In-Network Quality Optimization for Adaptive Video Streaming Services

Niels Bouten, *Student Member, IEEE,* Steven Latré, *Member, IEEE,* Jeroen Famaey, *Member, IEEE,* Werner Van Leekwijck, and Filip De Turck, *Senior Member, IEEE*

***Abstract*—HTTP Adaptive Streaming (HAS) services allow the quality of streaming video to be automatically adapted by the client application in face of network and device dynamics. Due to their advantages compared to traditional techniques, HAS-based protocols are widely used for Over-The-Top (OTT) video streaming. However, they are yet to be adopted in managed environments, such as ISP networks. A major obstacle is the purely client-driven design of current HAS approaches, which leads to excessive quality oscillations, suboptimal behavior, and the inability to enforce management policies. Moreover, the provider has no control over the quality that is provided, which is essential when offering a managed service. This article tackles these challenges and facilitate the adoption of HAS in managed networks. Specifically, several centralized and distributed algorithms and heuristics are proposed that allow nodes inside the network to steer the HAS client's quality selection process. The algorithms are able to enforce management policies by limiting the set of available qualities for specific clients. Additionally, simulation results show that by coordinating the quality selection process across multiple clients, the proposed algorithms significantly reduce quality oscillations with a factor 5 and increase the average delivered video quality with at least 14%.**

I. Introduction

THE increasing popularity of Over-The-Top (OTT) multimedia services has led to the widespread adoption of HTTP-based streaming protocols. Such protocols have many advantages compared to traditional streaming methods, such as reuse of existing HTTP infrastructure (e.g., servers, proxies and caches), reliable transmission and firewall compatibility. Originally, progressive download techniques were used, allowing the user to start consuming the content after a few seconds of buffering. However, progressive download methods cannot cope with congestion, the highly fluctuating throughput of mobile networks or diverging characteristics of devices and networks. To overcome said problems, a new generation of HTTP-based streaming protocols, collectively referred to as HTTP Adaptive Streaming (HAS), was introduced. The offered content is split into a set of temporal segments, which are encoded at multiple bit rates. In traditional HAS, a rate adaptation algorithm, deployed at the client, is then used to select the bit rate of each segment, based on the current network conditions, buffer status and device capabilities.

State-of-the-art 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 OTT scenarios. However, academia and industry are showing a growing interest in the use of HAS in managed networks [1][1][2], for example by optimizing the delivery by applying in-network bitrate adaptation[3] or by deploying IP multicasting to ease the distribution of linear TV HAS services [2][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 competing behavior among those clients, resulting in incorrect throughput estimations, causing excessive quality oscillations and suboptimal decisions [3], [4], negatively impacting QoE [5]. Second, management policies, such as user subscription constraints and guarantees on the delivered quality, cannot be easily enforced [6], [7]. In order to facilitate adoption of HAS for the delivery of multimedia services in a managed environment, these challenges should be tackled.

A straightforward solution to the resource scarcity affecting streaming services could be to increase the physical capacity of the delivery network. These updates are however associated with high costs for the service provider, while an in-network optimization based solution does not affect these infrastructure costs. Since technologies (e.g. the advent of Ultra High Definition Television streaming) are constantly evolving, frequent infrastructure updates are required to cope with the ever increasing traffic demands. Physical infrastructure upgrades are time-consuming, therefore we believe that there should be a coexistence of both approaches to deal with future demand by intelligently managing resources in attendance of physical capacity updates.

This article proposes a hybrid approach where the rate

N. Bouten and F. De Turck are with the Department of Electrical and Computer Engineering, Ghent University - iMinds, Belgium. S. Latré and J. Famaey are with the Department of Mathematics and Computer Science, University of Antwerp - iMinds, Belgium. W. Van Leekwijck is with Alcatel-Lucent Bell Labs, Antwerp, Belgium
E-mail: niels.bouten@intec.ugent.be

[1] http://www.juniper.net/us/en/local/pdf/solutionbriefs/3510463-en.pdf
[2] http://www.rgbnetworks.com/pdfs/RGB-Velocix_Adaptive_Streaming_CDN_White_Paper_0911-01.pdf
[3] http://www.cachelogic.com/vx-portfolio/solutions/velocixeve
[4] http://www.velocix.com/vx-portfolio/solutions/video-optimised-architecture

adaptation algorithm is steered by an in-network component to address the aforementioned challenges. It is deployed on intermediary proxies and supports client-side rate adaptation algorithms by dynamically limiting the possible set of bit rates to select from. Currently an operator's multimedia delivery network typically contains several transparent caches and QoE measurement platforms which interpret HTTP headers and reconstruct HTTP adaptive streaming sessions in order to evaluate the end-to-end QoE. These platforms can be extended to not only measure the QoE, but also optimize the QoE by performing in-network quality optimization, thus requiring only limited extensions to the already available infrastructure. The proposed hybrid approach allows clients to still react upon sudden network changes or scarcity in device resources, while increasing the overall quality and stability. Moreover, it can enforce a wide range of management policies, allowing providers to specify priorities when allocating resources to a certain group of users. This translates into several concrete contributions. First, the in-network rate adaptation problem is formally defined. Second, an optimal centralized algorithm is proposed that solves the problem as an Integer Linear Program (ILP). Third, a scalable variant of the algorithm is introduced that can be distributed across multiple logically hierarchical intermediary proxies. Finally, a heuristic with significantly lower computational complexity is proposed.

