as this is especially interesting when end-user clients have heterogeneous capabilities, e.g., if they have different device resolutions. Larger devices have higher demands on network resources than those with a small resolution. Due to a TCP fair share, this may lead to a degraded video quality on larger-resolution (e.g., HD) devices. By fairness we assume that each client—independent of its device resolution—perceives a similar displayed quality, measured with SSIM. To investigate the video quality fairness, we use the QoE-fairness index for shared systems introduced by Hossfeld et al. [32] and adjust it for the usage with SSIM, resulting in the following formula:

$$F_{SSIM} = 1 - \frac{2 \times \sigma_{SSIM}}{SSIM_{max} - SSIM_{min}},$$

whereby σ_{SSIM} denotes the standard deviation of the SSIM values observed for all end-user clients during the measurement runs. In our set up, $SSIM_{max}$ is of value 0.9907 and $SSIM_{min}$ is of value 0.8876, as these are the SSIM values of the highest and lowest video quality provided at the server. As we rely on SSIM to compute the video quality fairness, we will refer to it as SSIM-Fairness or F_{SSIM} in the following sections.

Investigated App-Net mechanisms

For our evaluation we choose three PCT-based App-Net solutions that enhance QoIs or video quality fairness for HAS-based video streaming.

Quality of experience fairness framework

The first approach is called QoE Fairness Framework (QoE-FF) [4], which employs a centralized controller (PM)

[2] http://zeromq.org.

Table 1 Classification of the presented App-Net solutions with respect to frequency of NMN, NCT, AMN, and ACT actions

Work	Service type	NMN	NCT	AMN	ACT	QoE-IFs
[4]	HAS	P	–	I	P	Media encoding and encoding bitrate, device type, network bandwidth
[5]	HAS	P	P	I	–	Media encoding and encoding bitrate, device type, network bandwidth
[6]	YouTube HAS	P	T	P	–	Client buffer level, media flow bandwidth demand, network bandwidth
[7]	HAS	P	T	P	–	Media encoding and encoding bitrate, network bandwidth
[8]	HAS	P	T	P	T	Media encoding and encoding bitrate, video buffer, network bandwidth
[13]	DASH	P	–	P	T	Media encoding and spatial/temporal characteristics, network bandwidth
[14]	VoIP	P	–	P	T	Audio encoding and packetization scheme, network delay
[15]	DASH	P	–	P	T	Media encoding and encoding bitrate, network bandwidth
[16]	Skype VoIP	P	–	P	T	Audio encoding, FEC coding scheme, packet loss and loss burstiness
[17]	HAS	P	–	P	T	Encoding bitrate, user subscription, operator cost, network bandwidth
[18]	Skype conferencing	P	T	P	T	Media encoding bitrate, network congestion
[19]	HAS	P	T	–	–	Network bandwidth
[20]	YouTube	P	P	P	–	Video buffer, network bandwidth
[21]	IPTV, audio streaming	P	T	I	–	Transmission delay,
[22]	DASH	P	T	T	–	Media encoding and encoding bitrate, video buffer, network bandwidth
[23]	DASH	X	–	X	X	Not specified
[24]	Multiple applications	T	T	I	P	Initial delay
[25, 26]	Mobile applications	P	P	I	–	Not specified

The actions can be performed periodically (P), initially (I), or triggered (T) by, e.g., an event. X denotes that it is not specified in the work and – denotes that the building block is not implemented

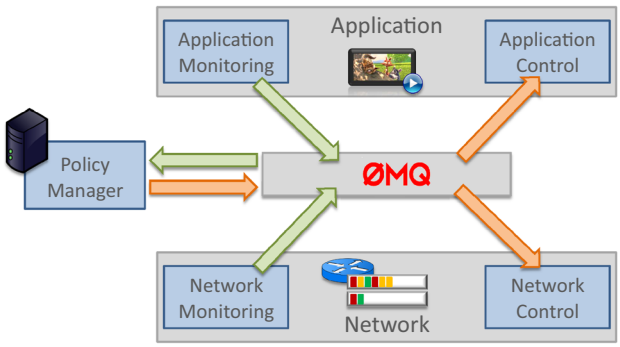


Fig. 3 ØMQ-based messaging service for enabling the information exchange for different App-Net strategies

to govern the video quality selection among heterogeneous HAS clients and to produce a fair QoE distribution. This mechanism is illustrated in Fig. 4.

When starting a new video stream, a HAS client communicates its screen resolution, the requested video bitrate and the associated bandwidth requirements to the PM (1). While monitoring network load and active HAS clients, the PM evaluates QoE for each client using a mapping between the client-specific parameters, the network load, and the SSIM metric (2), computes the fair-share video quality levels, and enforces them at each client. A new video stream or a video stream termination triggers the PM to recompute the bitrate and, therewith, the QoE in terms of video quality for each client (3). The corresponding quality levels are enforced at each cl ient (4). As [4] does not provide details on how the network is monitored, we implement an approach based on network throughput estimation at the clients. Each client regularly estimates its available bandwidth by using time duration of a segment download and the corresponding segment size. This estimation is collected by the PM, which, besides the client-based bandwidth estimations, also considers download and idle times to derive an overall bandwidth estimation.

Network assisted DASH fairness enhancement

The second approach, Network Assisted DASH Fairness Enhancement (NADE), is proposed by Cofano et al. [5]. Similarly to QoE-FF, this strategy targets the video quality fairness among heterogeneous clients in a shared system. For that, the mechanism allocates a guaranteed bandwidth slice to each of the active clients. The steps of this App-Net mechanism are detailed in Fig. 5.

The network monitoring at the router queries the current throughput in fixed intervals and forwards this information to the PM (1). When initiating a new video stream, the client signals its screen resolution to the PM (2), which triggers a re-computation of the bandwidth slices. The PM is able to compute a fair video quality distribution among heterogeneous clients by using a mapping between each resolution/bitrate tuple and the corresponding SSIM value (3). By using this mapping, the knowledge about current

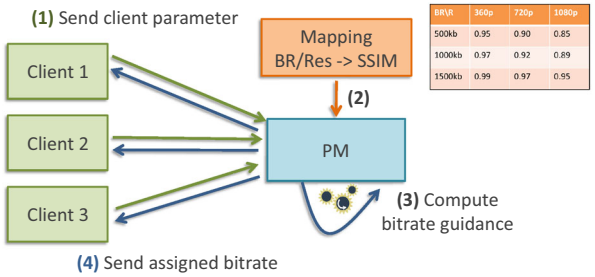


Fig. 4 Illustration of the functionality of the Quality of Experience Fairness Framework (QoE-FF)

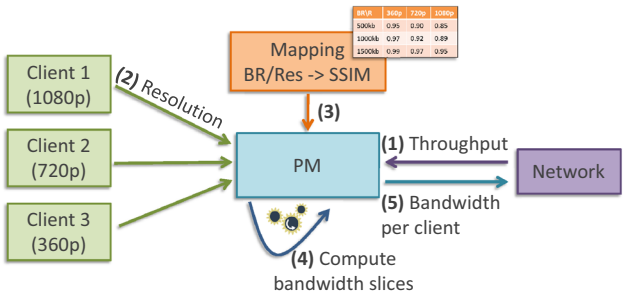


Fig. 5 Illustration of the functionality of NADE (Network Assisted DASH Fairness Enhancement)

throughput, and the number of active clients (including their resolutions), the PM decides on the bandwidth slice for each client or client type (4). The corresponding network configuration is then enforced (5).

Stalling prevention mechanism

The third approach is based on the work by Petrangeli et al. [8]. This App-Net mechanism avoids buffer under-runs in HAS clients by prioritizing download of video segments over other network traffic. We refer to the latter approach as Stalling Prevention Mechanism (SPM). Its functionality is illustrated in Fig. 6.

Network monitoring is realized by having a centralized controller (PM) regularly poll throughput of pre-configured queues in an SDN-enabled switch (1). To enable application monitoring, HAS clients are customized to add the video buffer status to HTTP requests, which are forwarded by the switch to the PM (2). The PM is then able to extract the application-level parameters and calculate whether a video segment will arrive before the corresponding client buffer runs out or not (3). In the case of a late arrival, the given segment is prioritized, while the associated HAS client is forced to request the lowest video quality so as to allow a fast refill of the buffer (4).

