Computer Networks 81 (2015) 96–115

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

QoE-driven in-network optimization for Adaptive Video Streaming based on packet sampling measurements

Niels Bouten [a,*], Ricardo de O. Schmidt [b], Jeroen Famaey [c], Steven Latré [c], Aiko Pras [b], Filip De Turck [a]

[a] Department of Information Technology, Ghent University–iMinds, Gaston Crommenlaan 8/201, B-9050 Ghent, Belgium
[b] Department of Computer Science and Electrical Engineering, University of Twente, Drienerlolaan 5, NL-7522 Enschede, The Netherlands
[c] Department of Mathematics and Computer Science, University of Antwerp, Middelheimlaan 1, B-2020 Antwerp, Belgium

ARTICLE INFO

Article history:
Received 20 September 2014
Received in revised form 16 January 2015
Accepted 10 February 2015
Available online 16 February 2015

Keywords:
Adaptive Video Streaming
Quality of Experience
Optimization
Sampling-based measurements

ABSTRACT

HTTP Adaptive Streaming (HAS) is becoming the de-facto standard for adaptive streaming solutions. In HAS, a video is temporally split into segments which are encoded at different quality rates. The client can then autonomously decide, based on the current buffer filling and network conditions, which quality representation it will download. Each of these players strives to optimize their individual quality, which leads to bandwidth competition, causing quality oscillations and buffer starvations. This article proposes a solution to alleviate these problems by deploying in-network quality optimization agents, which monitor the available throughput using sampling-based measurement techniques and optimize the quality of each client, based on a HAS Quality of Experience (QoE) metric. This in-network optimization is achieved by solving a linear optimization problem both using centralized as well as distributed algorithms. The proposed hybrid QoE-driven approach allows the client to take into account the in-network decisions during the rate adaptation process, while still keeping the ability to react to sudden bandwidth fluctuations in the local network. The proposed approach allows improving existing autonomous quality selection heuristics by at least 30%, while outperforming an in-network approach using purely bitrate-driven optimization by up to 19%.

1. Introduction

The increased popularity of video consumption over the Internet has led to the development of a range of protocols that allow Adaptive Video Streaming over HTTP. Some of the major industrial players have introduced their proprietary protocols such as Microsoft's Silverlight Smooth Streaming,[1] Apple's HTTP Live Streaming[2] and Adobe's HTTP Dynamic Streaming.[3] More recently, a standardized solution has been proposed by MPEG, called Dynamic Adaptive Streaming over HTTP (DASH) [1]. Although differences exist between these implementations, they are all based on the same basic principles: a video is split into temporal segments that are encoded at different quality rates, the client rate adaptation algorithm then dynamically adapts the quality, based on metrics such as average throughput and current buffer filling. The popularity of these adaptive HTTP-based streaming techniques was mainly induced by the advantages offered by HTTP streaming: the

* Corresponding author.
E-mail address: niels.bouten@intec.ugent.be (N. Bouten).

[1] Microsoft Smooth Streaming – http://www.iis.net/downloads/microsoft/smooth-streaming.

[2] Apple HTTP Live Streaming – http://tools.ietf.org/html/draft-pantos-http-live-streaming-12.

[3] Adobe HTTP Dynamic Streaming – http://www.adobe.com/products/hds-dynamic-streaming.html.

http://dx.doi.org/10.1016/j.comnet.2015.02.007
1389-1286/

reuse of caching infrastructure, the reliable transmission over HTTP and the compatibility with firewalls.

State-of-the-art HTTP Adaptive Streaming (HAS) solutions embed the rate adaptation algorithm inside the client application. This allows the client to independently choose its playback quality and prevents the need for intelligent components inside the network, which are the main reasons HAS is used in Over-The-Top (OTT) scenarios. However, academia and industry are showing a growing interest in the use of HAS in managed networks [2],[4,5] for example by optimizing the delivery by applying in-network bitrate adaptation[6] or by deploying IP multicasting to ease the distribution of linear TV HAS services [3].[7] In a recent survey, Seufert et al. argue that a centralized control unit or client-proxy based communication can enhance the quality and establish a fair Quality of Experience (QoE) distribution among competing clients [4]. The extensive content catalogue and increased flexibility in terms of supported devices of these OTT-services (e.g., YouTube, Hulu, Netflix) but delivered over the managed network, could greatly benefit both the provider and the end-user. However, in such environments, a purely client-driven approach has several significant disadvantages. First, the lack of coordination among clients leads to frequent quality oscillations and suboptimal decisions [5–8]. Second, management policies, such as user subscription constraints (e.g., gold, silver and bronze) and guarantees on the delivered quality, cannot be enforced [9,10]. In order to facilitate adoption of HAS for the delivery of multimedia services in a managed environment, these challenges should be tackled.

The first drawback of a purely client-driven approach is of course that control is distributed over the various clients and each client strives to optimize its individual quality. Several clients for which the video flows traverse the same path in the network therefore compete for the available bandwidth. This competition leads to instability in the quality decisions, causing frequent oscillations among different quality representations, bandwidth underutilization and unfairness between players. Most of the client heuristics try to maintain a buffer filling between the configured thresholds and use the download time of segments to estimate the available throughput. If a client has a sufficient buffer filling, the download process is paused until one of the thresholds is reached. During this pause, other clients can benefit from the released resources and increase their download rate. This causes their adaptation heuristic to overestimate the available throughput and wrongfully increase the quality for the next download. When the buffer filling of paused clients reaches the threshold again, their download process is resumed at the same quality level as before. This overestimation of available throughput could lead to congestion in the network, causing segments to arrive late, which in turn can lead to quality oscillations and eventually cause buffer depletion.

A second drawback is the inability for providers to exploit the merits of HAS in a managed environment, since the quality adaptation component is entirely controlled by the end-user. This prevents them from offering any type of QoE guarantees to their subscribers, possibly leading to low QoE and unfair quality distribution among clients. There is also no opportunity to offer service differentiation among the clients, delivering a higher QoE for a higher subscription fee. Moreover, since each client independently takes its quality decisions, there is no opportunity for coordinated management, optimizing the service provider's goals globally. Our proposed approach allows service providers to impose specific management policies. In this way, guaranteeing QoE, service differentiation and global optimization can successfully be deployed.

To alleviate the aforementioned problems caused by wrong estimation of the available throughput, we propose a hybrid quality decision process. Several in-network decision agents are deployed along the delivery path, which monitor the available throughput and based on this, inform the clients on the optimal quality selection. This optimal quality selection is achieved by solving a linear optimization taking as input the monitoring information and the connected subscribers. Clients then take this quality selection as an input to their quality decision heuristic, together with the current buffer filling and throughput estimations. In this way, the clients can still react to sudden throughput changes in their local delivery path, while they can react more dynamically and accurately to throughput changes in the network. To monitor the available throughput, the framework uses sampling techniques, allowing to make a distinction between HAS and cross traffic flows in a scalable way. The in-network quality selection can then take this available HAS throughput as input to its optimization and calculate the optimal quality assignation. This in-network optimization can take several objectives into account, based on the management policy of the provider. We propose a QoE-driven optimization, where the quality is maximized, while minimizing the impact of quality oscillations and buffer starvations. As will be shown in this paper, this approach outperforms the state-of-the-art HAS rate adaptation in terms of QoE.

The benefits of the proposed approach are threefold. First, since the in-network optimization has a global view on the connected video clients, it is able to optimally divide the available resources. This reduces the number of quality oscillations, increases the network utilization and reduces unfairness between players, benefitting the overall QoE of the HAS video delivery service. Second, the sampled monitoring framework allows to accurately forecast future in-network available bandwidth at low capturing costs. Using these estimations as input to the optimization process allows the in-network approach to cope with dynamic networks. Third, since providers are now able to guide clients to certain quality decisions, they can create new business opportunities when adding subscription based management policies.

The proposed approach requires the in-network monitoring and QoE-optimization components to be

[4] Juniper Broadband TV Solution – http://www.juniper.net/us/en/local/pdf/solutionbriefs/3510463-en.pdf.

[5] Velocix Live Streaming – http://www.rgbnetworks.com/pdfs/RGB-VelocixAdaptiveStreamingCDNWhitePaper0911-01.pdf.

[6] Velocix Enhanced Video Experience – http://www.cachelogic.com/vx-portfolio/solutions/velocixeve.

[7] Velocix Optimized Architectures – http://www.velocix.com/vx-portfolio/solutions/video-optimised-architecture.

deployed in the Internet Service Provider (ISP) network. There exist several scenarios in which this could be achieved, depending on which stakeholder deploys the components and offers the HAS streaming service. In a first scenario, the ISP wants to improve its HAS-based streaming service that is offered as part of its triple-play service. In this case, the ISP deploys the in-network components and offers the HAS streaming service as well, allowing subscribers to stream the content offered on the set-top-box, to their own devices (e.g., tablets, smartphones) as well. The second scenario pertains to an ISP that resells third party OTT services with better quality as part of a subscription plan. In this case, the ISP deploys both the in-network components and offers the service, or leases these components to the OTT or Content Delivery Network (CDN)[8] [11] service providers. ISPs such as Proximus[9] and Virgin Media[10] are currently offering Netflix as an additional service without affecting the home broadband connection. In a third scenario, an OTT provider[11] or CDN[12] deploys its own hardware components in the ISP network, which allows them to perform in-network optimization.

The remainder of this article is structured as follows. Section 2 discusses the related research on client-based and in-network HAS rate adaptation. Subsequently, the delivery architecture is described in Section 3. In Section 4 the in-network rate adaptation problem is formally defined, as well as the sampling-based monitoring. Section 5 evaluates the proposed delivery architecture and optimization algorithms and compares the approach with a client-based solution. Finally, Section 6 concludes the article.

2. Related work

At the client side, each commercial HAS implementation comes with a proprietary client heuristic as discussed in Section 1. Several heuristics have been proposed in literature as well, each focussing on a specific deployment. Miller et al. propose a receiver-driven adaptation heuristic for DASH that takes into account a history of available throughput and the buffer level [12]. The quality is adjusted to attain a buffer level between certain target thresholds, this improves the stability of the quality and avoids frequent switching as a consequence of short-term throughput variations. Jiang et al. identified the problems that arise when multiple clients share a link [13]. The authors propose a variety of techniques that can help avoid said undesirable behavior, such as harmonic bandwidth estimation, stateful and delayed bitrate update and randomized scheduling of requests, which are grouped in the FESTIVE adaptation algorithm. Tian et al. show that there is a trade-off between responsiveness and smoothness for client-side DASH adaptations [14]. The proposed rate-switching logic provides a dynamic control of this trade-off according to the trend of the buffer growth. The approach uses machine-learning based TCP throughput prediction to support multiple servers simultaneously. In previous work [15,16], the authors evaluated different client heuristics both for Advanced Video Coding (AVC) and Scalable Video Coding (SVC), applying optimizations such as pipelined and parallel download scheduling. The approach presented in this paper is applicable to both AVC and SVC. Liu et al. discuss a video client heuristic that is suited for a CDN by comparing the expected segment fetch time with the experienced segment fetch time to ensure a response to bandwidth fluctuations in the network [17], while Adzic et al. present a client heuristic which is tailored for mobile environments [18].

Among others [5,8], Akhshabi et al. compare several commercial and open source HAS players and indicate significant inefficiencies in each of them, such as frequent oscillations and unfairness when the number of competing clients increases [6,7]. Said quality oscillations are known to have a negative impact on QoE [19] and cause inefficient resource utilization within the bottleneck network [7,20]. In a recent survey, Seufert et al. argue that a centralized control unit or client-proxy based communication can enhance the quality and establish a fair QoE distribution among competing clients [4]. This paper aims at controlling the quality by introducing global QoE management, reducing the number of quality oscillations and allowing to reduce the required buffer size.

By altering the behavior of the streaming server, stability and bandwidth efficiency can be increased. Akhshabi et al. propose server-side rate adaptation to cope with unstable streaming players due to ON–OFF patterns when they compete for bandwidth [21]. The system detects sudden rate fluctuations in the client playout and tries to solve them by shaping the sending rate at the server to resemble the bitrate of the stream. Liu et al. follow a comparable approach where the rate is shaped according to QoE maximization metrics [22]. These systems are able to restore the streaming session when oscillation or freezing occurs and then remove the shaping when the client has stabilized. Our approach is not only able to solve the problems of oscillation or freezes when they occur, but actively tries to prevent them, while at the same time optimizing the QoE. In previous work, we have shown that, although the proposed server-side rate shaping can increase stability, they are not able to achieve the stability offered by in-network quality optimization due to a reactive, rather than proactive behavior [23]. Our approach is also able to handle multiple origin servers and intermediate caching nodes.

Li et al. propose a collaboration scheme between CDNs and ISPs and peer-assisted CDNs to reduce the load on both peering links and internal ISP links [24]. Distributed CDN servers alter the manifests to associate chunks with regional storage servers or by changing or increasing the available video quality levels by transcoding the video. The approach proposed in this paper could be complementary to the distributed CDN case, where intermediary

[8] Akamai Aura Managed CDN – http://www.akamai.com/html/solutions/managed-content-delivery-network.html.

[9] Netflix on Proximus TV – http://www.proximus.be/en/id_cr_netflix/personal/our-products/television/series-movies/proximus-and-netflix.html.

[10] Virgin Media, Netflix on TiVo – http://store.virginmedia.com/digital-tv/channels/netflix.html.

[11] Netflix Open Connect – https://openconnect.itp.netflix.com.