The remainder of this article is structured as follows. Section II lists and discusses state of the art research on client-based and in-network HAS rate adaptation. Subsequently, the in-network rate adaptation problem is formally defined in Section III. Sections IV and V describe and evaluate the three algorithms proposed to solve this problem respectively. Finally, Section VI concludes the article.

II. Related Work

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 players have introduced their proprietary protocols such as Microsoft's Silverlight Smooth Streaming[5], Apple's HTTP Live Streaming[6] and Adobe's HTTP Dynamic Streaming[7]. More recently, a standardized solution has been proposed by MPEG, called Dynamic Adaptive Streaming over HTTP (DASH) [8]. Although differences exist between these implementations they are based on the same basic principles: a video is split up into temporal segments which are encoded at different quality rates, the autonomic video client heuristic then dynamically adapts the quality, based on metrics such as average throughput, delay and jitter. The drawback of this approach is of course that all control lays in the hands of the clients which strive to maximize their individual quality. From the providers perspective however, other factors such as minimization of costs and prioritization of users with higher subscription levels are of equal importance. Current HAS approaches do not support intervention during the quality assignment process which is fully dominated by the clients. The approach presented in this paper therefore focuses on managing the quality for HAS by the service provider.

The performance of HAS-based services can be improved by applying changes both at the client and the delivery network. Each commercial HAS implementation comes with a proprietary client heuristic. *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 [3]. Several heuristics have been proposed in literature as well, each focussing on a specific deployment. *Liu et al.* discuss a video client heuristic that is suited for a Content Delivery Network (CDN) by comparing the expected segment fetch time with the experienced segment fetch time to ensure a response to bandwidth fluctuations in the network [9]. *Andelin et al.* provide a heuristic which was specifically designed for Scalable Video Coding (SVC) and using a slope to define the trade-off between downloading the next segment and upgrading a previously downloaded segment [10]. In previous work [11][12], the authors evaluated different client heuristics both for Advanced Video Coding (AVC) and SVC, applying optimizations such as pipelined and parallel download scheduling. Several of the aforementioned authors indicate the impact of competing HAS clients on the quality oscillations, which are known to have a negative impact on Quality of Experience (QoE) [5]. Furthermore, most of the commercial client heuristics require a considerably large buffer to be able to react to network changes. This paper therefore aims at controlling the quality by introducing global QoE management, reducing drastically the number of quality oscillations and allowing to reduce the required buffer size. The presented approach is applicable to both AVC and SVC.

An autonomic delivery framework for HAS-based Live TV and Time Shifted TV (TSTV) was presented in previous work [13][2] 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. In [14], an overview of interesting use cases for applying SVC in a network environment are presented, among which the graceful degradation of videos 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.* proposes 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 [15]. In [16], a prototype of an intermediary adaptation node is proposed, where the media gateway estimates the available bandwidth on the client link and extracts

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

[6]Apple HTTP Live Streaming - http://tools.ietf.org/html/draft-pantos-http-live-streaming-13

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

the supported SVC-streams. Similar to this, the WiDASH proxy is responsible for in-network video adaptation and is able to perform global optimization over multiple concurrent HAS flows by prioritizing clients which have poor channel quality [17]. *Wirth et al.* discuss the optimization of multi-user resource assignment for DASH video transmission over the LTE downlink [18]. By deploying a cross-layer technique for allocating the resources at the base station and taking into account the specific information of the video sessions, the number of playout starvations can be considerably reduced. In *Parakh et al.*, the authors propose a game theoretic approach towards decentralized bandwidth allocation for video streams in wireless systems, where users are charged for bandwidth resources proportionally to the requested bit-rate [19]. *Situnen et al.* propose dropping video layers based on their priority when network congestion arises for scalable video streaming over wireless networks [20]. 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 clients subscription. This allows the service provider to control the QoE on a per subscriber level, and thus offering different subscription types for the VoD HAS services.