The outlined App-Net mechanisms differ with respect to the implemented monitoring and control blocks. While all of them use application monitoring, the SPM mechanism is

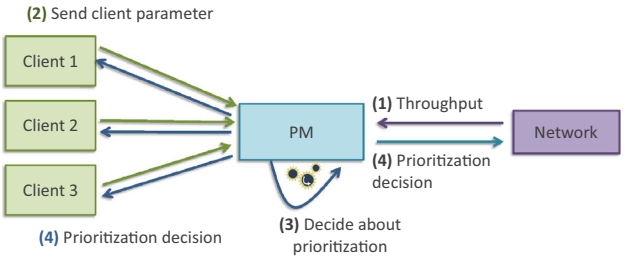


Fig. 6 Illustration of the functionality of SPM (Stalling Prevention Mechanism)

the only one to monitor the application factors on a regular basis. In addition, all three approaches rely on periodic network monitoring. While the other approaches either implement network control (NADE) or application control (QoE-FF), SPM implements functions for both application and network control. We evaluate the described App-Net approaches based on the metrics detailed in the following subsection.

Evaluation metrics

QoE-IFs for HAS-based video streaming are initial buffering delay, video stalling, video playback quality, and the number of quality level switches [3]. Stallings during video playback have a high impact on user's QoE, while initial buffering delay has a minor impact [33]. In the absence of stallings, video reproduction quality on the highest level dominates the user experience [31]. In that case, the number of quality level switches can be neglected. Accordingly, we rely on the overall stalling duration and the average video quality to compare the gains between the considered App-Net mechanisms. The investigated App-Net mechanisms differ in their complexity, which encompasses the number of involved functional entities, extent of their software implementation, and the number of exchanged messages. As a first step, we evaluate this complexity by considering the number of exchanged interaction messages sent via the brokering service.

Implemented testbed and evaluation scenario

Our testbed consists of four common personal computers. First one acts as a HAS server, second one as a network emulator, third one is running a PM and the ØMQ-Broker, while fourth one hosts six instances of the TAPAS HAS client [34]. The TAPAS clients use HAS heuristic *Conventional* [35] (which is a HAS baseline implementation). The HAS server stores the video Big Buck Bunny[3] in three different resolutions (1080p, 720p, 360p). To obtain

[3] https://peach.blender.org/.

different video quality levels for each resolution, we encode the video clip using several bitrates. All bitrates used for video encoding and the resulting video qualities measured with SSIM are shown in Fig. 7. For 360p videos, the encoding bitrate ranges from 100 to 800 kbit/s in steps of 100 kbit/s. Hence, eight quality levels are available for this resolution. For 720p, we provide 16 different qualities by encoding the video with bitrates between 200 and 1700 kbit/s, again in steps of 100 kbit/s. Similarly, the bitrates used for 1080p videos range between 300 and 3000 kbit/s.

We divide the video clip into segments of 4 s long, resulting in 143 segments and a total video length of about 9 min and 30 s. As an example, the following *ffmpeg* command encodes a 360p video with a specific bitrate *$br*, splits it into MPEG-2 TS segments of 4 s long, and creates the corresponding play-list in *m3u8* format:

```
ffmpeg -s:v 640x360 -r 25 -i bbb360.yuv -codec:v
    libx264 -codec:a libfaac -map 0 -segment_time
     4 -segment_list $newdir/out.m3u8 -
    force_key_frames "expr:gte(t,n_forced*4)" -f
    stream_segment -b:v "$(($br * 1000))" $newdir
    /output_%03d.ts
```

The evaluation scenario involves six TAPAS clients running in parallel. We assume the clients to have different screen resolutions, 1080p, 720p and 360p, and consider two clients for each resolution. To emulate the different device capabilities, each client only requests the video play-list of the corresponding resolution, e.g., 1080p devices only request the 1080p video play-list and, consequently, only download 1080p segments. This experimental assumption eases the computation and comparison of clients' SSIM values, as we only have to compare the compressed videos with the original video of the same resolution. Although the clients can not switch between different resolutions, they can still adapt the video quality, since several video bitrates are available for each resolution. We have to note that there is a specific range where video quality switches, either between different bitrates or different resolutions, are acceptable for an end-user. For bitrates below a certain threshold, the end-user will be annoyed, since the video quality or size is too small. Hence, the chosen video quality subset is sufficient to investigate different behavior of the outlined App-Net approaches.

One evaluation run lasts for 7 min, whereas the clients issue a video request randomly within the first 60 s of the run. Four different bandwidth limits are considered. The 1440 kbit/s limit corresponds to a link load of 90% if each client is streaming on the lowest quality available for its resolution. This limit is used to examine the App-Net strategies in a bottleneck scenario. For other experiments, we consider bandwidth limits of 2400, 4800, and 7000 bit/s. These bandwidth scenarios provide sufficient resources for a smooth streaming on moderate and good quality levels.

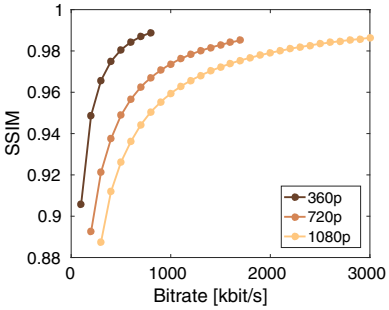


Fig. 7 Bitrates used for video encoding (marked as *dots*) and resulting SSIM values for different resolutions. This also represents the video qualities available at the content server

For each of the App-Net mechanisms, we investigate several configurations. Please note that the settings of one strategy only differ with regards to mechanism-specific, adjustable parameters, and that default configurations of the TAPAS player (e.g., initial buffer and buffer threshold) are not changed. For SPM, we investigate 12 configurations, which combine different values of network monitoring frequency (time intervals of half a second, 2, 4, and 8 s) and a safety margin for the computed segment arrival time (set to 1, 5, and 10% of the calculated value). For QoE-FF, we consider 100, 95, and 90% of the network bandwidth as available for the quality distribution algorithm. For NADE, we investigate two network monitoring frequencies (every second and every 10 s). Tables 2, 3 and 4 summarize the considered configurations for each App-Net mechanism. With respect to the QoE-IFs outlined above, there are only small differences between the settings of one App-Net strategy. For each App-Net mechanism, we, thus, choose one configuration as a representative. The representative configurations are marked bold in Tables 2, 3 and 4. For the sake of clarity, we will only illustrate the

Table 2 Configurations for SPM

	Network monitoring granularity			
Margin	0.5 s	2 s	4 s	8 s
0.01	s_1	s_4	s_7	s_{10}
0.05	s_2	$\boldsymbol{s_5}$	s_8	s_{11}
0.10	s_3	s_6	s_9	s_{12}

Table 3 Configurations for QoE-FF

BW allocation share		
0.90	0.95	1.00
q_1	$\boldsymbol{q_2}$	q_3

Table 4 Configurations for NADE

NMN granularity	
1 s	10 s
n_1	n_2

subset of representative configurations in the following subsections. The results for all settings and all scenarios can be found in Table 5.

For each of the four bandwidth configurations, we perform 15 measurement runs to evaluate the baseline HAS implementation and all configurations of the three App-Net approaches. This results in the total number of 1080 runs, taking about 126 h and providing 6480 client log files.

Evaluation results

In this section, we discuss evaluation results in terms of key QoE indicators and the number of messages required for the investigated App-Net approaches. For the following evaluations, we focus on the phase between the second and the seventh minute of the experiment, i.e. the impact of DASH clients joining the system is neglected. The only exception we make is when examining the initial buffering delay. In this case, we consider the whole streaming session.

Stalling number and buffer filling behavior

In the experiment with the 1440 kbit/s bandwidth limit, the baseline and the considered App-Net implementations strongly differ in terms of video stallings. For higher bandwidth configurations, the stalling occurrence is negligible for all the approaches.

Figure 8 shows the overall number of video freezes for each App-Net mechanism, as well as for the baseline (BL). On average, clients suffer 8.4 stallings when using the baseline HAS mechanism. In case of QoE-FF, the mean number of stallings is increased to 37.7. The other two approaches, SPM and NADE, are able to reduce the number of stallings significantly.

A deeper investigation of the stalling behavior reveals that in the cases of BL and QoE-FF, the high resolution devices suffer the majority of video interruptions. This is particularly obvious for QoE-FF. When applying this mechanism, 37.7 stallings occur on average, whereby the 1080p devices contribute with 60.4, and the 720p devices with 52.7 stallings. The 360p devices can almost stream smoothly without any interruptions.