[12] Akamai Accelerated Network Partner – http://www.akamai.com/html/solutions/akamai-accelerated-network-partner.html.

storage servers exist and client sessions consuming their content are terminated in these nodes. Our approach further increases the efficient use of the ISP's network by minimizing the negative impact of competing sessions on QoE. Famaey et al. assess the impact of increased latency on QoE caused by the redirection of HAS requests in CDNs and propose updated request routing schemes in order to reduce the number of redirects [25].

Network level adaptations allow a more efficient use of the underlying network resources for HAS. Essaili et al. propose a TCP rate shaping mechanism on a per flow level to enhance the QoE by rewriting client requests and offering control to the network operator which has better information on the load and radio conditions in the cell [26]. Houdaille et al. propose to use traffic shaping in the residential gateway to implement bandwidth arbitration between competing clients [8]. Others propose to exploit the advantages of Software Defined Networking (SDN) in a HAS environment to monitor the streaming sessions and, in conjunction with a client control plane, dynamically adjust video flow characteristics to ensure QoE [27]. Recently, Mueller et al. proposed to use HTTP/2.0 when deploying adaptive streaming and evaluate the overhead and impact on link utilization when using HTTP/2.0 for HAS [28]. Using new features in HTTP/2.0, such as server push, persistent connections and pipelining, HAS services can be improved. Wei et al. show that the increasing protocol overhead that is caused by decreasing the segment length can be overcome by automatically pushing a number of segments, allowing them to reduce the live latency [29,30]. Using the server push feature proposed in HTTP/2.0 would allow the in-network optimization to force certain quality decisions onto the clients, increasing the overall QoE.

In previous work, we proposed to use Differentiated Services (DiffServ) to guarantee the delivery of certain segments in a live streaming scenario [31]. This requires altering the relevant prioritization fields for these specific segments and implementing DiffServ routers in the ISP network. An autonomic delivery framework for HAS-based Live TV and Time Shifted TV (TSTV) was presented in previous work [32,3] which allows to reduce the consumed bandwidth by grouping unicast HAS sessions sharing the same content into a single multicast session. However, for Video on Demand (VoD) HAS sessions, the content is more diverse and only few sessions are potentially shared among multiple users. This prevents them to be grouped into a shared multicast session and therefore prevents them from being delivered in a scalable manner.

Petrangeli et al. use in-network proxies to determine the overall median chunk level requested by the clients and disseminates this information towards each client [33]. A reinforcement learning-based quality selection allows then to achieve fairness among the clients. Krishnamoorthi et al. use a set of intermediary proxies to evaluate the impact of caching policies on HAS [10]. Furthermore, they argue that client and proxy should cooperate and exchange information on buffer filling and cache contents, in order to optimize the delivery of cached segments. Our proposed approach could be extended to incorporate the intelligent caching mechanism, as it is a mere relocation of the content server. Since we optimize regularly, the caching dynamics and their impact on link utilization could be included during the optimization. Our approach extends the related work by furthermore alleviating instability and inefficiency problems that arise when multiple HAS clients compete. QDASH is a QoE-aware DASH system where an in-network measurement proxy is deployed to provide accurate bandwidth measurements to the client [34]. This measurement proxy only performs measurements on a per-client level, our approach supports multi-client scenarios and estimates upstream available throughput to divide among them in order to maximize QoE.

Other approaches limit the available video quality by altering the data transported over the network. In [35] a graceful degradation of video quality is proposed when the network load increases. The authors argue the need for Media Aware Network Elements (MANEs), capable of adjusting the SVC stream based on a set of policies specified by the network provider. Similar to this approach, Latré et al. proposed an in-network rate adaptation algorithm, responsible for determining which SVC quality layers should be dropped in combination with a Pre-Congestion Notification (PCN) based admission control mechanism [36]. In [37], a prototype of an intermediary adaptation node is proposed, where the media gateway estimates the available bandwidth on the client link and extracts the supported SVC-streams. Situnen et al. propose dropping video layers based on their priority when network congestion arises for scalable video streaming over wireless networks [38]. Most of the aforementioned research focuses on the dropping of quality layers when congestion arises, meaning the quality is limited in the same way for all users. Our proposed approach limits the maximum quality in a per client manner, allowing the service provider to differentiate the delivered video services based on the client's subscription. This allows the service provider to control the QoE on a subscriber level, and thus offering different subscription types for the VoD HAS services.

Lee et al. describe a three-tier streaming service where clients are connected through multiple intermediate proxies to a multimedia server [39]. The authors only consider live streaming, whereas our system also supports VoD streaming. Furthermore, videos need to be transcoded in the intermediary proxies, in standard HAS however, the quality levels are discrete and fixed, causing the objective function in the proposed solution to change drastically and leading to the inability to use the max–min composition. In [40,41], the authors focus on optimizing the allocation of bits of video sequences among multiple senders to stream to a single client. Peer-to-peer streaming and multi-server distributed streaming are the main use-cases of this approach, there is no simple extension of the work when multiple clients need to share the same server side bottleneck. Furthermore, this requires fine-grained scalable video streaming to support the allocation of non-overlapping bit ranges to multiple servers, while for HAS, fixed bitrate representations are available, encoded using advanced video coding, leading to video segments of which the quality cannot be improved in a straightforward way

by downloading additional bit ranges. Our work however, could also be extended to support scalable video in a straightforward way.

This article is an extension to our previous work on in-network quality management for HAS [42,23]. This previous work only considered managed environments, where a fixed capacity is available. We extended this approach to allow dynamic scenarios where non-HAS traffic is measured to estimate the available residual capacity. To this end, several throughput forecasting techniques are presented and evaluated. Additionally, other client-side rate adaptation heuristics were implemented and evaluated. Furthermore, both the model and evaluations were extended to take into account the impact of freezes, switches and playout quality on perceived QoE. These improvements over the previous approach allow a network-wide QoE optimization in a dynamic network environment.

3. QoE-driven delivery architecture for Adaptive Video Streaming

This article proposes a hybrid approach, steering the rate adaptation algorithm at the client by an in-network component. It is deployed on one or multiple intermediary proxies and supports client-side rate adaptation algorithms by dynamically limiting the possible set of bit rates to select from. Fig. 1 shows an overview of the in-network quality decision architecture and the different flows of data and information for a centralized deployment. Using a hybrid approach, where client heuristics take the in-network quality decisions as a maximum bound, allows clients to still react upon sudden network changes or scarcity in device resources, while increasing the overall quality and stability. One or more adaptation agents can be deployed to optimize the HAS delivery, depending on the size of the delivery network. Using distributed optimization algorithms, several cooperating agents can be deployed along the delivery path to achieve a scalable in-network quality adaptation. The in-network rate adaptation agents are modeled to resemble an autonomic control loop and consists out of the following steps:

Packet-based monitoring. Controlling the streaming quality from within the network, requires measuring the current traffic traversing the streaming paths links via monitoring agents to reconstruct the cross traffic on the delivery path. Operators estimate this available bandwidth by reading interface counters over specific time periods (e.g., every 5 to 15 min), which could be accomplished by using Simple Network Management Protocol (SNMP).[13] However, these measurements are of coarse granularity and might lead to very inaccurate estimations of bandwidth usage. More accurate approaches are available, but they often demand continuous packet capturing to calculate traffic statistics [43]. However, due to traffic rates observed nowadays, continuous packet capturing does not scale and requires expensive software/hardware. To avoid these issues, packet sampling is deployed within the monitoring process. Packet sampling provides a trade-off between accuracy and ease-of-use of bandwidth estimation approaches. That is, although providing only a summary of actual traffic, as compared to continuous packet capturing, sampled packet capturing provides much more granular data than interface counters at a limited capturing overhead. Schmidt et al. propose a monitoring technology using packet filtering and sampling to provide scalable packet-based monitoring in high-speed networks and use it for link dimensioning [44]. The monitoring architecture consists of agents embedded in switches and routers and a centralized controller. Several monitoring agents can be grouped to monitor the traffic along a fraction of the media delivery path. This allows both a centralized and distributed approach towards packet-based monitoring and in-network quality decision.

Residual Bandwidth Estimation. To achieve dynamic quality adaptation, several packet-based monitoring agents are deployed along the HAS delivery paths. The monitoring information of these monitoring agents is then processed to estimate the bandwidth taken by the different flows. Using these estimations on each link's cross traffic, the residual bandwidth available for HAS along the path can be predicted. This provides the in-network quality control with a prediction of the future residual bandwidth available for HAS traffic. The packet-based residual capacity estimation will be discussed in Section 4.2. The HAS sessions that are monitored along the delivery path and the predicted residual bandwidth then serve as input towards the in-network quality decision algorithm.

QoE-driven quality optimization. This step takes the predicted residual bandwidth along the delivery paths as input and optimizes the selected quality for each individual HAS flow. In Fig. 1, a centralized approach is shown, where the monitoring information of all the monitoring agents is forwarded to a centralized controller. This controller then has a full view on the delivery network and is able to optimize the QoE as will be discussed in Section 4.3. To provide a more scalable solution, multiple intermediate proxies could be deployed within the delivery network. They have a local view of the network and only require input of a subset of monitoring agents, limiting the communication overhead of the monitoring data. These intermediate proxies then cooperate to achieve a global policy as will be discussed in Section 4.4. Based on the management policy of the operator, the objective of this optimization can be altered.

Quality selection guidance. The result of the aforementioned step is a list of quality limitations for each individual client. The clients are then guided towards selecting the optimal quality in order to optimize the objective. This could be done in several ways. A first method is to throttle the throughput of the client's flows [21] at the server which allows to impact the actions of the client decision heuristic in a transparent way. Since the client will measure a lower or higher throughput, depending on the configured rate at the server, the decision will be steered towards the optimal quality. A disadvantage of this approach is that the quality guidance process is indirect and it could possibly take a while for the client to converge to the optimal quality. A second approach is to rewrite the manifest files in a per client

[13] Cisco Systems Inc. (2005) – http://www.cisco.com/image/gif/paws/8141/calculate_bandwidth_snmp.pdf.

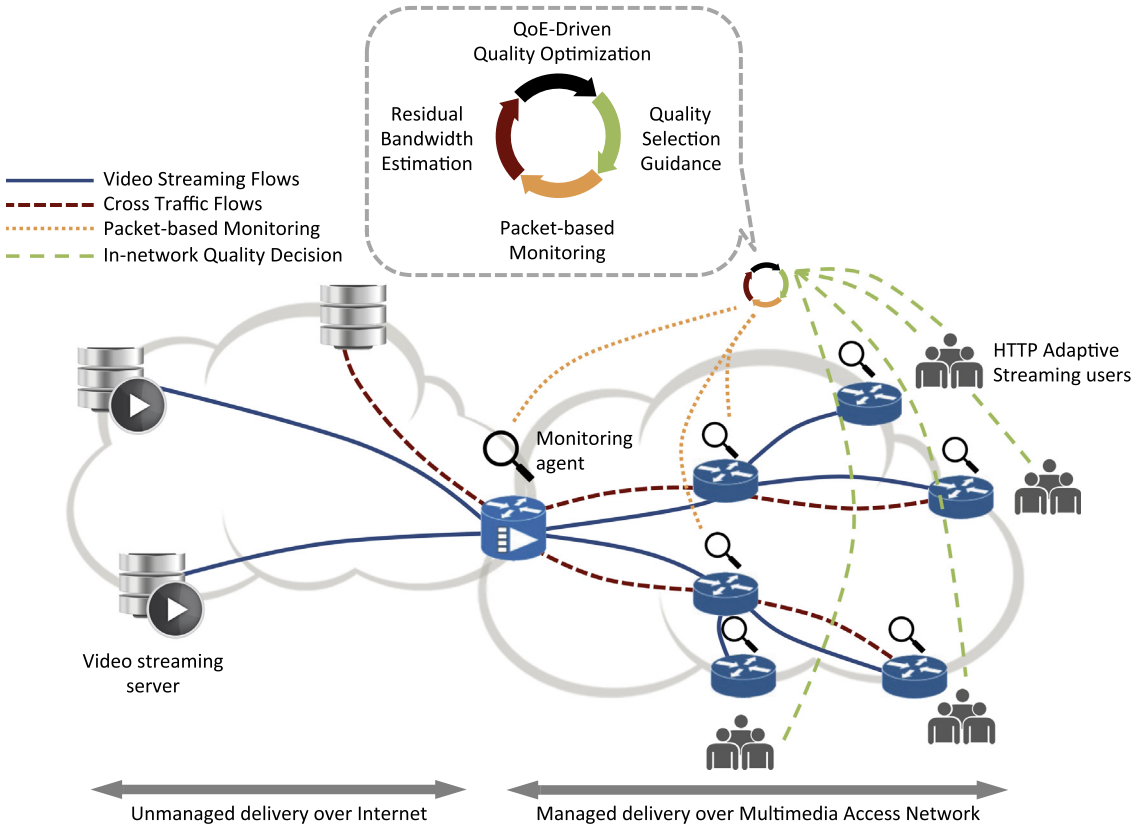


Fig. 1. In-network quality management architecture, showing a centralized component gathering monitoring information, estimating residual bandwidth which serves as input for the QoE-driven quality optimization and quality selection guidance for the clients. Further on in the paper, also a distributed version is considered.

way, by leaving out quality representations that are not feasible under the current circumstances. This allows the client to still autonomously select the best quality, based on its measurements, while enforcing a certain maximum quality. Since the manifest is regularly updated, this approach requires the client to download the manifest frequently. A third way, is to extend HAS protocol to allow specification of optimal quality guidelines, which are then processed by the clients. Quality selection information could be embedded in the HAS header of the segments, causing limited overhead of only a few bytes without requiring the introduction of additional protocol messages. In this paper, we chose to add additional information to the segment headers which is then processed by the proposed client, since the guidance is direct and causes limited overhead.