The stallings occur due to an insufficient buffer filling. The video buffer, or buffered playtime, is the portion of video data already available at the client, but not yet played back. Re-buffering is necessary if the buffer drains out, causing a video interruption that lasts until the next video segment arrives at the client. We depict in Fig. 9 the buffered playtime per device type to get a better understanding of the unfair stalling distribution when QoE-FF is used.

The y-axis represents the buffered playtime in seconds, the x-axis shows the strategies. The black dashed lined indicates the buffer threshold of 15 s configured at the DASH controller. As long as a client has a buffer below this threshold, it continues requesting segments. Otherwise, the client enters the idle state. Figure 9a shows the outcomes for the 1440 kbit/s bandwidth limit. In case of baseline, the 360p and 720p devices have a fairly high average buffer of 13.9 and 13.47 s. The 1080p devices, however, have a mean buffer of only 3.74 s. If QoE-FF is applied, the mean buffered playtime of the smallest device type nearly reaches the maximum buffer (14.5 s). Whereas, only 2.22 s are buffered at the 720p device and 1.54 s at the 1080p device. This is in line with the high number of stallings for these device types. SPM is capable to balance the buffered playtime among the different devices Despite its high resolution, a 1080p device has an average buffer of 6.63 s. The impact of the bandwidth slice reservation performed under NADE is clearly visible: All client-types have a similar average buffer. Contrary to all aforementioned strategies, the 1080p devices have the highest buffer filling level (13.63 s). On average, the 360p devices buffer 12.4 s of video content, the 720p device have a similar buffer length of 12.9 s.

The unbalanced buffer state in case of QoE-FF results in a high quantity of video interruptions for larger devices, particularly in low bandwidth scenarios. To explain this behavior, we distinguish two different phases for this mechanism. During the first phase, the clients constantly request video segments to quickly fill up the buffer to a predefined threshold. During the second phase, the clients request segments in a on-off manner, to keep the buffer on a level which allows both, a smooth playback and a fast reaction on changing network conditions. During the first phase, where video segments are constantly downloaded for a fast buffer filling, the diverse device types show different behaviors. The devices with a small resolution are instructed by the PM to request lower qualities in order to achieve a fair level of video quality among all clients. As the network resources are shared in a fair manner across all clients, these devices can faster build up their video buffer. The higher resolution devices, however, are directed by the PM to request higher bitrates to achieve fairness among the clients. As a consequence, these devices are not able to fill their buffer up to the threshold, continue to constantly

Table 5 Obtained results for the baseline implementation and all App-Net mechanism configurations. *F* denotes Fairness (in terms of SSIM and stallings). All results, except fairness indices, are mean values along with the 95% confidence interval

Bandwidth (kbit/s)	Mechanism	Mean SSIM	Min. SSIM	Iniital delay (s)	Stalling	Stallingdur. (s)	Buffer (s)	Quality-switches	F (SSIM)	F (stallings)
1440	*bl*	0.904 ± 0.004	0.888 ± 0.000	9.51 ± 0.73	8.38 ± 2.71	4.901 ± 1.61	10.37 ± 1.00	2.50 ± 0.54	0.62	0.60
	q_1	0.913 ± 0.001	0.906 ± 0.000	13.46 ± 1.19	36.70 ± 5.64	42.824 ± 8.43	6.08 ± 1.25	0.04 ± 0.06	0.88	0.16
	q_2	0.913 ± 0.001	0.906 ± 0.000	15.19 ± 1.75	37.71 ± 5.69	44.013 ± 8.58	6.09 ± 1.26	0.10 ± 0.08	0.88	0.15
	q_3	0.913 ± 0.001	0.906 ± 0.000	15.43 ± 1.53	37.62 ± 5.69	43.971 ± 8.59	6.07 ± 1.25	0.17 ± 0.11	0.88	0.15
	s_1	0.901 ± 0.003	0.888 ± 0.000	9.33 ± 0.76	0.23 ± 0.12	0.298 ± 0.25	11.38 ± 0.66	3.78 ± 0.85	0.72	0.98
	s_2	0.901 ± 0.003	0.888 ± 0.000	9.23 ± 0.65	0.31 ± 0.16	0.362 ± 0.25	11.10 ± 0.66	3.69 ± 0.75	0.71	0.98
	s_3	0.901 ± 0.003	0.888 ± 0.000	9.24 ± 0.67	0.24 ± 0.12	0.313 ± 0.22	11.40 ± 0.61	3.62 ± 0.71	0.72	0.98
	s_4	0.901 ± 0.003	0.888 ± 0.000	8.96 ± 0.67	0.19 ± 0.10	0.307 ± 0.22	11.40 ± 0.68	3.53 ± 0.70	0.71	0.98
	s_5	0.901 ± 0.003	0.888 ± 0.000	8.88 ± 0.63	0.50 ± 0.27	0.893 ± 0.73	11.07 ± 0.70	3.73 ± 0.69	0.71	0.96
	s_6	0.901 ± 0.003	0.888 ± 0.000	9.02 ± 0.67	0.12 ± 0.08	0.224 ± 0.25	11.38 ± 0.65	3.20 ± 0.70	0.72	0.99
	s_7	0.901 ± 0.003	0.888 ± 0.000	8.40 ± 0.69	0.20 ± 0.10	0.213 ± 0.19	11.33 ± 0.70	3.80 ± 0.76	0.70	0.99
	s_8	0.901 ± 0.003	0.888 ± 0.000	8.20 ± 0.69	0.13 ± 0.08	0.200 ± 0.22	11.52 ± 0.65	3.41 ± 0.83	0.71	0.99
	s_9	0.901 ± 0.003	0.888 ± 0.000	8.84 ± 0.93	0.20 ± 0.12	0.380 ± 0.29	11.32 ± 0.66	3.74 ± 0.81	0.70	0.98
	s_{10}	0.901 ± 0.003	0.888 ± 0.000	7.49 ± 0.68	0.22 ± 0.14	0.410 ± 0.30	11.27 ± 0.67	3.82 ± 0.81	0.71	0.98
	s_{11}	0.901 ± 0.003	0.888 ± 0.000	7.73 ± 0.85	0.17 ± 0.08	0.369 ± 0.29	11.31 ± 0.62	3.81 ± 0.77	0.72	0.99
	s_{12}	0.901 ± 0.003	0.888 ± 0.000	7.02 ± 0.59	0.11 ± 0.09	0.231 ± 0.23	11.70 ± 0.59	3.43 ± 0.69	0.72	0.99
	n_1	0.899 ± 0.001	0.893 ± 0.001	14.49 ± 1.38	0.02 ± 0.03	0.016 ± 0.02	12.97 ± 0.15	6.82 ± 1.39	0.88	1.00
	n_2	0.902 ± 0.002	0.893 ± 0.001	24.68 ± 4.14	2.56 ± 1.36	11.096 ± 7.02	11.54 ± 0.71	8.71 ± 1.81	0.83	0.80
2400	*bl*	0.937 ± 0.006	0.906 ± 0.002	7.30 ± 0.47	0.29 ± 0.14	0.700 ± 0.36	13.18 ± 0.06	30.00 ± 2.10	0.49	0.97
	q_1	0.927 ± 0.002	0.913 ± 0.002	9.82 ± 0.82	0.13 ± 0.08	0.130 ± 0.09	14.26 ± 0.26	6.77 ± 1.15	0.79	0.98
	q_2	0.927 ± 0.002	0.911 ± 0.002	9.74 ± 0.99	0.24 ± 0.20	0.156 ± 0.14	13.95 ± 0.37	3.96 ± 0.83	0.77	0.96
	q_3	0.931 ± 0.002	0.920 ± 0.005	10.27 ± 0.79	3.10 ± 1.95	2.164 ± 1.34	12.17 ± 0.83	4.08 ± 0.87	0.82	0.59
	s_1	0.937 ± 0.006	0.904 ± 0.001	7.45 ± 0.39	0.00 ± 0.00	0.000 ± 0.00	13.23 ± 0.03	31.56 ± 1.42	0.48	1.00
	s_2	0.938 ± 0.006	0.905 ± 0.002	7.56 ± 0.43	0.00 ± 0.00	0.000 ± 0.00	13.17 ± 0.04	34.93 ± 1.38	0.48	1.00
	s_3	0.937 ± 0.006	0.905 ± 0.002	7.50 ± 0.50	0.09 ± 0.10	0.096 ± 0.16	13.14 ± 0.07	32.02 ± 1.83	0.49	0.98
	s_4	0.937 ± 0.005	0.906 ± 0.002	7.06 ± 0.51	0.02 ± 0.03	0.041 ± 0.06	13.20 ± 0.04	32.64 ± 2.04	0.49	0.99
	s_5	0.937 ± 0.006	0.905 ± 0.001	7.06 ± 0.50	0.00 ± 0.00	0.000 ± 0.00	13.23 ± 0.04	31.47 ± 1.90	0.48	1.00
	s_6	0.937 ± 0.006	0.905 ± 0.002	6.99 ± 0.47	0.01 ± 0.02	0.011 ± 0.02	13.22 ± 0.04	30.93 ± 1.56	0.48	1.00
	s_7	0.938 ± 0.006	0.906 ± 0.002	6.66 ± 0.54	0.00 ± 0.00	0.000 ± 0.00	13.17 ± 0.04	33.64 ± 1.61	0.49	1.00
	s_8	0.937 ± 0.006	0.906 ± 0.001	6.65 ± 0.55	0.01 ± 0.02	0.007 ± 0.01	13.18 ± 0.04	33.08 ± 1.46	0.49	1.00
	s_9	0.937 ± 0.006	0.906 ± 0.002	6.76 ± 0.55	0.00 ± 0.00	0.000 ± 0.00	13.17 ± 0.04	34.58 ± 1.55	0.48	1.00
	s_{10}	0.938 ± 0.005	0.907 ± 0.001	5.58 ± 0.53	0.00 ± 0.00	0.000 ± 0.00	13.16 ± 0.04	34.30 ± 1.55	0.49	1.00
	s_{11}	0.937 ± 0.006	0.905 ± 0.001	5.37 ± 0.47	0.01 ± 0.02	0.021 ± 0.04	13.17 ± 0.05	33.16 ± 1.56	0.49	1.00
	s_{12}	0.937 ± 0.006	0.906 ± 0.001	5.42 ± 0.53	0.00 ± 0.00	0.000 ± 0.00	13.21 ± 0.05	31.53 ± 1.61	0.48	1.00
	n_1	0.929 ± 0.003	0.909 ± 0.002	10.49 ± 1.18	0.04 ± 0.06	0.037 ± 0.07	13.51 ± 0.09	25.01 ± 2.53	0.74	0.99
	n_2	0.929 ± 0.003	0.908 ± 0.003	17.14 ± 3.51	0.92 ± 1.05	4.284 ± 4.74	13.00 ± 0.49	26.00 ± 2.77	0.72	0.78