4. Algorithm description

4.1. Definition of variables and assumptions

We consider a network topology modeled as a graph, consisting of a set of nodes $\mathcal{N}$, which encompasses servers $\mathcal{S} \subset \mathcal{N}$, proxies $\mathcal{P} \subset \mathcal{N}$, and clients $\mathcal{C} \subset \mathcal{N}$. A set of edges $\mathcal{E}$ connects the nodes in a logical tree topology which is typically used for video delivery networks, although the underlying physical network might not be a tree due to replication concerns. Each node $n \in \mathcal{N}$ has an incoming edge $e_{n^-} \in \mathcal{E}$ connecting the node to its predecessor $n^- \in \mathcal{N}$ as well as a set of outgoing edges $\mathcal{E}_{\mathcal{N}^+} \subset \mathcal{E}$ connecting to its successors $\mathcal{N}_n^+ \subset \mathcal{N}$ in the logical tree topology. Every edge $e \in \mathcal{E}$ has an associated capacity B_e as well as an estimation for the cross traffic $T_{e,t}$ for the next timeslot t. This results in an estimated residual capacity $R_{e,t}$ which is available to assign to HAS traffic. Each video $v \in \mathcal{V}$ has an associated set of quality representations $\mathcal{Q}_v$, for which every quality representation $q \in \mathcal{Q}_v$ has a bit rate β_q. Each client $c \in \mathcal{C}$ has a unique delivery path $\mathcal{E}_c \subseteq \mathcal{E}$ from server to client. The set of clients that have an edge $e \in \mathcal{E}$ as part of their delivery path $\mathcal{E}_c$, is represented by $\mathcal{C}_e \subseteq \mathcal{C}$. Correspondingly, the set of clients for which the delivery path crosses a node $n \in \mathcal{N}$ is represented by $\mathcal{C}_n \subseteq \mathcal{C}$.

4.2. Packet-based residual capacity estimation

To estimate the future traffic on an edge e, we use sampled traffic measurements. We want to distinguish the HAS traffic from unrelated traffic generated by other services, to estimate the impact of this cross traffic on the edge e. This can be obtained by intercepting packets and inspecting the TCP and HTTP packet headers. However, due to increasing traffic rates, full packet capturing is often not advised due to scalability issues. Packet sampling is an attractive alternative to continuous packet capturing, while still providing

highly granular traffic measurements (as compared to, e.g., interface counters). If the sampled traffic amount $L_{e,t}$ was obtained at time t at a rate $1/r$ (e.g., $r = 100$ for 1:100 sampling) every τ seconds, the original amount of traffic $A_{e,t}$ can be estimated by Eq. (1). This value gives an estimate of the actual amount of cross traffic during the interval τ.

$$A_{e,t} = r \times L_{e,t} \tag{1}$$

Tailoring the decisions of the HAS clients during the next interval, requires a prediction of the future load. Several forecasting techniques can be used to predict the future load on e:

- *Exponential Weighted Moving Average.* Exponentially Weighted Moving Average (EWMA) attempts to forecast the future load $T_{e,t}$ on edge e based on the packet-based traffic estimation $A_{e,t}$ using a smoothing factor α as shown in Eq. (2) [45].

$$T_{e,t} = \alpha \times A_{e,t} + (1-\alpha) \times T_{e,t-1} \tag{2}$$

- *Holt Winters.* The non-seasonal Holt Winters (HW) predictor is a variation of EWMA, attempting to capture trends in time series [46]. A separate smoothing component $T^s_{e,t}$ and a trend component $T^t_{e,t}$ are defined in Eqs. (4) and (5) respectively and depend on the parameters α and β.

$$T_{e,t} = T^s_{e,t} + T^t_{e,t} \tag{3}$$
$$T^s_{e,t} = \alpha \times A_{e,t} + (1-\alpha) \times T_{e,t-1} \tag{4}$$
$$T^t_{e,t} = \beta \times (T^s_{e,t} - T^s_{e,t-1}) + (1-\beta) \times T^t_{e,t-1} \tag{5}$$

- *Exponential trend method.* A variation of Holt's linear trend method is generated by using multiplicative adjustments to the level and slope, rather than additive. This leads to an Exponential Trend (ET) $T^t_{e,t}$, rather than a linear trend. These adjustments are shown in Eqs. (6)–(8).

$$T_{e,t} = T^s_{e,t} \times T^t_{e,t} \tag{6}$$
$$T^s_{e,t} = \alpha \times A_{e,t} + (1-\alpha) \times T_{e,t-1} \tag{7}$$
$$T^t_{e,t} = \beta \times (T^s_{e,t}/T^s_{e,t-1}) + (1-\beta) \times T^t_{e,t-1} \tag{8}$$

- *Autoregressive model.* In Auto Regression (AR), a history of past values of the measured cross traffic are used to forecast the future load. Using an AR model of order h, the prediction can be performed as shown in Eq. (9), where c is a constant and the random variable ϵ_t is white noise. To determine the order h, the Akaike information criterion can be used [47], while least median of squares estimation can be used to find the parameters $\phi_i : i = 1 \ldots h$ [48].

$$T_{e,t} = c + \sum_{i=1}^{h} (\phi_i \times A_{e,t-i}) + \epsilon_t \tag{9}$$

- *Support Vector Regression.* Time series can also be predicted using a lagged vector of previous measurements. The basic principle of Support Vector Regression (SVR) is to estimate the output variable $T_{e,t}$, from $\phi(\mathcal{T}_{e,t})$. $\mathcal{T}_{e,t} = (A_{e,t-h}, \ldots, A_{e,t-1})^\intercal$ is an input vector containing h previous measurements [49]. Using a non-linear mapping $\phi(\cdot)$, the vector $\mathcal{T}_{e,t}$ is projected to a higher dimensional space. The regression model is then shown in Eq. (10), where ω is the weight vector and b the bias term. Using Sequential Minimal Optimization (SMO), the regression problem can be solved [50].

$$T_{e,t} = \omega^\intercal \times \phi(\mathcal{T}_{e,t}) + b \tag{10}$$

- *Multi Layer Perceptron* Also a two-layer Multi Layer Perceptron (MLP) can be used to predict the future load. The network consists of a single linear output activation (n_0) and m hidden sigmoid activations ($n_1, \ldots, n_m$) and takes as inputs a history of h measurements, normalized with respect to the available capacity in the interval $\tau : \tau \times B_e$. Each node takes as input the output of the preceding nodes in the network, a weight vector ω^n and a bias value b_n. The linear output and hidden sigmoid activations are shown in Eqs. (11) and (12) respectively, where $\mathcal{O}_i$ are intermediary outputs of the hidden layer nodes. The logistic sigmoid function $\varphi(x) = \frac{1}{1+\exp(-x)}$ is used as activation function for the hidden layer.

$$T_{e,t} = \tau \times B_e \times \left(b_0 + \sum_{i=1}^{m} \omega^0_i * \mathcal{O}_i \right) \tag{11}$$
$$\mathcal{O}_i = \varphi\left(b_i + \sum_{j=0}^{h} \omega^i_j * \frac{A_{e,t-j}}{\tau \times B_e} \right) \tag{12}$$

The residual capacity at edge e can then be estimated as defined in Eq. (13).

$$R_e = B_e - \frac{T_{e,t}}{\tau}$$

4.3. QoE-driven quality optimization

The optimization is modeled as an Integer Linear Programming (ILP) problem and consists of maximizing the QoE over all clients $c \in \mathcal{C}$, while adhering to the edge bandwidth constraints. The solution is characterised by a boolean decision matrix A. The element $a_{c,q} \in A$ is equal to 1 if quality $q \in \mathcal{Q}_v$ is selected for client $c \in \mathcal{C}$, and 0 otherwise. The constraints in Eqs. (15) and (16), state that the decision variables are boolean values and that only one quality representation can be selected per client. The total consumed bandwidth of HAS-traffic on every edge $e \in \mathcal{E}$ caused by all clients $c \in \mathcal{C}_e$ should not exceed the estimated residual bandwidth for HAS traffic $R_{e,t}$ as defined in Eq. (17). According to Padhye et al., the maximum achievable throughput B for a TCP connection subject to a round trip time RTT and maximum window size W_{max}, is limited by $\frac{W_{max}}{RTT}$ [51]. This adds an additional constraint on the per flow achievable throughput a shown in Eq. (14).

$$\forall c \in \mathcal{C} : \sum_{q \in \mathcal{Q}_c} a_{c,q} \times \beta_q \leqslant \frac{W_{max,c}}{RTT_c} \tag{14}$$

$$\forall c \in \mathcal{C},\ \forall q \in \mathcal{Q}_v :\ a_{c,q} \in [0,1] \tag{15}$$
$$\forall c \in \mathcal{C} : \sum_{q \in \mathcal{Q}_v} a_{c,q} = 1 \tag{16}$$
$$\forall e \in \mathcal{E} : \sum_{c \in \mathcal{C}_e} \sum_{q \in \mathcal{Q}_v} a_{c,q} \times \beta_q \leqslant R_e \tag{17}$$

The provider can choose to optimize video delivery in several ways using a different objective function. One possibility is to maximize the total video bitrate over all clients using the following optimization function:

$$\max \sum_{c \in \mathcal{C}} \sum_{q \in \mathcal{Q}_v} a_{c,q} \times \beta_q \tag{18}$$

A major drawback of the aforementioned approach is that it neglects the impact of quality switches on QoE [19]. Therefore, we adapted a QoE-metric for HAS to include both quality and switching information during the in-network optimization. The QoE-driven objective aims to optimize the global QoE over all clients. For HAS services, the QoE as shown in Eq. (19) is a weighted combination of the average delivered quality (μ), the standard deviation of quality (σ) [52,53] and a correction factor to incorporate the impact of frame freezes (ϕ) [54]. The formulas for calculating the values μ, σ and ϕ are defined in Eqs. (20)–(24), with K the number of played segments, N the number of quality levels for video, Q_k the quality played for segment $k \in [1, K]$ and F the set of frame freezes.

$$eMOS = \alpha \times \mu - \beta \times \sigma - \gamma \times \phi + \delta \tag{19}$$

$$\mu = \frac{\sum_{k=1}^{K} \frac{Q_k}{N}}{K} \tag{20}$$

$$\sigma = \sqrt{\frac{\sum_{k=1}^{K} \left(\frac{Q_k}{N} - \mu\right)^2}{K}} \tag{21}$$

$$F_{freq} = \frac{|F|}{K} \tag{22}$$

$$F_{avg} = \frac{\sum_{f \in F} duration(f)}{|F|} \tag{23}$$

$$\phi = \frac{7}{8} \max\left(\frac{\ln(F_{freq})}{6} + 1, 0\right) + \frac{1}{8} \min\left(\frac{F_{avg}}{15}, 1\right) \tag{24}$$

The approximation for MOS in Eq. (19) was used to model the objective function. The term ϕ, quantifying the impact of freezes was omitted, since the constraints of the proposed approach guarantee the delivery of the lowest quality, thus avoiding freezes. The average quality over time μ includes the decision variables for each client. The formula to calculate the switching term σ is dependent on μ and is quadratic in this term. Including the estimated MOS as is, would yield a non-linear objective function. In order to avoid this, the calculation of μ and σ values is split into several calculations $\mu_{c,q}$ and $\sigma_{c,q}$ per client c and per quality q. Since the decision variables $a_{c,q}$ are modeled as binary values, multiplying this decision variable with the corresponding $\mu_{c,q}$ and $\sigma_{c,q}$ values allows a linear objective function. To this end, a history $\mathcal{H}_c$ of previous quality decisions is required for every client $c \in \mathcal{C}$. This history allows calculating the average quality $\mu_{c,q}$ for every client $c \in \mathcal{C}$ and every possible quality decision $q \in \mathcal{Q}_v$ for the next timeslot in Eq. (25). The variation in quality $\sigma_{c,q}$ can be calculated for each client using the set of averages $\mu_{c,q}$ for each quality decision $q \in \mathcal{Q}_v$ as shown in Eq. (26).

$$\forall c \in \mathcal{C},\ \forall q \in \mathcal{Q}_v:\ \mu_{c,q} = \frac{1}{|\mathcal{H}_c| + 1} \left(\frac{q}{|\mathcal{Q}_v|} + \sum_{h_{c,t} \in \mathcal{H}_c} \frac{h_{c,t}}{|\mathcal{Q}_v|} \right) \tag{25}$$

$$\forall c \in \mathcal{C},\ \forall q \in \mathcal{Q}_v:\ \sigma_{c,q} = \sqrt{\frac{1}{|\mathcal{H}_c| + 1} \left(\left(\frac{q}{|\mathcal{Q}_v|} - \mu_{c,q} \right)^2 + \sum_{h_{c,t} \in \mathcal{H}_c} \left(\frac{h_{c,t}}{|\mathcal{Q}_v|} - \mu_{c,q} \right)^2 \right)} \tag{26}$$

Calculating the specific averages $\mu_{c,q}$ and deviations $\sigma_{c,q}$ for each possible decision allows to formulate the objective function without introducing quadratic terms for the decision variables. Using the decision variable $a_{c,q}$, the estimated MOS can then be maximized as follows:

$$\max \sum_{c \in \mathcal{C}} \sum_{q \in \mathcal{Q}_v} a_{c,q} \times \left(\alpha \times \mu_{c,q} - \beta \times \sigma_{c,q} + \delta \right) \tag{27}$$

4.4. Distributed QoE-driven quality optimization

The number of constraints for the QoE-driven quality optimization grows significantly with the number of proxies and clients in the service delivery tree. Solving the optimization problem using only one agent will increase the execution times. Prolonged optimization times endanger the ability to react adequately to sudden throughput changes along the edges of the delivery network. The problem can be distributed over the different nodes along the delivery network to obtain a scalable solution. The proposed approach uses a bottom-up technique, that was described in previous work [23].

Fig. 2 shows a graphical overview of how the distributed optimization process is performed. Each node $n \in \mathcal{N}$ monitors the upstream edge e_{n^-} and estimates the residual bandwidth $R_{e_{n^-}}$ for HAS traffic along that edge. Using this information the local optimization objective for node n can be formulated as shown in Eq. (28). This optimization is constrained by the available HAS capacity of the upstream edge as detailed in Eq. (29).

$$\max \sum_{c \in \mathcal{C}_n} \sum_{q \in \mathcal{Q}_v} a_{c,q} \times \left(\alpha \times \mu_{c,q} - \beta \times \sigma_{c,q} + \delta \right) \tag{28}$$

$$\sum_{c \in \mathcal{C}_n} \sum_{q \in \mathcal{Q}_v} a_{c,q} \times \beta_q \leqslant R_{e_{n^-}} \tag{29}$$

$$\forall c \in \mathcal{C}_n,\ \sum_{q \in \mathcal{Q}_v : q > s_{n^+,c}} a_{c,q} = 0 \tag{30}$$

The local optimization is then constrained by (15) and (16) as before but only for the subset $\mathcal{C}_n$ of clients for which the traffic traverses node n. In order not to violate any bitrate limitations further downstream the topology, the local optimization is constrained by the solutions obtained by its set of successor nodes $\mathcal{N}_n^+$. For each client $c \in \mathcal{C}_n, s_{n^+,c}$ determines the maximum quality a client is allowed to download according to the local optimizations downstream. Eq. (30) puts an additional constraint on the optimization determining that the selected quality for client c is not allowed to violate the downstream limitations. The aforementioned distributed approach has several advantages. Each node only requires local information on the upstream edge, while the number of edge constraints

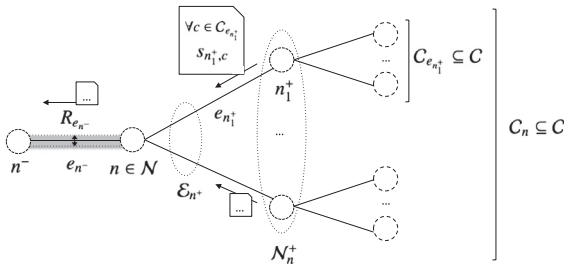


Fig. 2. Distributed optimization process for a node n gathering monitoring information from its upstream link e_{n^-} to estimate $R_{e_{n^-}}$ and downstream restrictions $s_{n^+,c}$ from its successor node set $\mathcal{N}^+$.

per local optimization process is equal to one. Furthermore, since the optimization for different subtrees is independent of subtrees at the same level, the processes can be executed in parallel.

The complexity of the optimization can be reduced at the expense of optimality by moving from an ILP formulation towards a Relaxed Linear Programming (LP) formulation. This can be achieved by removing the boolean constraints on the decision variables $a_{c,q}$ as defined in Eq. (15) and replacing them by the following floating point decision variables as shown in Eq. (31). We refer to our previous work on how this floating point solution can be transformed into an integer solution [23].

$$\forall c \in \mathcal{C}_n,\ \forall q \in \mathcal{Q}_v:\ 0 \leqslant a_{c,q} \leqslant 1 \tag{31}$$

5. Evaluation results

5.1. Experiment setup

A VoD HAS scenario was implemented using the discrete-event network simulator NS3,[14] simulating the transmission of HAS-based video [42]. The framework has been extended with support for packet-based measurements in network routers and switches. The HAS servers and proxies have been adapted to incorporate these measurements during the QoE-driven network optimization. For the autonomic HAS Clients, several heuristics found in literature were implemented. A first implementation uses the *Microsoft Smooth Streaming (MSS)* algorithm, which is based on the implementation of an open source version of the algorithm of the *MSS* video player[15] and is extensively described by Famaey et al. [15]. A second implementation is based on the heuristic proposed by Miller et al. [12], which is a receiver-driven adaptation algorithm based on buffer filling level and throughput estimations. This heuristic is referred to as *Miller*. A third heuristic, called *Festive*, is based on the implementation described by Jiang et al. using randomized scheduling and stateful bitrate selection [13]. The parameters of the aforementioned heuristics were optimized to attain the best QoE for a variety of scenarios.

An additional client heuristic was implemented, which downloads each segment using the QoE management quality decision is proposed in this article. We refer to this as *AVC Steered*. This heuristic takes the signaled quality decision as an input and decides wether the assigned quality level is feasible under the current network circumstances. This allows the *AVC Steered* client to address local network congestion, which the QoE-driven in-network optimization is unable to take into account.

The optimization algorithms were implemented using the IBM CPLEX[16] solver. We use two versions of the optimization algorithm: an *Exact* calculation, modeled as an ILP and the relaxation of the problem denoted as *Relaxed*. We implemented both the *QoE-driven optimization* and the simpler *bitrate optimization*, which only optimizes the average quality, disregarding quality oscillations [42]. Next to a *Centralized*, a *Distributed* heuristic approach was implemented to address scalability issues. This *Distributed* approach requires upstream information exchange between the different cascading proxies. This exchange is modeled as network communication to take into account network delays when exchanging solutions and installing the optimal configuration at the clients. Also the delay introduced by forwarding monitoring information and processing this data to predict the future bandwidth are taken into account during the simulations.

Fig. 3 shows a typical tree-structured video service delivery network overlay. The first level has K branches, while the second level has M branches, both with a default value of 4. To each of these branches, a number of connected clients is assigned randomly within the interval $[0, N]$, representing the number of active clients out of the number of connected homes N. The links are dimensioned proportionally to the maximum number of clients N per branch. The average Round Trip Time (RTT) for each client c is set to RTT = 40 ms [55]. Clients are started using a Weibull startup process with shape 2.5 and mean of 300 s. The *Big Buck Bunny* video[17] was encoded at 7 different quality rates and divided into 200 segments with an average duration of 2 s. Table 1 gives an overview of the different quality layers and their associated bitrates.

To introduce cross-traffic in the network with a realistic degree of variability, the cross traffic was modeled based on a set of bandwidth traces described by Riiser et al. [56]. The traces[18] provide a highly variable throughput, which makes the prediction of the future bandwidth challenging. They have a total duration of about 220 min. The available bandwidth fluctuates between 202 bps and 6335 kbps with an average of 2192 kbps and a standard deviation of 1317 kbps. The bandwidth traces were cut in 2000 second parts which were used to generate traffic on different paths during the evaluations. Fig. 4 shows an example of such a bandwidth trace excerpt of 2000 seconds. As discussed in Section 4.2, several forecasting techniques can be used to predict the future load on an edge. The

[14] ns-3 - https://www.nsnam.org.
[15] Source available from https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming.
[16] IBM ILOG CPLEX Optimizer: http://www.ibm.com/software/integration/optimization/cplex-optimizer/.
[17] Big Buck Bunny available from http://www.bigbuckbunny.org/.
[18] Dataset available from http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/bus.ljansbakken-oslo/.

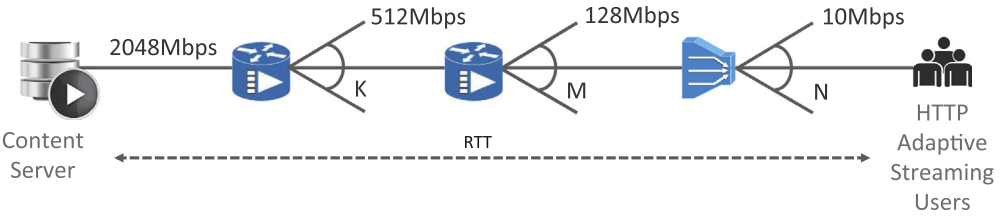


Fig. 3. Network topology, modeling a typical video service delivery network.

Table 1
Overview of the quality layers for the Big Buck Bunny video.

Quality layer index	Average bitrate (kbps)	Average PSNR (dB)
0	300	32.04
1	427	32.72
2	608	34.41
3	866	35.71
4	1233	36.88
5	1636	37.64
6	2436	40.07

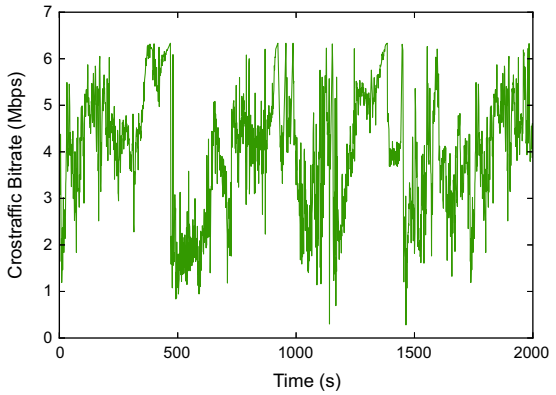


Fig. 4. Example cross traffic trace.

parameters of each of the presented prediction techniques were estimated using a training set of the bandwidth data. For the implementation of AR, SVR and MLP prediction, WEKA 3[19] was used to perform training and prediction.

The estimated average Mean Opinion Score (MOS) defined by Eq. (19) was used to evaluate the performance of both the purely client-based approaches and the in-network assisted approach [52–54]. To tune the parameters (α, β, γ and δ), a set of 15 adaptive streaming scenarios was generated to assess the end-user perception of bitrate switching and buffer starvations using the *Big Buck Bunny* video. Fig. 5 shows a graphical example of three such test scenarios, where we tried to capture the impact of frequent switching (a), gradual switching (b) and buffer starvations (c). During a subjective screening test with 10 experts in the field of video streaming, the MOS values for each of the test sequences ($n \in [1, N]$) were measured [57]. By minimizing the Root Mean Squared Error (RMSE) between the predicted values $MOS_{pred,n}$ and the actual measured values $MOS_{subj,n}$, the parameters were tuned:

$$\min \sum_{n\in[1,N]} \frac{\left(MOS_{pred,n} - MOS_{subj,n}\right)^2}{N} \tag{32}$$

The following values were obtained: $\alpha = 5.67$, $\beta = 6.72$, $\gamma = 4.95$ and $\delta = 0.17$. The MOS estimation assumes that there is a linear relationship between the relative quality level of each video representation and the MOS score for that quality level.

The individual values of each of these QoE-terms are used during the discussion to indicate how the behavior of the evaluated approaches differs. Since frame freezes have a negative impact on QoE, the total buffer starvation time of each client was measured and is indicated as the total buffer starvation time in seconds. Also the total number of quality oscillations each client experiences, as well as the average played quality bitrate in Mbps was examined. To indicate the buffering behavior of the different approaches, the average buffer filling is expressed as a percentage of the maximum allowed buffer. The in-network optimization uses the prediction methods discussed in Section 4.2 to forecast the available throughput for HAS during the next timeslot. To indicate the correlation between the actual values and the estimated values, the Pearson Correlation Coefficient is used. All of the following results are averaged over $n = 20$ iterations, using different cross traffic traces, with the graphs showing the 95% confidence intervals.

To assess the impact of the different parameters on the in-network QoE optimization, we first carry out a parameter analysis where we perform simulations for different values of the *history size* $|\mathcal{H}_c|$, the *optimization interval* τ, the *sampling rate r* and the *buffer size B*. The optimal values (as discussed in Section 5.2.5) for the respective parameters obtained during the analysis are used further on during the evaluations. Next, the different forecasting techniques are evaluated in terms of precision and their impact on QoE. The interference between the in-network optimization and client-side adaptation are discussed afterward. Furthermore, the overheads of the proposed approach introduced by exchanging partial solutions and measurement data are quantified and the scalability of the proposed approach are evaluated for increasing *network size M* and *number of homes N*.

5.2. Parameter analysis

5.2.1. Impact of the decision history

In order to optimize the QoE in terms of average quality and quality switches, the in-network quality optimization maintains a list of previous quality decisions on a per client

[19] WEKA: Data Mining Software in Java – http://www.cs.waikato.ac.nz/ml/weka/.