Table 5 continued

Bandwidth (kbit/s)	Mechanism	Mean SSIM	Min. SSIM	Iniital delay (s)	Stalling	Stallingdur. (s)	Buffer (s)	Quality-switches	F (SSIM)	F (stallings)
4800	bl	0.969 ± 0.003	0.949 ± 0.001	5.09 ± 0.48	0.21 ± 0.11	0.304 ± 0.20	13.08 ± 0.07	34.29 ± 2.26	0.68	0.97
	q_1	0.963 ± 0.001	0.961 ± 0.003	7.25 ± 0.65	0.11 ± 0.07	0.181 ± 0.17	14.60 ± 0.25	17.24 ± 2.47	0.93	0.98
	q_2	0.963 ± 0.001	0.961 ± 0.004	7.40 ± 0.75	0.08 ± 0.06	0.101 ± 0.10	14.23 ± 0.29	14.53 ± 2.48	0.91	0.98
	q_3	0.966 ± 0.000	0.965 ± 0.000	8.02 ± 0.77	0.83 ± 1.04	0.540 ± 0.65	13.53 ± 0.48	13.49 ± 2.19	0.98	0.72
	s_1	0.969 ± 0.003	0.949 ± 0.001	5.26 ± 0.48	0.00 ± 0.00	0.000 ± 0.00	13.13 ± 0.04	33.18 ± 1.96	0.68	1.00
	s_2	0.968 ± 0.003	0.949 ± 0.000	5.10 ± 0.46	0.02 ± 0.03	0.030 ± 0.05	13.18 ± 0.05	31.91 ± 1.99	0.68	0.99
	s_3	0.969 ± 0.003	0.949 ± 0.001	5.09 ± 0.43	0.02 ± 0.03	0.076 ± 0.11	13.12 ± 0.05	33.77 ± 2.05	0.68	0.99
	s_4	0.969 ± 0.003	0.949 ± 0.001	5.00 ± 0.49	0.02 ± 0.03	0.048 ± 0.07	13.11 ± 0.05	33.27 ± 1.97	0.68	0.99
	s_5	0.969 ± 0.003	0.949 ± 0.001	5.07 ± 0.51	0.01 ± 0.02	0.012 ± 0.02	13.10 ± 0.05	34.53 ± 2.27	0.68	0.99
	s_6	0.969 ± 0.003	0.949 ± 0.001	4.82 ± 0.44	0.02 ± 0.03	0.023 ± 0.03	13.11 ± 0.05	33.41 ± 2.11	0.68	0.99
	s_7	0.969 ± 0.003	0.949 ± 0.001	4.79 ± 0.53	0.01 ± 0.02	0.012 ± 0.02	13.16 ± 0.04	32.84 ± 1.97	0.68	0.99
	s_8	0.969 ± 0.003	0.949 ± 0.000	4.89 ± 0.57	0.02 ± 0.03	0.014 ± 0.02	13.11 ± 0.05	35.02 ± 2.25	0.68	0.99
	s_9	0.969 ± 0.003	0.949 ± 0.001	4.72 ± 0.59	0.11 ± 0.14	0.169 ± 0.22	13.08 ± 0.08	34.37 ± 2.23	0.68	0.96
	s_{10}	0.969 ± 0.003	0.949 ± 0.001	4.01 ± 0.56	0.06 ± 0.07	0.098 ± 0.14	13.08 ± 0.06	35.40 ± 2.19	0.69	0.98
	s_{11}	0.969 ± 0.003	0.949 ± 0.001	3.92 ± 0.57	0.23 ± 0.29	0.390 ± 0.57	13.02 ± 0.13	34.72 ± 2.22	0.69	0.92
	s_{12}	0.968 ± 0.003	0.949 ± 0.001	3.91 ± 0.52	0.03 ± 0.05	0.026 ± 0.05	13.11 ± 0.05	34.63 ± 2.10	0.68	0.99
	n_1	0.964 ± 0.001	0.954 ± 0.002	7.28 ± 0.76	0.06 ± 0.05	0.126 ± 0.13	13.16 ± 0.07	36.23 ± 2.43	0.86	0.99
	n_2	0.964 ± 0.002	0.952 ± 0.007	11.98 ± 2.75	0.47 ± 0.77	1.171 ± 2.17	13.02 ± 0.27	38.34 ± 2.18	0.83	0.79
7000	bl	0.980 ± 0.002	0.968 ± 0.001	3.80 ± 0.45	0.27 ± 0.13	0.442 ± 0.27	13.33 ± 0.15	30.74 ± 4.64	0.82	0.97
	q_1	0.972 ± 0.001	0.971 ± 0.002	5.47 ± 0.55	0.19 ± 0.10	0.268 ± 0.18	14.81 ± 0.21	19.32 ± 3.21	0.93	0.98
	q_2	0.973 ± 0.001	0.971 ± 0.004	5.68 ± 0.63	0.11 ± 0.08	0.201 ± 0.19	14.64 ± 0.27	19.92 ± 3.49	0.90	0.98
	q_3	0.974 ± 0.001	0.973 ± 0.002	5.90 ± 0.61	0.10 ± 0.08	0.189 ± 0.18	14.45 ± 0.32	20.00 ± 3.59	0.94	0.98
	s_1	0.980 ± 0.002	0.968 ± 0.001	4.11 ± 0.44	0.03 ± 0.04	0.249 ± 0.32	13.29 ± 0.16	33.59 ± 4.91	0.82	0.99
	s_2	0.980 ± 0.002	0.967 ± 0.001	3.99 ± 0.45	0.02 ± 0.03	0.051 ± 0.08	13.33 ± 0.16	32.18 ± 4.85	0.82	0.99
	s_3	0.979 ± 0.002	0.967 ± 0.001	4.22 ± 0.47	0.01 ± 0.02	0.079 ± 0.16	13.37 ± 0.15	32.24 ± 4.77	0.82	0.99
	s_4	0.980 ± 0.002	0.967 ± 0.001	4.76 ± 1.47	0.01 ± 0.02	0.019 ± 0.04	13.32 ± 0.15	32.61 ± 4.76	0.82	0.99
	s_5	0.980 ± 0.002	0.968 ± 0.001	4.50 ± 0.83	0.03 ± 0.04	0.072 ± 0.09	13.31 ± 0.15	33.26 ± 4.94	0.82	0.99
	s_6	0.980 ± 0.002	0.968 ± 0.001	3.95 ± 0.50	0.03 ± 0.04	0.010 ± 0.01	13.31 ± 0.16	35.06 ± 5.11	0.82	0.99
	s_7	0.980 ± 0.002	0.967 ± 0.000	3.82 ± 0.53	0.01 ± 0.02	0.049 ± 0.10	13.34 ± 0.15	31.81 ± 4.79	0.82	0.99
	s_8	0.979 ± 0.002	0.967 ± 0.001	3.84 ± 0.50	0.01 ± 0.02	0.121 ± 0.22	13.32 ± 0.15	33.10 ± 4.82	0.82	0.99
	s_9	0.980 ± 0.002	0.968 ± 0.001	3.75 ± 0.52	0.02 ± 0.03	0.041 ± 0.06	13.29 ± 0.16	33.16 ± 4.88	0.82	0.99
	s_{10}	0.979 ± 0.002	0.967 ± 0.000	3.38 ± 0.56	0.00 ± 0.00	0.000 ± 0.00	13.32 ± 0.15	32.94 ± 4.93	0.82	1.00
	s_{11}	0.979 ± 0.002	0.967 ± 0.001	4.25 ± 1.50	0.02 ± 0.03	0.029 ± 0.05	13.32 ± 0.15	32.68 ± 4.82	0.81	0.99
	s_{12}	0.979 ± 0.002	0.967 ± 0.001	3.31 ± 0.49	0.01 ± 0.02	0.003 ± 0.01	13.28 ± 0.16	34.67 ± 5.06	0.82	0.99
	n_1	0.976 ± 0.001	0.972 ± 0.001	5.32 ± 0.55	0.10 ± 0.08	0.116 ± 0.09	13.07 ± 0.07	40.26 ± 2.34	0.93	0.98
	n_2	0.976 ± 0.002	0.968 ± 0.010	8.59 ± 1.58	0.47 ± 0.84	1.110 ± 2.13	12.88 ± 0.27	42.77 ± 2.46	0.84	0.79