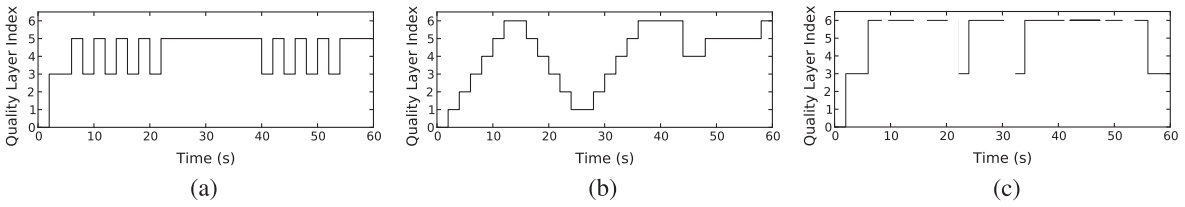
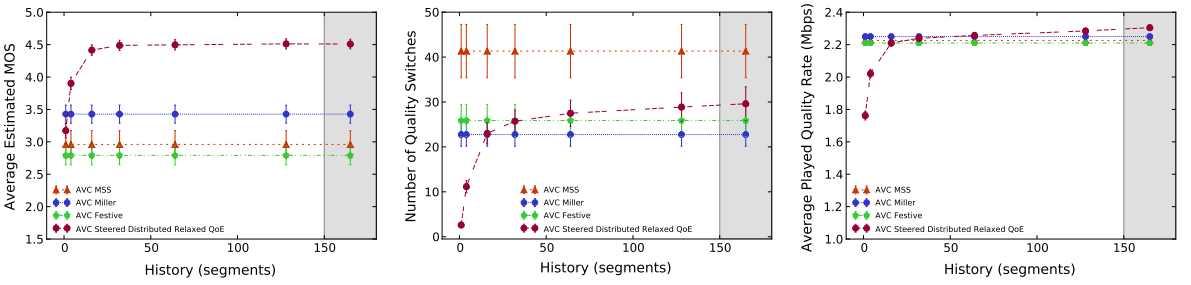


Fig. 5. Graphical overview of three example sequences that were used to assess the end-user subjective quality perception. Gaps in (c) indicate the occurrence of buffer starvations.

basis. The size of this history $|\mathcal{H}_c|$ has an impact on the actual quality perceived by the client in terms of MOS. This is illustrated in Fig. 6(a) where the history size $|\mathcal{H}_c|$ is varied within $[1, 2, 4, 8, 16, 32, 64, 128, \infty]$, with $|\mathcal{H}_c| = \infty$ corresponding to the situation where we keep a history of all decisions for each client $c \in \mathcal{C}$. Increasing the size of the maintained history has a positive effect on the average estimated MOS. A history of only 1 quality decision will cause the optimization to severely penalize quality switches, leading to a low number of quality switches as indicated in Fig. 6(b) but at the same time preventing the decision algorithm to increase the quality, negatively impacting the average quality as indicated in Fig. 6(c). With a history of only 1 decision, the impact of the current decision will be much higher than with a larger history, leading to a high variance which heavily increases the switching penalty and thus prevents the optimization to switch to another quality. There is a local optimum for a history size $|\mathcal{H}_c|$ of 128 previous decisions.

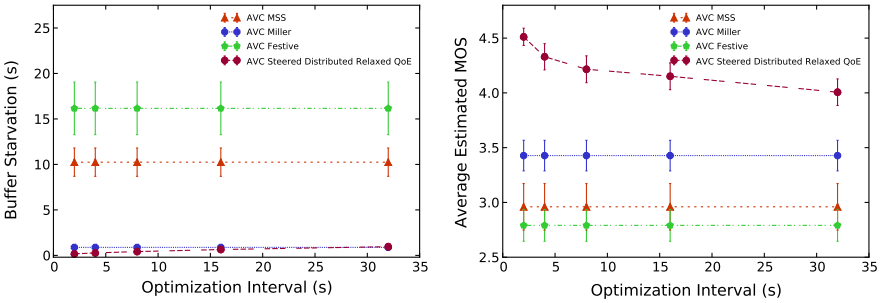
5.2.2. Impact of the optimization interval

The in-network optimization calculates the optimal quality for each client when clients join or leave the network. Additionally, to adapt to the fluctuating cross traffic, the optimization is performed periodically. Fig. 7(a) shows the increased risk of running into buffer starvations when the optimization interval τ is increased. This can be attributed to the larger timespan between two consecutive bandwidth estimations, leading to less accurate cross traffic estimations and quality decisions when traffic fluctuates heavily during that period of time. An interval of 2 s allows the in-network QoE-optimization to completely avoid frame freezes, positively impacting the QoE, while purely client-driven approaches are suffering from frequent frame freezing due to wrong estimations of the available throughput. Fig. 7(b) shows the impact on the average MOS of these frame freezes. This shows the trade-off between optimization frequency and overal QoE. Under highly fluctuating cross traffic, optimizing the quality



(a) Impact on average estimated MOS (b) Impact on average number of switches (c) Impact on average quality (Mbps)

Fig. 6. Impact of history size $|\mathcal{H}_c|$ ($RTT = 40$ ms, $r = 100$, $\tau = 2$ s, $B = 12$ s, $N = 32$). These results show a local optimum of $|\mathcal{H}|_c = 128$.



(a) Impact on buffer starvations (s) (b) Impact on average estimated MOS

Fig. 7. Impact of optimization interval τ ($RTT = 40$ ms, $r = 100$, $|\mathcal{H}_c| = 128$, $B = 12$ s, $N = 32$). Setting the optimization interval τ at the segment length of 2 s yields the highest QoE at the cost of frequent optimization.

decisions every 2 s yields a 12% improvement compared to optimizing every 32 s. The 2 s interval corresponds to the segment length of the video, indicating that optimizing the quality for each segment produces optimal results.

5.2.3. Impact of the sampling rate

When sampling network traffic at a rate $1/r$, there is a tradeoff between the scalability of the monitoring and the precision of the estimated bandwidth as illustrated in Fig. 8 and Table 2. Randomly sampling the traffic with a sampling size $r = 100$ as shown in Fig. 8 yields precise estimations using MLP with very few outliers and a high corellation of $\rho = 0.965$. Increasing the sampling size r has a negative impact on estimation precision and as a result, a negative impact on the in-network optimization process as illustrated in Fig. 9(a). Making incorrect predictions on the estimated throughput of the cross traffic can yield wrong decisions during the optimization, leading to an increased number of frame freezes and more frequent switching as illustrated in Fig. 9(b) and (c) respectively. A sampling size of $r = 100$ allows accurate predictions of the throughput, while limiting the overhead to sampling only 1% of the packets.

5.2.4. Impact of the buffer size

Fig. 10(a) shows the impact of the buffer size on average estimated MOS for *MSS*, Miller, *Festive*, bitrate-based and QoE-based optimization. Depending on the buffer size, the in-network QoE-driven optimization is able to achieve an improvement between 30% and 43% in terms of estimated MOS compared to the best client-side adaptation heuristic (Miller). The bitrate-based optimization allows increasing the estimated MOS with 10% to 25%. These results show a significant increase of 13% to 19% in average QoE when also quality oscillations are included during the optimization process, instead of only optimizing the quality in terms of average bitrate per client. The maximum improvement over traditional autonomic adaptation is achieved for a buffer of 8 s. Starting from 12 s, increasing the buffer size for the in-network optimization has only a limited effect on the average MOS, while the client-side algorithms continue to improve. This can be attributed to the increased quality stability and larger safety margin to cope with bandwidth changes when the buffer is increased. This higher quality stability can be achieved by the QoE-driven optimization for smaller buffer sizes as there is additional knowledge on estimated throughput and optimal quality decisions from within the network. As shown in Fig. 10(b), the in-network optimization almost completely fills the buffer for any size, while client-side heuristics try to improve the quality when the buffer filling is sufficiently high and thus trade in a higher buffer filling for a quality increase. Especially when considering live streaming, small buffers are important since they allow reducing the latency with respect to the live event. Furthermore, maintaining only a limited buffer reduces the memory costs of the video application. For a buffer of about 4 s, the in-network optimization is able to achieve a similar QoE as the best performing client-side heuristic using a buffer of four times that size. Using in-network optimization allows shrinking the client-side buffer without compromising the QoE.

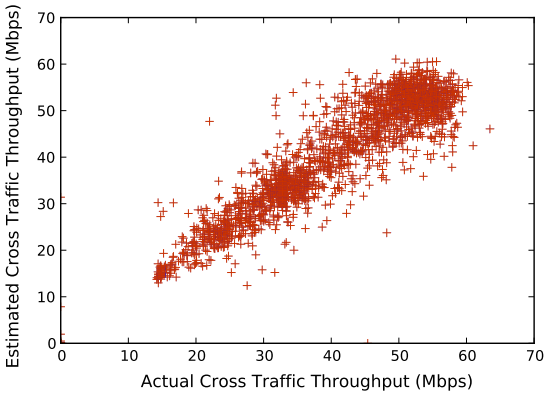


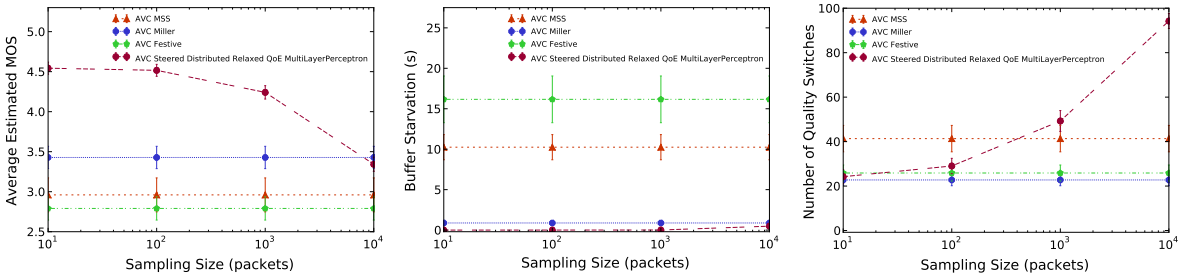
Fig. 8. Impact of sampling size r on packet-based throughtput estimation for $r = 100$ showing a high correlation.

Table 2
Pearson correlation between the estimated throughput for the next interval τ and the actual cross traffic rate. A sampling size of $r = 100$ yields a high Pearson correlation of $\rho = 0.965$.

Sampling size (r)	Pearson correlation (ρ)
10	0.970
100	0.965
1000	0.939
10,000	0.827

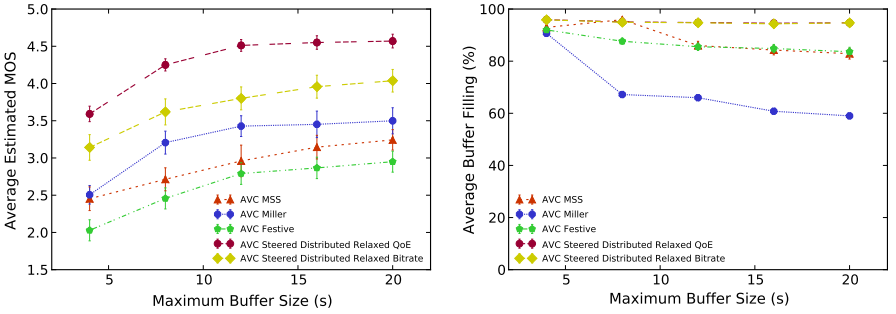
5.2.5. Selected parameter values

Based on the above parameter analysis, the best configuration is determined to be $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $r = 100$ and $B = 12$ s. These parameter configurations allow increasing the overall QoE with about 30% compared to pure client-side state-of-the-art adaptation heuristics. In the previous sections, we analyzed that keeping a history of previous decisions allows the in-network optimization to reduce the number of quality oscillations, which benefits the overall QoE. Choosing a history that is too small, will prevent quality switching, but may impact the quality rate. Setting the history too large, increases storage overhead. Therefore we suggest to keep a history $|\mathcal{H}_c|$ of 128 previous decisions. In a highly dynamic network environment, it is important to frequently reassess the optimal quality allocation. Not only clients joining or leaving the network can cause suboptimal or infeasible allocations, also the fluctuating available network throughput can impact the decisions. Therefore, the optimization interval τ should be sufficiently small in order to respond to these changes in a timely fashion. Since clients need to make a decision for every segment they download, we suggest to select τ equal to the segment length of 2 s. Continuously capturing packets is not scalable and increases the complexity of capturing equipment and thus the overall network cost. Therefore, packet sampling is deployed within the monitoring process. Sampling packets at a rate $r = 100$, allows a high correlation $\rho = 0.965$, while reducing the capturing to 1% of total data packets. A buffer storing segments deployed at the client side, allows resolving



(a) Impact on average estimated MOS (b) Impact on buffer starvations (s) (c) Impact on number of quality switches

Fig. 9. Impact of sampling size r ($RTT = 40$ ms, $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $B = 12$ s, $N = 32$). Increasing the sampling size r beyond 10^3 negatively impacts the QoE due to wrong estimations on the future throughput. Setting $r = 100$ gives good estimations at a limited sampling cost of 1%.



(a) Impact on average estimated MOS (b) Impact on average buffer filling (%)

Fig. 10. Impact of buffer size B ($RTT = 40$ ms, $r = 100$, $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $N = 32$). For a buffer of about 4 s, the in-network optimization is able to achieve a similar QoE as the best performing client-side heuristic using a buffer of four times that size.