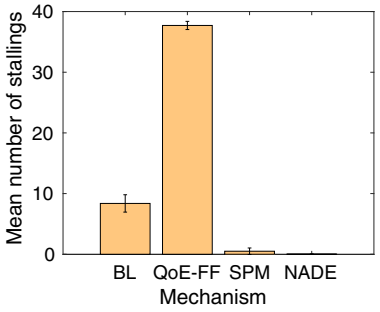
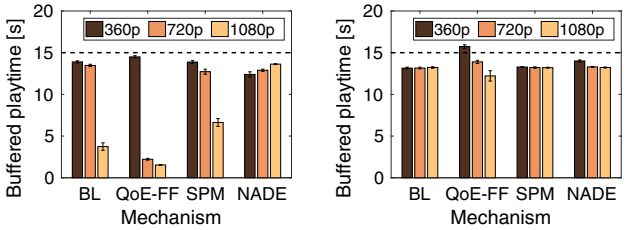


Fig. 8 Mean number of stallings per client at the bandwidth limit of 1440 kbit/s



(a) Bandwidth limit: 1440 kbit/s.

(b) Bandwidth limit: 2400 kbit/s.

Fig. 9 Illustration of the buffered video playtime with respect to different device types and different bandwidth limits

request video segments, and do not enter idle states. Hence, the resources, which are shared in a QoS fair manner, do not suffice for the clients with larger devices to fill their buffer sufficiently for a smooth video playback. In the second phase, all clients have filled their buffers up to the threshold. Hence, they request different bitrates in an on-off manner, achieving a fair share of network resources with respect to video quality. As the second phase is not reached in this bottleneck scenario during our examined period, the devices with the higher resolutions suffer a large number of stallings. In a conventional scenario, the DASH heuristic would prevent this issue by switching to a lower quality for a faster transmission of video segments. However, the client-side DASH heuristic is displaced by the centralized PM application control, which does not consider a client's video buffer. Monitoring the video buffers can enhance the mechanism's performance w.r.t. stallings in bottleneck scenarios. Then, the PM can detect critical buffer states and decide about a lower quality for the affected clients, to support a smooth playback on the expense of a lower video quality fairness.

To summarize, NADE and SPM enable a smooth video playback even in low bandwidth scenarios, while the baseline approach and QoE-FF do not facilitate a smooth playback without stallings.

In scenarios with higher bandwidth limits, all mechanisms and the baseline allow all clients to sufficiently fill the video buffer. The outcomes of the experiments with a limit of 2400 kbit/s are shown in Fig. 9b. Please note that the results for the remaining bandwidth limits (4800, 7000 kbit/s) are hardly different and for that reason are not shown in the paper. Furthermore, the stallings are negligible for bandwidths of at least 2400 kbit/s and not investigated in detail here.

Mean SSIM and SSIM fairness

The average SSIM of all clients and the fairness in terms of SSIM are illustrated in Figure 10.

The x-axis represents the mean SSIM, the y-axis the shows the SSIM fairness.

For a bandwidth limit of 1440 kbit/s, each App-Net strategy outperforms the baseline's SSIM fairness of $F_{SSIM} = 0.62$. NADE and QoE-FF yield similar values of about $F_{SSIM} = 0.88$. When SPM is applied, the resulting fairness is $F_{SSIM} = 0.71$. In case of SPM and NADE, the mean SSIM is reduced while the video quality fairness is enhanced. In case of QoE-FF, the mean SSIM and the fairness are enhanced. This, however, comes at the costs of an increased number of stallings, which is illustrated in Fig. 8.

For a bandwidth limit of 2400 kbit/s, SPM and baseline yield similar results for both, the mean SSIM and the fairness in terms of SSIM. NADE enhances the baseline's fairness by 0.25, resulting in an index value of $F_{SSIM} = 0.74$. QoE-FF achieves an SSIM fairness of $F_{SSIM} = 0.77$. However, for both approaches, the mean SSIM is reduced. In case of NADE, which yields a slightly lower fairness than QoE-FF, the average video quality is decreased to a lesser extent. A similar behavior is observable for the 4800 kbit/s scenario. In case of a 7000 kbit/s bandwidth limit, the baseline's fairness of $F_{SSIM} = 0.82$ is increased by NADE and QoE-FF to a value of roughly $F_{SSIM} = 0.93$.

The results clearly indicate the potential of QoE-FF and NADE to enhance the video quality fairness among heterogeneous clients. However, in the absence of stallings, this enhancement comes on the expense of a reduced mean SSIM of all clients.

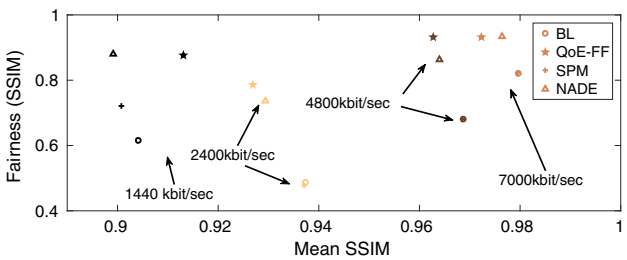


Fig. 10 Video quality fairness versus mean SSIM for different bandwidth limits

Initial delay

This section targets the results with respect to the initial delay. In the configuration with a bandwidth limit of 1440 kbit/s, QoE-FF and NADE enlarge the initial delay. The initial delay for baseline is on average 9.5 s. SPM slightly reduces this value to 8.88 s. When applying QoE-FF, the users have to wait 15.19 s on average until the video playback starts. NADE enlarges the initial delay to 14.49 s on average. To identify a possible correlation between device type and initial delay, we perform a deeper investigation of the initial delay for the different mechanisms with respect to the clients' device resolutions. The results for all bandwidth limits are depicted in Fig. 11.

For the 1440 kbit/s bandwidth limit (Fig. 11a), baseline and SPM both show the behavior that devices with higher resolution need more time to fill the initial buffer level. Users of a 360p device have to wait on average 6.62 s until the video starts. Users of a 1080p device have to wait roughly twice as long. SPM slightly shortens the 1080p device's initial time to an average of 11 s. For the remaining device types, no significant differences are observable. When QoE-FF is applied, the 1080p device has a mean initial delay of 19.65 s. The initial delay is enlarged for the other device types, too. Due to the overlapping confidence intervals, no conclusion can be drawn whether NADE preserves the trend of longer initial delay for higher resolution devices. However, the average initial waiting time is increased for the 360p and 720p devices compared to the baseline.