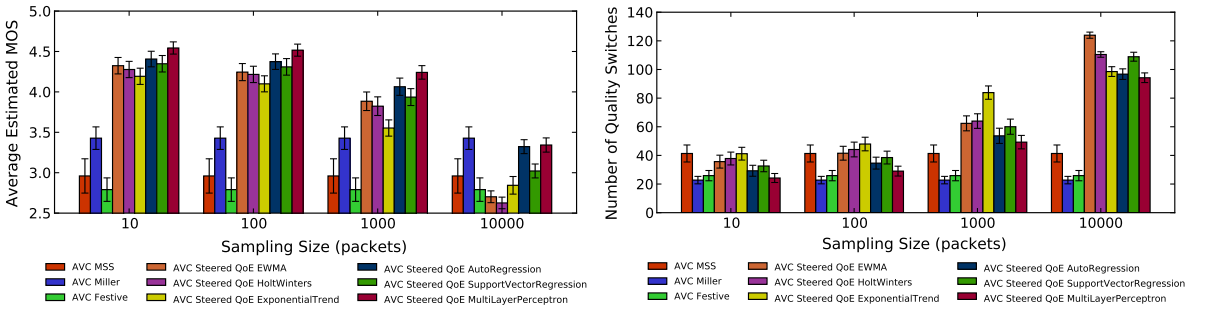
temporal throughput fluctuations. A large buffer increases storage requirements at the client device and increases latency with respect to the live signal. In case of channel switching the prefetched content of this buffer is cleared, which means that the buffered content was not useful. Therefore, the size of the buffer should be chosen large enough so that temporal fluctuations can be resolved, but at the same time sufficiently small to limit the storage, useless prefetching and live latency. We propose a buffer size of 12 s, since it requires limited storage and greatly improves QoE over client-side heuristics. The optimal parameter values deduced during the analysis are used to evaluate the proposed system further on.

5.3. Impact of the forecasting method

In Section 4.2, several forecasting techniques were proposed to estimate the future available capacity for the HAS traffic on each edge e. Table 3 gives an overview of the average Pearson correlation for the different forecasting techniques when using a sampling rate $r = 100$. More complex techniques, such as MLP, SVR and AR yield slightly higher correlation than more straightforward estimation techniques, such as EWMA, HW and ET. The latter only require setting the weighting parameters α and/or β, while the more complex techniques require a training step, yielding more accurate forecasts. Improved estimation of the future available bandwidth for HAS also translates in more accurate QoE optimization. Fig. 11(a) shows the impact of the sampling size r for the various forecasting techniques on QoE. These results show, that although EWMA, HW and ET have lower correlation values, they still allow an improvement over state-of-the-art adaptation heuristics in terms of QoE. However, when the sampling size r increases, these forecasting techniques are more prone to misestimations, leading to even lower prediction accuracy and eventually lower QoE. Deploying more complex forecasting techniques allows improving overall QoE, even if the sampling size r increases. When comparing MLP to EWMA, better estimations allow an increase of 9% in QoE for a sampling size of $r = 100$ and 28% if the

Table 3
Pearson correlation and standard deviation for different forecasting techniques for a sampling size of $r = 100$

Forecasting technique	Pearson correlation (ρ)	Standard deviation (δ_ρ)
Exponential Weighted MA (EWMA)	0.932	0.036
Holt Winters (HW)	0.926	0.037
Exponential Trend (ET)	0.909	0.042
Auto Regression (AR)	0.949	0.034
Support Vector Regression (SVR)	0.939	0.034
Multi Layer Perceptron (MLP)	0.966	0.035



(a) Impact on average estimated MOS

(b) Impact on number of quality switches

Fig. 11. Impact of forecasting method ($RTT = 40$ ms, $r = 100$, $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $B = 12$ s, $N = 32$) for multiple values of the sampling rate r.

sampling size is further increased up to $r = 10,000$. MLP is even able to outperform EWMA for a higher sampling size r: QoE of 4.24 at $r = 1000$ for MLP compared to a QoE of 4.14 at $r = 100$ for EWMA. Even though there is a slight increase in complexity due to the more sophisticated forecasting, MLP is able to reduce the sampling overhead with a factor 10 compared to EWMA.

5.4. Impact of last mile bandwidth fluctuations

In this section, the impact of last mile throughput fluctuations is assessed and how the in-network optimization and client adaptation heuristic interact in such scenarios. To this end, additional cross-traffic is introduced on the last mile connection to simulate the network behavior of mobile devices. Several sets of traces were generated leading to an average available throughput of 0.8 Mbps, 1.6 Mbps, 2.4 Mbps, 3.2 Mbps, 4 Mbps, 4.8 Mbps and 5.6 Mbps respectively. We also evaluated two additional adaptation schemes. For the first scheme, we use the *AVC Steered* client side adaptation, without the in-network control. This comes down to a purely client-driven adaptation scheme that only takes into account the estimated throughput at the client side to select the next quality. In the second scheme (*AVC Distributed Relaxed QoE*), only the in-network adaptation is performed, without taking into account possible throughput fluctuations due to the mobile network. Here, the client downloads the quality that is suggested by the in-network optimization without checking the feasibility in view of the current network conditions. Furthermore, the combination of both schemes is evaluated (*AVC Steered Distributed Relaxed QoE*) together with Miller, which yielded the best results in the previous sections. This allows us to compare a purely client-driven adaptation, a purely network-driven adaptation and the combination of both techniques.

Fig. 12(a) shows the impact on QoE for the different scenarios. If the average throughput at the last mile is low, the purely network-driven approach is not able to detect this and overestimates the available throughput, negatively impacting the QoE. As the last mile throughput increases, the bottleneck shifts to the shared part of the network and the network-driven adaptation is able to increase the QoE thanks to additional monitoring information and global view of the subscribers. The purely client-driven approach shows a similar course as Miller, but about 20% lower due to the basic adaptation heuristic. Fig. 12(c) shows the impact of the different scenarios on buffer starvations. Since the purely network-driven adaptation constantly overestimates the available throughput, most of the segments arrive late, leading to almost 350 s of frame freezing, plunging the QoE as shown before. In Fig. 12(d) the results for the purely network-driven approach are left out for presentation reasons. These results show that the simple client-driven adaptation is not able to cope with the ON–OFF behavior that occurs when multiple HAS clients are connected. Also the adaptation heuristic proposed by Miller et al. shows an increasing number of frame freezes, when the last mile throughput increases. This indicates that it is able to cope with throughput fluctuations of the mobile network, but suffers from the congestion that is induced by other connected clients in the network. The combination of in-network and client-driven adaptation is able to overcome both the throughput fluctuations caused by the local network and those caused by competition between connected clients. When comparing the number of switches in Fig. 12(b), a similar behavior is shown, where the basic client-driven approach is not able to reach a stable quality level and the purely network-based adaptation makes wrong estimations on the available throughput, causing frame freezes, which are interpreted as quality switches. These results show that a combination of in-network and client-side adaptation is required to cope with both the competition of HAS clients and the local network fluctuations, which are not monitored by the in-network adaptation.

5.5. Overhead of in-network optimization

When deploying in-network QoE-optimization for HAS, several overheads are introduced, ranging from monitoring data transfers to partial solution exchanges. The overheads that are incurred differ between the centralized and distributed approach.

To allow a distributed optimization, local subsolutions need to be propagated to their parent nodes. If $|\mathcal{C}|$ is the total number of subscriptions, this list of local solutions, or any combination made by any node, contains at most $|\mathcal{C}|$ entries. To be able to uniquely identify each client, the list should contain an ID composed out of at least

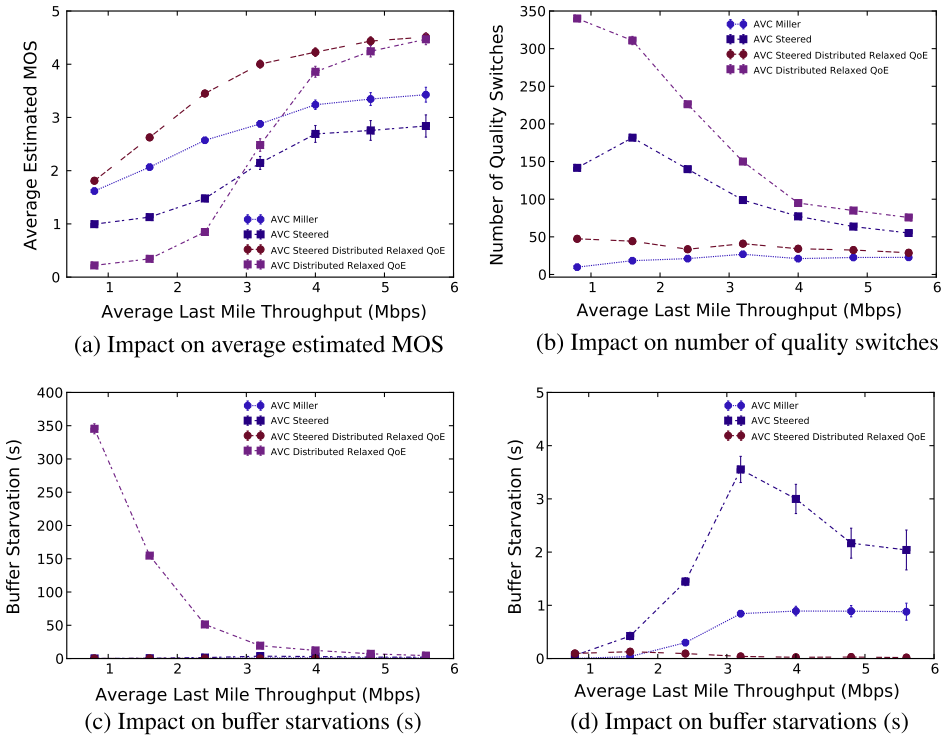
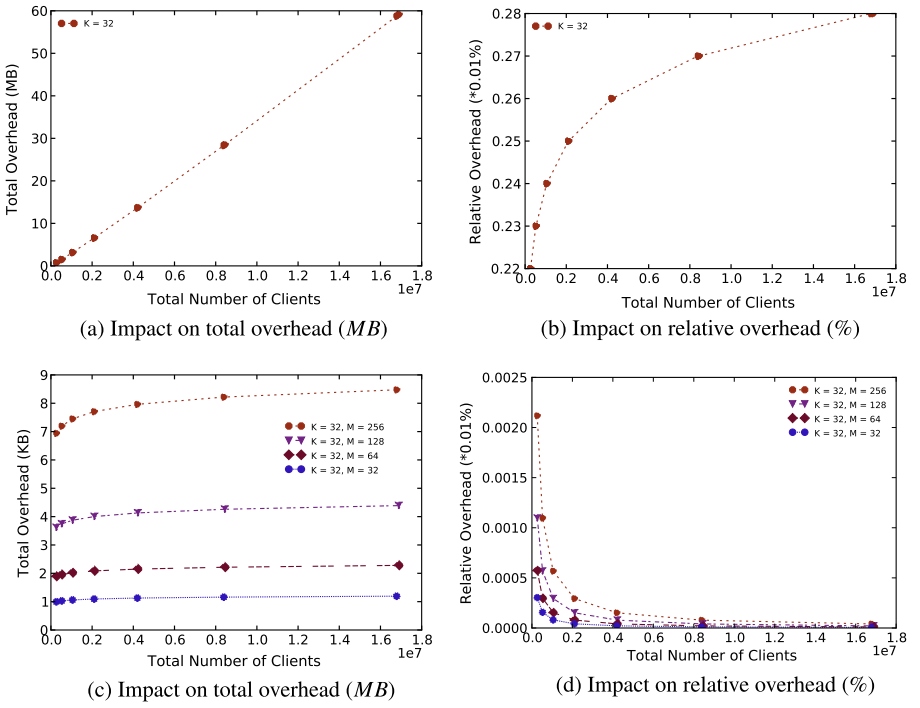


Fig. 12. Impact of access network bandwidth fluctuations ($RTT = 40$ ms, $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $B = 12$ s, $N = 32$).

$\lceil \log_2 |\mathcal{C}| \rceil$ bits. If the number of available quality rates is $|\mathcal{Q}|$, there are $\lceil \log_2 |\mathcal{Q}| \rceil$ bits required to encode the local decision for each client. Fig. 13(a) shows how the total network size impacts the communication overhead incurred by distributed optimization. This graph shows a linear relation ($\mathcal{O}(|\mathcal{C}|)$) between the total number of clients (for various configurations of K, M and N) and the total incurred overhead in *MB* during each execution of the in-network optimization. Fig. 13(b) shows the overhead relatively to the total network usage when assuming an average load of 1 Mbps induced by each connected client. With an impact of around 0.0028% on total network usage, the communication overhead introduced by the distributed optimization can be considered negligible.

The centralized approach does not require the exchange of subsolutions. However, since a global view of the network is required, the throughput measurements performed at several locations in the network need to be forwarded to the node performing the centralized optimization. For each edge e, the estimated amount of traffic (in bits) $A_{e,t}$ is forwarded to this node every τ seconds. Since only one value has to be transferred per link every τ seconds, the total overhead ($\sum_{e \in \mathcal{E}} \lceil \log_2 A_{e,t} \rceil$) involved with exchanging these measurements is negligible compared to the total streaming traffic. Fig. 13(c) shows the total overhead in function of the number of connected clients for different values of M. This shows the ($\mathcal{O}(\log(|\mathcal{C}|))$) relationship between the measurement data and the total number of connected clients per interval τ for a specific value of M. Increasing K or M in the topology shown in Fig. 3, increases the number of links that need to be monitored, while increasing N, only increases the number of connected clients, leaving the number of monitored links unchanged. This shows that the structure of the topology has an impact on the communication overhead for the centralized optimization. Fig. 13(d) shows an inverse logarithmic behavior for the relative impact of measurement communication overhead when the number of clients increases. Increasing N leads to more clients and thus higher traffic rates, decreasing the relative impact of the communication overhead. Increasing K or M, increases the number of links that need to be monitored, so for the same number of clients $|\mathcal{C}|$, the relative impact is higher when K or M are increased. Increasing the sampling interval τ can further reduce the impact of the measurement communication overhead, but at the cost of lower prediction precision and hence QoE as was shown in Section 5.2.2.