For higher bandwidth limits (Fig. 11b–d), all initial delays decrease. When baseline and SPM are applied, the 360p devices are capable of starting playback after a short waiting time, 720p and 1080p devices need slightly more time. This is similar for QoE-FF, however, for the 2400 kbit/s limit, it cannot be concluded whether 720p or 1080p devices have a longer intial waiting time, as the confidence intervals overlap. In the case of NADE, no conclusions can be drawn whether the initial waiting time is correllated with the screen resolution. This holds for all investigated bandwidth limits

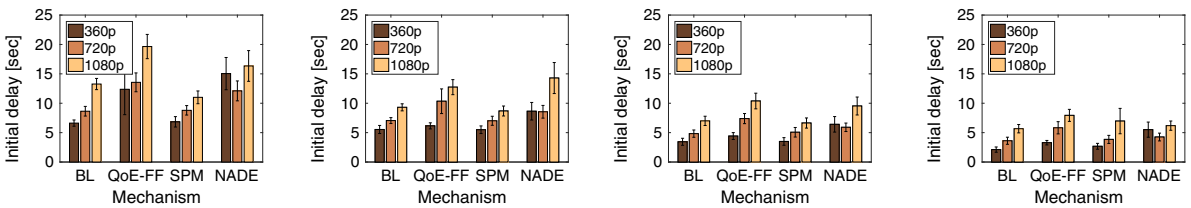
To summarize, SPM and baseline provide the smallest initial delay. Further, the high resolution devices experience a larger initial delay than devices with smaller resolutions for all investigated cases. Further, the initial waiting time increases for higher resolutions if baseline or SPM are applied. This holds for all of the four investigated scenarios.

Number of exchanged messages

Based on the monitoring and control capabilities of a particular App-Net approach, different information is exchanged with the PM with different granularity. Consequently, the total number of exchanged monitoring and control messages differs between the mechanisms and their specific configurations.

We examine in the following the total amount of exchanged control messages and the share each of four different message types contributes. The types are defined by a message's purpose. Hence, we differentiate network monitoring (NMN) and application monitoring (AMN) messages as well as the respective messages for controlling, namely NCT and ACT messages.

We depict two different configurations for each strategy, in order to highlight configuration-specific characteristics. The bandwidth limit is set to 4800 kbit/s. The illustrated behavior is representative for all other bandwidth configurations, where the number of exchanged messages slightly differs. The number of exchanged messages illustrated in Fig. 12 covers the time between 120 and 360 s of the experiment. Hence, the transient phase is excluded. In case of QoE-FF, the AMN messages serve as keep-alive messages and are sent from the clients to the PM before each GET request. As the number of GET requests is not influenced by the underlying QoE-FF configuration, no difference concerning the amount of this type of message is observable. In our implementation, network monitoring information is transmitted using the keep-alive messages (AMN). Since this information is logically separated from the AMN messages, they are illustrated in the figure. For both configurations of QoE-FF, q_2 and q_3, around 360 AMN and NMN messages are exchanged. The



(a) Bandwidth limit: 1440 kbit/s. **(b)** Bandwidth limit: 2400 kbit/s. **(c)** Bandwidth limit: 4800 kbit/s. **(d)** Bandwidth limit: 7000 kbit/s.

Fig. 11 Illustration of the initial delay with respect to different device types and different bandwidth limits

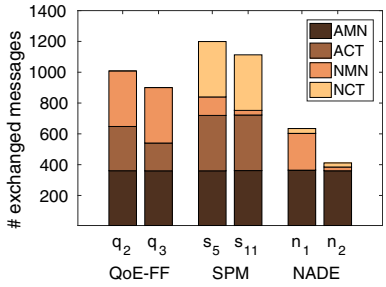


Fig. 12 Number of exchanged messages, distinguished by the message purpose. Application Monitoring (AMN), Application Control (ACT), Network Monitoring (NMN), Network Control (NCT)

configuration affects the number of exchanged ACT messages. On average, 287 messages of this type are sent when applying q_2, where 95% of the bandwidth is considered at the algorithm. When configuration q_3 is applied, where the whole bandwidth is considered to be available, 180 ACT messages are sent on average. QoE-FF does not perform network control. Hence, no NCT messages are exchanged when using this strategy. In total, 1008 messages are exchanged for q_2 and 900 for q_3

For the SPM configuration s_5, the network throughput is polled each 2 s, for the configuration s_{11}, this time interval is enlarged to 8 s. Consequently, the s_5 configuration leads to a higher number of NMN messages as compared to s_{11}. After each decision, which is triggered by a client's AMN message before its GET request, both, network and application, are informed about the outcome. As the number of GET requests hardly differs for different configurations, there is only minor impact on the number of AMN, ACT, and NCT messages. In total, 1191 messages are sent when s_5 is applied, 1113 when s_{11} is applied.

For NADE configuration n_1, where the throughput is polled once per second, 239 NMN messages are sent. With a network monitoring granularity of 10 s only (n_2), 24 messages of this type are exchanged. 31.5 NCT messages are exchanged with configuration n_1 and 26.5 for the second configuration. As the AMN messages are sent before each GET request, their number is not influenced by the configuration. Since no application control is performed using NADE, no ACT messages are exchanged. This leads to an overall number of 635 (n_1) and 411 (n_2) messages.

All examined strategies exchange a similar number of messages of the type AMN. However, whilst SPM requires each of the four message classes, there is no need for NCT messages in case of QoE-FF, and no need for ACT messages if NADE is applied.

Holistic comparison of the investigated mechanisms

In the following, the three presented App-Net strategies are compared to each other with respect to several metrics and under different network conditions. To illustrate the differences between the investigated mechanisms we rely on spider plots.

Figure 13 depicts the overall performance of baseline and App-Net strategies for a bandwidth limit of 1440 kbit/s.

QoE-FF clearly outperforms the other strategies in terms of minimum SSIM, mean SSIM, and the number of quality switches. Further, it enhances the fairness in terms of video quality to a similar extent as NADE. However, QoE-FF drastically enlarges the number of stallings, the overall stalling duration, and the initial delay.

NADE leads to an increase of the initial delay, too. It yields the highest number of quality switches but can enhance the baseline's minimum SSIM. At the expense of the mean quality, NADE also enhances the video quality fairness. Furthermore, it is capable of reducing the number of stalling events and the stalling duration compared to the baseline implementation.

This also holds for SPM. Besides the stalling reduction and the overall stalling time, SPM is also capable to slightly shorten the initial delay. However, SPM leads to a higher number of quality switches and reduces the mean SSIM.

In each dimension, the baseline implementation can be surpassed by at least one of the strategies. However, none of the strategies can accomplish an improvement of the baseline with respect to all evaluated metrics.

Using the implemented QoE-FF strategy in the outlined scenario results in a severe degradation of the user experience due to the high number of stallings. This might even lead to user abandonment after experiencing a few stallings. NADE should be applied in a scenario like this, as it

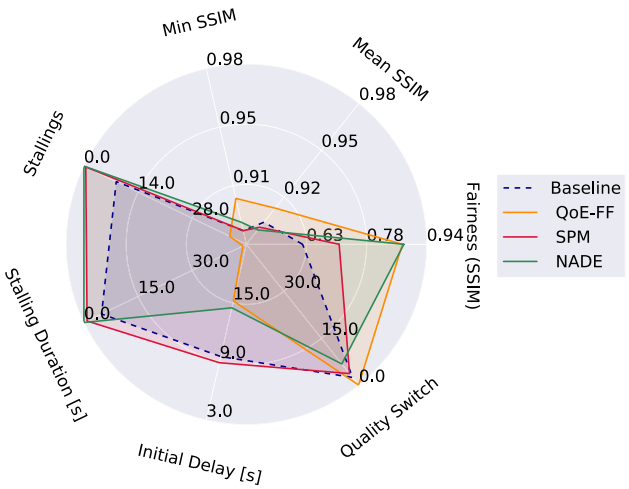


Fig. 13 Illustration of the results for baseline implementation and the three App-Net strategies with respect to several QoE-IFs. Bandwidth limit 1440 kbit/s

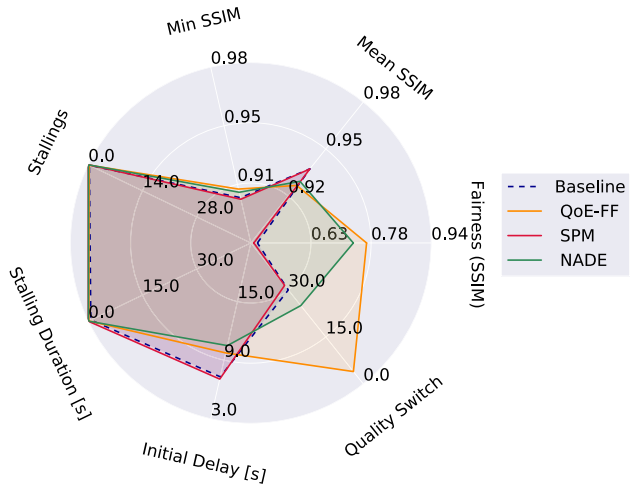


Fig. 14 Illustration of the results for baseline implementation and the three App-Net strategies with respect to several QoE-IFs. Bandwidth limit 2400 kbit/s

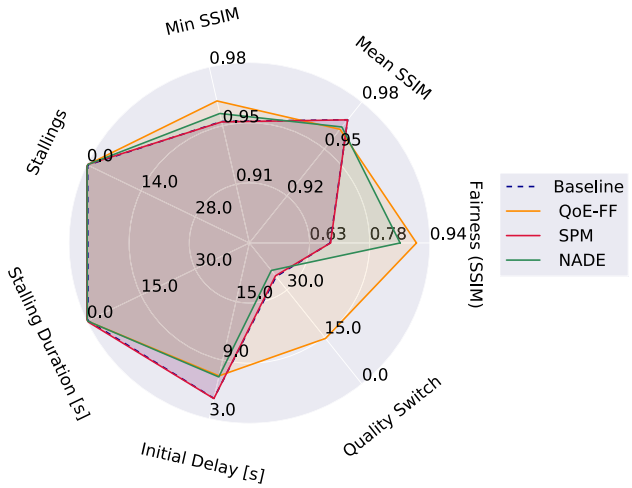


Fig. 15 Illustration of the results for baseline implementation and the three App-Net strategies with respect to several QoE-IFs. Bandwidth limit 4800 kbit/s

achieves both, a high SSIM fairness and a smooth streaming.

Figure 14 shows the results for a bandwidth limit of 2400 kbit/s.

The Quality of Experience Fairness Framework outperforms baseline and the other approaches with regard to video quality fairness, the number of quality switches, and minimum SSIM. However, it has the lowest mean SSIM. On average, there is less than a half stalling for any of the investigated solutions. The Stalling Prevention Mechanism is capable to stream without any interruption, and hence, slightly improves the baseline. Except from stalling number and duration, SPM and baseline hardly differ in their outcomes. NADE also enhances the video quality fairness, however, not to such an extent as QoE-FF does. In return, NADE yields a higher mean SSIM than QoE-FF. However, with an initial delay of 10.5 s on average, NADE needs the longest time until the video playback starts. As QoE-FF and NADE both enable smooth streaming whilst being fair, one of these mechanisms should be applied. Effectively, SPM has no significant positive impact, as the available network resources nearly allow an interruption-free playback.

Figure 15 shows the outcomes for a higher bandwidth limit of 4800 kbit/s.

As the fairly high bandwidth limit of 4800 kbit/s triggers hardly any actions for the Stalling Prevention Mechanism, its results are quite similar to the baseline's outcome. However, the baseline's small stalling number and stalling duration can still be reduced by using SPM. SPM and baseline achieve the shortest initial delay and the highest mean SSIM. Contrary, they suffer under a lower video quality fairness compared to QoE-FF and NADE.

QoE-FF achieves the highest SSIM fairness. In consequence of providing all clients a similar video quality, the minimum SSIM increases and the overall mean SSIM decreases. Compared to the baseline and all other App-Net strategies, QoE-FF enlarges the initial delay but also yields the fewest quality switches.

NADE also enhances the SSIM fairness and the minimum SSIM for the sake of a lower mean SSIM. It has the highest number of quality switches among all examined strategies and enlarges the initial delay compared to the baseline implementation. For this bandwidth configuration, we propose to apply NADE or QoE-FF as both of them enhance the SSIM fairness and slightly reduce the number of stallings compared to the baseline.

Figure 16 illustrates the results for the highest bandwidth limit of 7000 kbit/s.

In this case, NADE is the mechanism that yields the highest fairness with respect to video quality. However, it also causes the highest number of quality switches. QoE-FF outperforms NADE, SPM, and the baseline with respect to the quality switches. The resulting fairness index for QoE-FF is lower than for NADE. For both mechanisms, there is less than half a stalling event on average per client, they are both suitable in this situation. Again, SPM should not be applied, as it cannot significantly reduce more stallings than the other mechanisms, but enlarges the initial delay and the number of quality switches.

Conclusion

This paper is a first step towards a systematic comparison of different QoE-centric App-Net interaction mechanisms. Firstly, we give an overview of such approaches and use

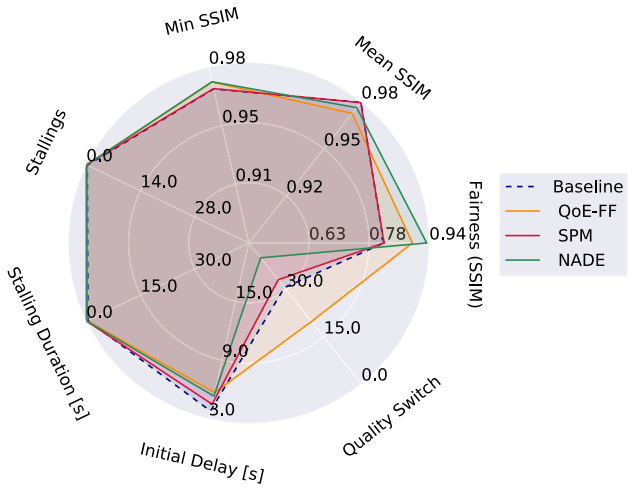


Fig. 16 Illustration of the results for baseline implementation and the three App-Net strategies with respect to several QoE-IFs. Bandwidth limit: 7000 kbit/s

them to derive an abstract interaction model based on the identified functional blocks. Secondly, we evaluate three App-Net approaches for adaptive video streaming.

We show that the investigated approaches differ with respect to the number of exchanged messages and their influence on stalling and average video quality. The stalling prevention mechanism requires a constant message exchange and is thus very costly. It reduces stalling if little network resources are available, i.e., if all clients can barely be supplied with the lowest quality. Its impact on the user-centric metrics is negligible if enough network resources for higher video qualities are available. The QoE fairness framework provides a fair video quality among heterogeneous clients for the price of a lower average video quality. As long as enough network resources are available, no stalling times occur. If the lowest video quality can be barely supplied, large stalling times are observed, indicating that the approach should not be used in such a scenario. The network assisted DASH fairness enhancement provides similar, slightly smaller results compared to the QoE fairness framework in terms of video quality fairness and average video quality. However, it allows a smooth video playback for low bandwidth scenarios. The results further indicate, that for a constant bandwidth scenario, the stalling prevention mechanism is very costly in terms of exchanged messages. The smallest amount of messages is exchanged when using the network assisted DASH fairness enhancement. Accordingly, this approach seems to be the most reasonable solution for constant bandwidth scenarios.

The presented results illustrate the importance of a detailed understanding of application–network interaction approaches and of identifying corresponding use-cases for them. The introduced framework is the first step towards a holistic reproducible evaluation of App-Net approaches that focus on improving QoE. For that, we provide an environment and the respective building blocks that allow an investigation of a variety of App-Net mechanisms. Firstly, the implemented ØMQ messaging broker facilitates the interaction of distributed components that execute monitoring and control functions. This messaging broker is easy to set up and enables a straightforward integration of additional entities and algorithms, which simplifies extensions of the framework with new mechanisms and its use for different applications. Secondly, we provide a basic set of concrete implementations of monitoring and control functions for DASH-based video streaming, as well as a corresponding evaluation methodology that combines various QoE indicators for DASH. Hence, we encourage the community to extend our work by integrating new App-Net functions and mechanisms, comparing them with the existing ones, and evaluating other multimedia applications.

Acknowledgements This work has been partially supported by the ICT COST Action IC1304 "Autonomous Control for a Reliable Internet of Services (ACROSS)", Nov 2013–Nov 2017, funded by the European Union, as well as by the DFG Grant "SDN-enabled Application-aware Network Control Architectures and their Performance Assessment" (ZI1334/2-1, TR257/43-1).