To optimize the QoE, a history of prior decisions $\mathcal{H}_c$ needs to be maintained for each client $c \in \mathcal{C}_n$. In Section 5.2.1, we established that $|\mathcal{H}_c| = 128$ yields the highest QoE. However, maintaining a history of previous decisions for each client also comes at the cost of increased memory requirements. The amount of information that needs to be maintained by each node is in the order of $|\mathcal{C}_n| * |\mathcal{H}_c| * \lceil \log_2 |\mathcal{Q}| \rceil$, where $\lceil \log_2 |\mathcal{Q}| \rceil$ is the number of bits required to uniquely represent $|\mathcal{Q}|$ quality levels. Concretely, for a client set $\mathcal{C}_n$ of $1,000,000$, a history size $|\mathcal{H}_c|$ of 128 and videos with a quality set $|\mathcal{Q}_v|$ of maximum 7 representations, the required storage is 48 MB. Using compression techniques, these storage requirements could



(a) Impact on total overhead (MB) (b) Impact on relative overhead (%)

(c) Impact on total overhead (MB) (d) Impact on relative overhead (%)

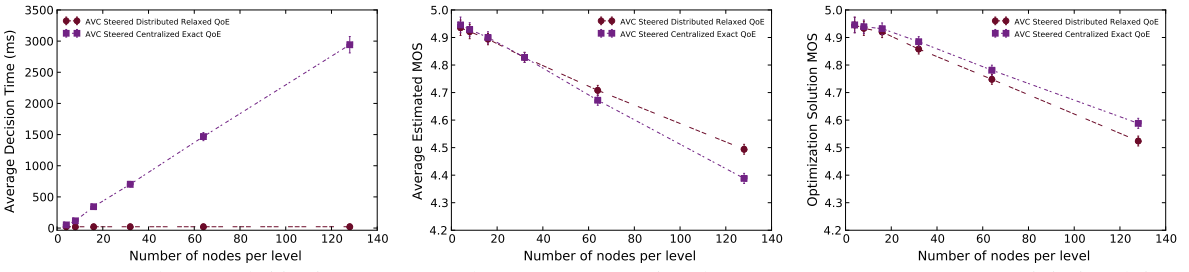
Fig. 13. Impact of total number of clients $|\mathcal{C}|$ for various combinations of network size parameters ($K = 32, M \in [32, 64, 128, 256]$ and $N \in [512, 1024, 2048, 4096]$) on the communication overhead for distributed (a and b) and centralized (c and d) in-network QoE optimization respectively.

be further reduced thanks to repetitive character of the quality values, caused by the avoidance of quality oscillations.

The delay incurred by the network affects the exchange of these partial solutions for the distributed optimization. Since each local optimization requires input of the previous ones, the communication delay incurred by these partial solutions, also affects the execution time of the global optimization as was discussed in previous work [23]. Also the centralized optimization is affected by network delay, since the global optimization requires input of all link states in the network.

5.6. Scalability of the QoE-driven quality optimization

Fig. 14(a) shows the impact of increasing the number of nodes per level M and the associated link dimensions on the average execution time of the in-network optimization. The network delay impacts both the *Centralized* and *Distributed* approach for the distribution of cross traffic information and quality decisions, respectively. The *Centralized* optimization times linearly increase with the number of nodes per level, since the number of connected clients and edge constraints increase. The *Distributed* optimization experiences only limited impact thanks to the parallel optimization, leading to execution times of about 21 ms, mostly caused by the network delay. Increasing the number of nodes per level does not increase the execution times at that level since the optimizations can be executed in parallel. For small problem sizes, the *Centralized* optimization outperforms the *Distributed* optimization in terms of optimality, thanks to the complete knowledge of the problem. As can be seen from Fig. 14(b), this optimality is lost when the number of nodes



(a) Impact on the average decision time (ms) (b) Impact on average estimated MOS (c) Impact on average optimization solution

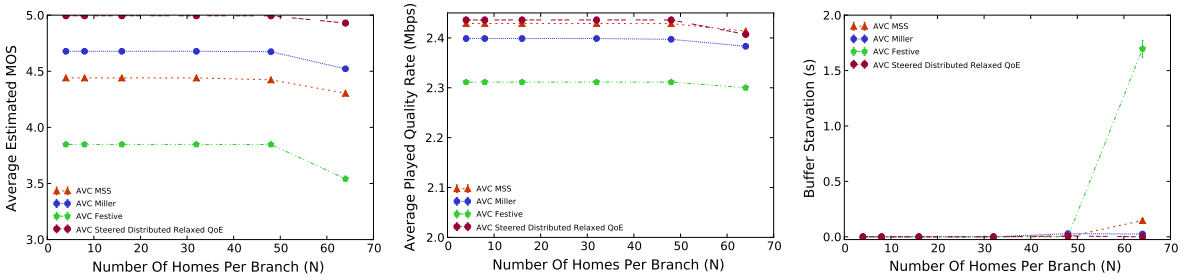
Fig. 14. Impact of increasing the number of nodes per level M ($RTT = 40$ ms, $r = 100$, $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $B = 12$ s, $N = 16$). Since the calculation delay for the *Centralized* optimization is quite high, the practical results differ significantly from the optimal solution, showing the benefits of the more scalable *Distributed* optimization.

per level M exceeds 32. Due to the longer solution times (up to 2.9 s), suboptimal solutions are installed, leading to occasional buffer starvations and more frequent switching, negatively impacting the overall QoE. In a real-life setting, these execution times heavily impact the optimality of the calculated solution due to delayed availability of the optimal solution. Fig. 14(c) shows the resulting QoE for the solutions obtained by the in-network optimization. This is the theoretical solution that could be achieved by the optimization in absence of calculation delays and buffering at the client. The optimal solution for the *Distributed* relaxation is almost identical to the results obtained in Fig. 14(b), since there is little impact of the calculation delay. For the *Centralized* optimization however, the results are quite different. Since the calculation delay is quite high, the practical results differ significantly from the optimal solution. The *Centralized* ILP optimization outperforms the *Distributed* relaxation by 1.4% in terms of theoretical optimality, while in a practical setting, the *Centralized* optimization is outperformed by 2.4%.

To measure the impact of increasing the number of homes N per branch, two scenarios were evaluated. In the first scenario, no additional traffic was sent over the links in the network. This ensures that the competition for throughput among clients is the sole source of quality adaptations. We refer to the first scenario as *uncongested*. For the second scenario, additional cross traffic was introduced that competes with the video traffic, causing additional quality adaptations. This additional traffic causes the network to be congested and is referred to as *congested* scenario.

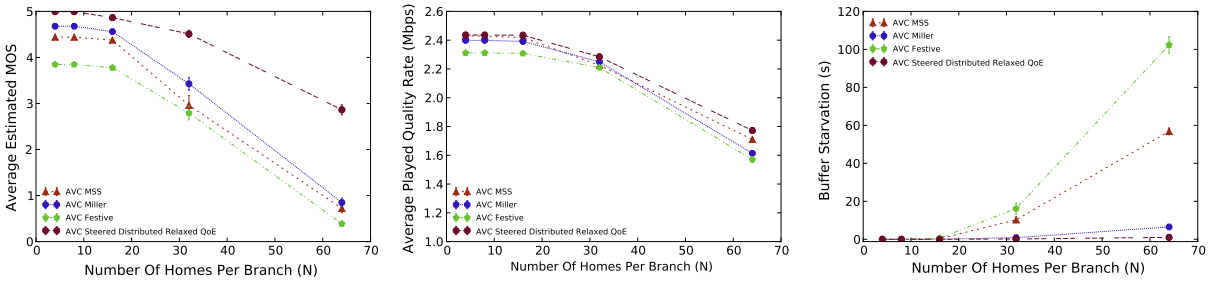
In absence of cross traffic, the QoE-driven optimization achieves an average QoE that is about 7% higher than for Miller as illustrated in Fig. 15(a). This can be attributed to the faster startup quality of in-network based quality optimization, whereas Miller gradually increases the quality, leading to quality switches. Increasing the number of connected clients per branch (N), introduces congestion by creating additional competition between the different clients. This decreases the average quality in terms of bitrate as shown in Fig. 15(b) and can even lead to freezes for the client-side heuristics as shown in Fig. 15(c). Even in the absence of cross traffic, the client-side heuristics are not able to achieve a comparable quality as in-networks driven optimization due to the competition between clients and over-estimation of available throughput, potentially leading to quality oscillations and buffer starvations.

Increasing the number of connected homes N and introducing additional cross traffic further demonstrates the benefits of in-network quality adaptation. Additional cross traffic from other sources than video traffic leads to wrong estimations for autonomic adaptation heuristics such as Miller as is shown in Fig. 16(a). This causes some of the clients to switch to a higher quality as is shown in Fig. 16(b), but unlike the wrong estimation predicted, this quality can not be maintained and ultimately leads to an increasing number of buffer starvations as is shown in Fig. 16(c). The in-network optimization also suffers from the



(a) Impact on average estimated MOS (b) Impact on average quality (Mbps) (c) Impact on buffer starvations (s)

Fig. 15. Impact of number of clients $|\mathcal{C}|$ for an uncongested scenario ($RTT = 40$ ms, $r = 100$, $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $B = 12$ s). Even in the absence of cross traffic, the client-side heuristics are not able to achieve a comparable quality as in-networks driven optimization due to the competition between clients.



(a) Impact on average estimated MOS (b) Impact on average quality (Mbps) (c) Impact on buffer starvations (s)

Fig. 16. Impact of number of homes N for a congested scenario ($RTT = 40$ ms, $r = 100$, $|\mathcal{H}_c| = 128$, $\tau = 2$ s, $B = 12$ s). The in-network optimization also suffers from the congested network ($N = 64$), but is able to maintain an average QoE that is 130% higher compared to the client-side heuristics and is still acceptable since the MOS is higher than 3.

congested network ($N = 64$), but is able to maintain an average QoE that is 3 times as high compared to the client-side heuristics and is still acceptable since the MOS is higher than 3. These results indicate the benefits of the additional knowledge for the *AVC Steered* adaptation heuristic provided by the in-network quality adaptation. Measuring the cross traffic along the delivery paths and estimating the achievable quality for each HAS session benefits QoE both in an uncongested scenario (7% gain) and a heavily congested scenario (340% gain) compared to traditional adaptation heuristics. This increase can mainly be attributed to a higher quality stability and the avoidance of buffer starvations. Fig. 16(c) shows how the QoE optimization has almost no freezes in the congested scenario, while for purely client-driven approaches the buffer starvations range from 6 s to 102 s, seriously degrading the QoE.

6. Conclusions

In HTTP Adaptive Streaming (HAS), competing clients impact the behavior of each other, causing wrong estimations of the available throughput. This leads to frequent quality oscillations and frame freezes, heavily impacting QoE. This paper therefore proposes a hybrid alternative, where in-network proxies monitor the available throughput using packet-based sampling and estimate the optimal quality selection for each client. The in-network optimization is driven by QoE by maintaining a history of previous decisions and maximizing the QoE in terms of both quality, quality oscillations and buffer starvations. This allows the QoE-driven in-network optimization to outperform standard autonomic quality heuristics by 30% to 43% and bitrate-based in-network optimization by 13% to 19%. Both a *Centralized* optimization and a scalable *Distributed* heuristic approach are presented. The *Centralized* optimization allows an optimal solution of the optimization problem, which is about 1.4% higher than the solutions obtained by the *Distributed* optimization. However, due to the longer execution times, the actual output in terms of QoE for the *Centralized* optimization is outperformed by the *Distributed* heuristic when the network size grows. The impact of the historic information as well as the optimization interval were evaluated, showing that a sufficiently large history ($\geqslant 100$) and optimizing with an interval equal to the average segment length yields significantly better results when compared to an autonomic quality selection heuristic with the same buffer size. Sampling the cross traffic with a sampling size of 100 yields a high correlation with the actual traffic, while limiting the overhead of sampling 1% of the packets. The proposed solution is able to achieve comparable QoE as a purely client-based quality selection with a buffer that is four times as small.

Acknowledgment

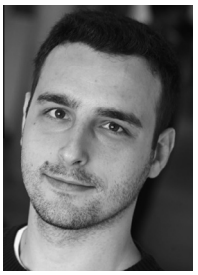
Niels Bouten is funded by a Ph.D. Grant of the Agency for Innovation by Science and Technology (IWT). This work was partly funded by FLAMINGO, a Network of Excellence Project (ICT-318488) supported by the European Commission under its Seventh Framework Programme, and the iMinds V-FORCE (Video 4K Composition and Efficient streaming) project (under IWT Grant agreement No. 130655).

References

[1] T. Stockhammer, Dynamic adaptive streaming over HTTP: standards and design principles, in: Proceedings of the second annual ACM conference on Multimedia systems, MMSys '11, ACM, New York, NY, USA, 2011, pp. 133–144.
[2] X. Wang, Network-assistance and server management in adaptive streaming on the internet, in: Proceedings of the Fourth W3C Web and TV Workshop, 2014.
[3] N. Bouten, S. Latré, W. Van de Meerssche, B. De Vleeschauwer, K. De Schepper, W. Van Leekwijck, F. De Turck, A multicast-enabled delivery framework for QoE assurance of over-the-top services in multimedia access networks, J. Netw. Syst. Manage. 21 (4) (2013) 677–706.
[4] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, P. Tran-Gia, A survey on quality of experience of HTTP Adaptive Streaming, IEEE Commun. Surv. Tut. PP (99) (2014). 1-1.
[5] R. Kuschnig, I. Kofler, H. Hellwagner, An evaluation of TCP-based rate-control algorithms for adaptive internet streaming of H.264/SVC, in: Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems, MMSys '10, 2010, pp. 157–168.
[6] S. Akhshabi, A.C. Begen, C. Dovrolis, An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP, in: Proceedings of the Second Annual ACM Conference on Multimedia Systems (MMSys), 2011, pp. 157–168.
[7] S. Akhshabi, L. Anantakrishnan, A.C. Begen, C. Dovrolis, What happens when HTTP adaptive streaming players compete for bandwidth?, in: Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), 2012, pp. 9–14.
[8] R. Houdaille, S. Gouache, Shaping HTTP adaptive streams for a better user experience, in: Proceedings of the 3rd Multimedia Systems Conference, MMSys '12, 2012, pp. 1–9.
[9] S. Benno, J.O. Esteban, I. Rimac, Adaptive streaming: the network HAS to help, Bell Labs Tech. J. 16 (2) (2011) 101–114.
[10] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, N. Shahmehri, Helping hand or hidden hurdle: proxy-assisted HTTP-based adaptive streaming performance, in: Proceedings of the IEEE International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013, pp. 182–191.
[11] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, R. Weber, Pushing CDN-ISP collaboration to the limit, SIGCOMM Comput. Commun. Rev. 43 (3) (2013) 34–44.
[12] K. Miller, E. Quacchio, G. Gennari, A. Wolisz, Adaptation algorithm for adaptive streaming over HTTP, in: Proceedings of the 19th International Packet Video Workshop (PV), IEEE, 2012, pp. 173–178.
[13] J. Jiang, V. Sekar, H. Zhang, Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '12, ACM, 2012, pp. 97–108.
[14] G. Tian, Y. Liu, Towards agile and smooth video adaptation in dynamic HTTP streaming, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, CoNEXT '12, 2012, pp. 109–120.
[15] J. Famaey, S. Latré, N. Bouten, W. Van de Meerssche, B. De Vleeschauwer, W. Van Leekwijck, F. De Turck, On the merits of SVC-based HTTP adaptive streaming, in: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), 2013, pp. 419–426.
[16] N. Bouten, S. Latré, J. Famaey, F. De Turck, W. Van Leekwijck, Minimizing the impact of delay on live SVC-based HTTP adaptive streaming services, in: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), 2013, pp. 1399–1404.
[17] C. Liu, I. Bouazizi, M.M. Hannuksela, M. Gabbouj, Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network, Signal Process.: Image Commun. 27 (4) (2012) 288–311.
[18] V. Adzic, H. Kalva, B. Furht, Optimized adaptive HTTP streaming for mobile devices, in: SPIE Optical Engineering+ Applications, International Society for Optics and Photonics, 2011, pp. 81350T–81350T.

[19] D.C. Robinson, Y. Jutras, V. Craciun, Subjective video quality assessment of HTTP adaptive streaming technologies, Bell Labs Tech. J. 16 (4) (2012) 5–23.
[20] J. Esteban, S.A. Benno, A. Beck, Y. Guo, V. Hilt, I. Rimac, Interactions between HTTP adaptive streaming and TCP, in: Proceedings of the 22Nd International Workshop on Network and Operating System Support for Digital Audio and Video, NOSSDAV '12, 2012, pp. 21–26.
[21] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, A.C. Begen, Server-based traffic shaping for stabilizing oscillating adaptive streaming players, in: Proceedings of the ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2013, pp. 19–24.
[22] X. Liu, A. Men, QoE-aware traffic shaping for HTTP adaptive streaming, Int. J. Multimedia Ubiquitous Eng. 9 (2) (2014) 33–44. http://www.sersc.org/journals/IJMUE/vol9_no2_2014/4.pdf..
[23] N. Bouten, S. Latré, J. Famaey, W. Van Leekwijck, F. De Turck, In-network quality optimization for adaptive video streaming services, IEEE Trans Multimedia 16 (8) (2014) 2281–2293.
[24] Z. Li, M. Sbai, Y. Hadjadj-Aoul, A. Gravey, D. Alliez, J. Garnier, G. Madec, G. Simon, K. Singh, Network friendly video distribution, in: Network of the Future (NOF), 2012 Third International Conference on the, 2012, pp. 1–8.
[25] J. Famaey, S. Latré, R. van Brandenburg, M.O. van Deventer, F. De Turck, On the impact of redirection on HTTP adaptive streaming services in federated CDNs, in: Proceedings of the 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS'13, vol. 7943, 2013, pp. 13–24.
[26] A. El Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, E. Steinbach, Quality-of-experience driven adaptive HTTP media delivery, in: Proceedings of the IEEE International Conference on Communications (ICC), 2013, pp. 2480–2485.
[27] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, N. Race, 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, ACM, 2013, pp. 15–20.
[28] C. Mueller, S. Lederer, C. Timmerer, H. Hellwagner, Dynamic adaptive streaming over HTTP/2.0, in: Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), 2013, pp. 1–6.
[29] S. Wei, V. Swaminathan, Low latency live video streaming over HTTP 2.0, in: Proceedings of Network and Operating System Support on Digital Audio and Video Workshop, NOSSDAV '14, 2014, pp. 37:37–37:42.
[30] S. Wei, V. Swaminathan, Cost effective video streaming using server push over HTTP 2.0, in: Proceedings of the IEEE 16th International Workshop on Multimedia Signal Processing (MMSP), 2014, pp. 1–5.
[31] N. Bouten, M. Claeys, S. Latre, J. Famaey, W. Van Leekwijck, F. De Turck, Deadline-based approach for improving delivery of SVC-based HTTP Adaptive Streaming content, in: Proceedings of the IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–7.
[32] N. Bouten, S. Latré, W. Van de Meerssche, K. De Schepper, B. De Vleeschauwer, W. Van Leekwijck, F. De Turck, An autonomic delivery framework for HTTP Adaptive Streaming in multicast-enabled multimedia access networks, in: Proceedings of the IEEE Network Operations and Management Symposium (NOMS), 2012, pp. 1248–1253.
[33] S. Petrangeli, M. Claeys, S. Latre, J. Famaey, F. De Turck, A multi-agent Q-Learning-based framework for achieving fairness in HTTP Adaptive Streaming, in: Proceedings of the IEEE Network Operations and Management Symposium (NOMS), 2014, pp. 1–9.
[34] R.K.P. Mok, X. Luo, E.W.W. Chan, R.K.C. Chang, QDASH: a QoE-aware DASH system, in: Proceedings of the 3rd Multimedia Systems Conference, MMSys '12, 2012, pp. 11–22.
[35] T. Schierl, C. Hellge, S. Mirta, K. Grneberg, T. Wiegand, Using H.264/AVC-based Scalable Video Coding (SVC) for real time streaming in wireless IP networks, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), 2007, pp. 3455–3458.
[36] S. Latré, F. De Turck, Joint in-network video rate adaptation and measurement-based admission control: algorithm design and evaluation, J. Netw. Syst. Manage. 21 (4) (2013) 588–622.
[37] Y.-M. Hsiao, S.-W. Yeh, J.-S. Chen, Y.-S. Chu, A design of bandwidth adaptive multimedia gateway for scalable video coding, in: Proceedings of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2010, pp. 160–163.
[38] T. Sutinen, J. Vehkaperä, E. Piri, M. Uitto, Towards ubiquitous video services through scalable video coding and cross-layer optimization, EURASIP J. Wirel. Commun. Netw. 2012 (2012) 25. http://jwcn.eurasipjournals.com/content/2012/1/25.
[39] H.-C. Lee, S.-M. Guu, On the optimal three-tier multimedia streaming services, Fuzzy Optim. Decis. Making 2 (1) (2003) 31–39.
[40] C. Hsu, M. Hefeeda, Optimal bit allocation for fine-grained scalable video sequences in distributed streaming environments, in: Electronic Imaging 2007, International Society for Optics and Photonics, 2007, pp. 650402–650402.
[41] M. Hefeeda, C.-H. Hsu, Rate-distortion optimized streaming of fine-grained scalable video sequences, ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) 4 (1) (2008) 2:1–2:28.
[42] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. De Vleeschauwer, W. Van Leekwijck, F. De Turck, QoE optimization through in-network quality adaptation for HTTP Adaptive Streaming, in: Proceedings of the International Conference on Network and Service Management (CNSM), 2012, pp. 336–342.
[43] A. Pras, L. Nieuwenhuis, R. van de Meent, M. Mandjes, Dimensioning network links: a new look at equivalent bandwidth, IEEE Netw. 23 (2) (2009) 5–10.
[44] R.d.O. Schmidt, R. Sadre, A. Sperotto, A. Pras, Lightweight link dimensioning using sFlow sampling, in: Proceedings of the 9th International Conference on Network and Service Management (CNSM), IEEE, 2013, pp. 152–155.
[45] R.G. Brown, Exponential smoothing for predicting demand, Oper. Res. 5 (1957). 145–145.
[46] C. Chatfield, The holt-winters forecasting procedure, Appl. Stat. 27 (3) (1978) 264–279.
[47] R. Shibata, Selection of the order of an autoregressive model by Akaike's information criterion, Biometrika 63 (1) (1976) 117–126.
[48] P.J. Rousseeuw, Least median of squares regression, J. Am. Stat. Assoc. 79 (388) (1984) 871–880.
[49] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (3) (2004) 199–222.
[50] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, K.R.K. Murthy, Improvements to the SMO algorithm for SVM regression, IEEE Trans. Neural Netw. 11 (5) (2000) 1188–1193.
[51] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and its empirical validation, in: Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM), 1998, pp. 303–314.
[52] J. De Vriendt, D. De Vleeschauwer, D. Robinson, Model for estimating QoE of video delivered using HTTP adaptive streaming, in: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), IEEE, 2013, pp. 1288–1293.
[53] J. De Vriendt, D. De Vleeschauwer, D.C. Robinson, QoE model for video delivered over an LTE network using HTTP adaptive streaming, Bell Labs Tech. J. 18 (4) (2014) 45–62.
[54] M. Claeys, S. Latré, J. Famaey, F. De Turck, Design and evaluation of a self-learning HTTP adaptive video streaming client, IEEE Commun. Lett. 18 (4) (2014) 716–719.
[55] L. Plissonneau, E. Biersack, A longitudinal view of HTTP video streaming performance, in: Proceedings of the Multimedia Systems Conference (MMSys), 2012, pp. 203–214.
[56] H. Riiser, P. Vigmostad, C. Griwodz, P. Halvorsen, Commute path bandwidth traces from 3G networks: analysis and applications, in: Proceedings of the 4th ACM Multimedia Systems Conference, 2013, pp. 114–118.
[57] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, F. De Turck, Design and optimisation of a FAQ-learning-based HTTP adaptive streaming client, Connect. Sci. 26 (1) (2014) 25–43.

Niels Bouten obtained a masters degree in computer science from Ghent University, Belgium, in June 2011. In August 2011, he joined the Department of Information Technology at Ghent University, where he is active as a Ph.D. student. His main research interests are the application of autonomic network management approaches in multimedia delivery. The focus of this research is mainly on the end-to-end Quality of Experience optimization, ranging from the design of a single autonomic control loop to the federated management of these distributed loops.



Ricardo de Oliveira Schmidt is a Postdoctoral researcher at the Design and Analysis of Communication Systems (DACS) group of the University of Twente, the Netherlands. He received a Ph.D. degree from the University of Twente, a M.Sc. degree in Computer Science from the Federal University of Pernambuco (UFPE), Brazil, and a B.Sc. degree in Computer Science from the University of Passo Fundo (UPF), Brazil. His research interests include network management, traffic measurements and analysis and traffic engineering.

Jeroen Famaey is affiliated with the department of Information Technology at Ghent University and iMinds as a postdoctoral researcher. He received his M.Sc. degree in computer science from Ghent University in 2007 and a Ph.D. in Computer Science Engineering in June 2012 on federated management of multimedia streaming services. His research interests include multimedia streaming services, inter-domain network management, and network virtualization.

Steven Latré is an assistant professor at the University of Antwerp, Belgium. He received a Master of Science degree in computer science from Ghent University, Belgium and a Ph.D. in Computer Science Engineering from the same university. His research activity focuses on autonomous management and control of both networking and computing applications. His recent work has focused on Quality of Experience optimization and management, distributed control and network virtualization. He has also been involved in several national and European research projects. He is author or co-author of more than 45 papers published in international journals or in the proceedings of international conferences. He is in the program committee of several conferences and regular reviewer for conferences and journals in this field.

Aiko Pras is professor in the area of Network Operations and Management at the University of Twente, the Netherlands and member of the Design and Analysis of Communication Systems Group. He received a Ph.D. degree for his thesis titled "Network Management Architectures". His research interests include network management technologies, network monitoring, measurements and security. He is chairing the IFIP Technical Committee on "Communications Systems", and is Coordinator of the European Network of Excellence on "Management of the Future Internet" (FLAMINGO). He is steering committee member of several conferences, including IM/NOMS and CNSM, and series/associate editor of ComMag, IJNM and TNSM.

Filip De Turck leads the network and service management research group at the Department of Information Technology of the Ghent University and iMinds in Belgium. His main research interests include scalable software architectures for network and service management, design and performance evaluation of novel QoE-aware multimedia delivery systems. He served as TPC chair of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2012) and the IFIP/IEEE Integrated Network Management Symposium (IM 2013). He is associate editor of the Journal on Network and System Management, the International Journal of Network Management and IEEE Transactions on Network and Service Management.