References

1. "Qualinet White Paper on Definitions of Quality of Experience (2012)" (2013) European network on quality of experience in multimedia systems and services (COST Action IC 1003). In: Le Callet P, Möller S, Perkis A (eds) Lausanne, Switzerland, version 1.2
2. Varela M, Skorin-Kapov L, Moor KD, Reichl P (2015) QoE-defining a user-centric concept for service quality. In: Chen CW, Chatzimisios P, Dagiuklas T, Atzori L (eds) Multimedia quality of experience (QoE): current status and future requirements. John Wiley & Sons Ltd, Hoboken, pp 5–27
3. Seufert M, Egger S, Slanina M, Zinner T, Hoßfeld T, Tran-Gia P (2015) A survey on quality of experience of HTTP adaptive streaming. IEEE Commun Surv Tutor 17(1):469–492
4. Georgopoulos P, Elkhatib Y, Broadbent M, Mu M, Race N (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), Hong Kong, China, pp 15–20
5. Cofano G, Cicco LD, Zinner T, Nguyen-Ngoc A, Tran-Gia P, Mascolo S (2016) Design and experimental evaluation of network-assisted strategies for HTTP adaptive video streaming. In: Proceedings of the 7th ACM multimedia systems conference (MMSys 2016), Klagenfurt, Austria, May 2016
6. Jarschel M, Wamser F, Höhn T, Zinner T, Tran-Gia P (2013) SDN-based application-aware networking on the example of YouTube video streaming. In: Proceedings of the 2nd European workshop on software defined networks (EWSDN 2013), Berlin, Germany, pp 87–92
7. Nam H, Kim KH, Kim JY, Schulzrinne H (2014) Towards QoE-aware video streaming using SDN. In: Proceedings of the 2014 IEEE global communications conference (GLOBECOM), Austin, TX, USA, pp 1317–1322

8. Petrangeli S, Wauters T, Huysegems R, Bostoen T, De Turck F, Network-based dynamic prioritization of HTTP adaptive streams to avoid video freezes. In: Proceedings of the 2015 IFIP/IEEE int'l symposium on integrated network management (IM), Ottawa, Canada, pp 1242–1248
9. Schwarzmann S, Zinner T, Dobrijevic O (2016) Towards a framework for comparing application–network interaction mechanisms. In: Proceedings of the 4th int'l IEEE workshop on quality of experience centric management (QCMAN 2016)
10. Barakovic S, Skorin-Kapov L (2013) Survey and challenges of QoE management issues in wireless networks. J Comput Netw Commun 2013:28
11. Martini MG, Hewage CT, Nasrall MM, Ognenoski O (2015) QoE control, monitoring, and management strategies. In: Chen CW, Chatzimisios P, Dagiuklas T, Atzori L (eds) Multimedia quality of experience (QoE): current status and future requirements. John Wiley & Sons Ltd, Hoboken, pp 149–168
12. Stockhammer T (2011) Dynamic adaptive streaming over http—standards and design principles. In: Proceedings of the second annual ACM conference on multimedia systems. ACM, pp 133–144
13. Adzic V, Kalva H, Furht B (2011) Optimized adaptive HTTP streaming for mobile devices. In: Proceedings of SPIE, pp 81 350T–81 350T–10
14. Fajardo J-O, Liberal F, Mkwawa I-H, Sun L, Koumaras H (2010) QoE-driven dynamic management proposals for 3G VoIP services. Comput Commun 33(14):1707–1724
15. Mok RK, Luo X, Chan EW, Chang RK (2012) QDASH: a QoE-aware DASH system. In: Proceedings of the 3rd multimedia systems conference (MMSys 2012), Chapel Hill, NC, USA, pp 11–22
16. Huang TY, Huang P, Chen KT, Wang PJ (2010) Could Skype be more satisfying? A QoE-centric study of the FEC mechanism in an internet-scale VoIP system. IEEE Netw 24(2):42–48
17. Bouten N, Famaey J, Latré S, Huysegems R, De Vleeschauwer B, Van Leekwijck W, De Turck F (2012) QoE optimization through in-network quality adaptation for HTTP adaptive streaming. In: Proceedings of the 8th int'l conference on network and service management (CNSM 2012), Las Vegas, NV, USA, pp 336–342
18. Zhu J, Vannithamby R, Rodbro C, Chen M, Vang Andersen S (2012) Improving QoE for Skype video call in mobile broadband network. In: Proceedings of the 2012 IEEE global communications conference (GLOBECOM), Anaheim, CA, USA, pp 1938–1943
19. Houdaille R, Gouache S (2012) Shaping HTTP adaptive streams for a better user experience. In: Proceedings of the 3rd multimedia systems conference (MMSys 2012), Chapel Hill, NC, USA, pp 1–9
20. Wamser F, Zinner T, Tran-Gia P, Zhu J (2014) Dynamic bandwidth allocation for multiple network connections: improving user QoE and network usage of YouTube in mobile broadband. In: Proceedings of the 2014 ACM SIGCOMM capacity sharing workshop (CSWS), Chicago, IL, USA, pp 57–62
21. Dobrijevic O, Santl M, Matijasevic M (2015) Ant colony optimization for QoE-centric flow routing in software-defined networks. In: Proceedings of the 11th int'l conference on network and service management (CNSM 2015), Barcelona, Spain, pp 274–278
22. Joseph V, de Veciana G (2014) NOVA: QoE-driven optimization of DASH-based video delivery in networks. In: Proceedings of the 2014 IEEE int'l conference on computer communications (INFOCOM), Toronto, Canada, pp 82–90
23. Thomas E, van Deventer M, Stockhammer T, Begen AC, Famaey J (2017) Enhancing MPEG DASH performance via server and network assistance. SMPTE Motion Imaging J 126(1):22–27
24. Ferguson AD, Guha A, Liang C, Fonseca R, Krishnamurthi S (2013) Participatory networking: an API for application control of SDNs. ACM SIGCOMM Comput Commun Rev 43(4):327–338
25. Paul S, Jain R (2012) OpenADN: mobile apps on global clouds using OpenFlow and software defined networking. In: Proceedings of the 2012 IEEE global communications conference (GLOBECOM) workshops, Anaheim, CA, USA, pp 719–723
26. Paul S, Jain R, Pan J, Iyer J, Oran D (2013) OpenADN: a case for open application delivery networking. In: Proceedings of the 22nd int'l conference on computer communication and networks (ICCCN 2013), Nassau, Bahamas, pp 1–7
27. Wu T (2003) Network neutrality, broadband discrimination. J Telecommun High Technol Law 2:141–180
28. Yiakoumis Y, Katti S, McKeown N (2016) Neutral net neutrality. In: Proceedings of the 2016 ACM SIGCOMM conference. ACM, pp 483–496
29. Wang Z, Lu L, Bovik AC (2004) Video quality assessment based on structural distortion measurement. Signal Process Image Commun 19(2):121–132
30. Seufert M, Slanina M, Egger S, Kottkamp M (2013) "to pool or not to pool": a comparison of temporal pooling methods for HTTP adaptive video streaming. In: Quality of multimedia experience (QoMEX), 2013 fifth international workshop on. IEEE 2013, pp 52–57
31. Seufert M, Hoßfeld T, Sieber C (2015) Impact of intermediate layer on quality of experience of HTTP adaptive streaming. In: Network and service management (CNSM), 2015 11th international conference on. IEEE, 2015, pp 256–260
32. Hossfeld T, Skorin-Kapov L, Heegaard PE, Varela M (2017) Definition of QoE fairness in shared systems. IEEE Commun Lett 21(1):184–187
33. Hoßfeld T, Egger S, Schatz R, Fiedler M, Masuch K, Lorentzen C (2012) Initial delay vs. interruptions: between the devil and the deep blue sea. In: Proceedings of the 4th int'l workshop on quality of multimedia experience (QoMEX 2012), Yarra Valley, Australia, pp 1–6
34. De Cicco L, Caldaralo V, Palmisano V, Mascolo S (2014) TAPAS: a tool for rApid prototyping of adaptive streaming algorithms. In: Proceedings of the 2014 workshop on design, quality and deployment of adaptive video streaming (VideoNext), Sydney, Australia, pp 1–6
35. Li Z, Zhu X, Gahm J, Pan R, Hu H, Begen A, Oran D (2014) Probe and adapt: rate adaptation for HTTP video streaming at scale. IEEE J Select Areas Commun 32(4):719–733