Lee et al. describe a three-tier streaming service where multiple clients are connected through multiple intermediate proxies to a multimedia server [21]. The authors only consider live streaming, if however VoD streaming would be targeted, the streaming service can no longer be delivered in an efficient way using multicast streaming, since a lot of requests are on unpopular content which is infrequently requested. This causes the content to be delivered using unicast transport from origin to regional servers and thus having the risk of running into bandwidth bottlenecks on these links as well, which is not addressed within the cited paper. 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. Unlike Real Time Streaming Protocol (RTSP) en Real Time Transport Protocol (RTP) based streaming protocols, there is no server-side bitrate adaptation required, the client decides autonomously which quality it will select, based on the current state of the network and from a list of permitted qualities, selected from within the network. This also implies that if a client struggles to achieve its assigned quality level, for example due to poor wireless connection quality or limited CPU resources, it can still decide to switch to a lower quality.

In [22][23], 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

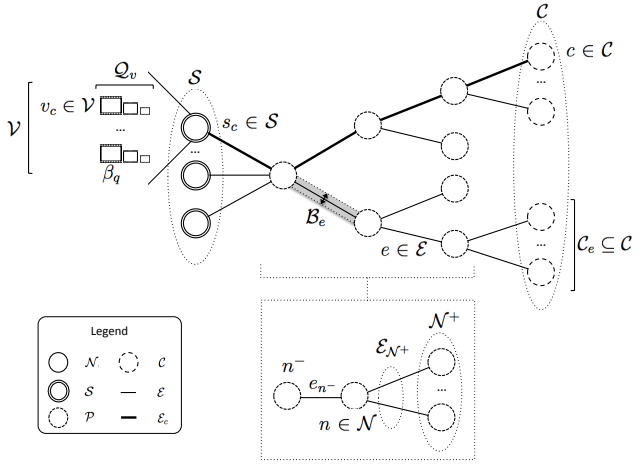


Fig. 1: Graphical representation of variables and assumptions.

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. *Akhshabi et al.* propose server-side rate adaptation to cope with unstable streaming players due to ON-OFF patterns when they compete for bandwidth [24]. The systems detect sudden rate fluctuations in the client playout and try to solve them by shaping the sending rate at the server to resemble the bitrate of the stream. 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. This is because we can use more detailed in-network information. This article is an extension to our previous work on in-network quality management for HAS [25]. However, the problem formulation is generalized and we significantly extended the approach with a centralized, distributed and relaxed optimization. Furthermore, the previous work only considered simple topologies with a single bottleneck where multiple clients directly connect to the server. Whereas this article supports more complex topologies with multiple levels, multiple bottlenecks and intermediary proxies, as well as asymmetric topologies.

III. Formal Problem Description

Providers are exploring how they can offer VoD HAS services next to traditional TV services over their managed network environment. HAS services offer the same content at multiple qualities, each at their corresponding rate. This allows providers to perform QoE management by adjusting each sessions quality level, based on the current network utilization. At peak times, the consequences of an inadequate amount of resources in the network, can thus be anticipated by reducing the quality of individual streaming sessions, while still allowing admittance of all users.

TABLE I: Variables used for the rate decision

α_s	Weighing factor to model the tradeoff between quality and quality switches
$\mathcal{B}_e$	Bandwidth reserved for HAS traffic for edge $e \in \mathcal{E}$
β_q	Bitrate associated with layer $q \in \mathcal{Q}$
β_{max}	Highest bitrate in $\mathcal{Q}$
$\mathcal{C} \subset \mathcal{N}$	The set of HAS VoD clients
$\mathcal{C}_e \subseteq \mathcal{C}$	The set of clients in the service delivery tree for which the VoD traffic traverses edge $e \in \mathcal{E}$
$\mathcal{C}_n \subseteq \mathcal{C}$	The set of clients in the service delivery tree for which the VoD traffic traverses node $n \in \mathcal{N}$
$\mathcal{E}$	The set of edges in the service delivery tree
$\mathcal{E}_c \subseteq \mathcal{E}$	The unique delivery path from server s_c to client $c \in \mathcal{C}$
e_{n^-}	The edge connecting node n to its predecessor n^-
$\mathcal{E}_{n^+} \subseteq \mathcal{E}$	The set of edges connecting node n to its successors $\mathcal{N}_{n^+}$
$\mathcal{H}_c$	The history of previous quality decisions for client $c \in \mathcal{C}$
$h_{c,q,t} \in \mathcal{H}_c$	Binary variable indicating wether client $c \in \mathcal{C}$ was assigned quality $q \in \mathcal{Q}$ at time t
$\mathcal{N}$	The set of nodes in the service delivery topology
$\mathcal{N}_{n^+} \subset \mathcal{N}$	The set of successors of node n
n^-	The predecessor of node n
$\mathcal{P} \subset \mathcal{N}$	The set of proxies in the delivery tree
$\mathcal{Q}$	Available quality rates for video
$\mathcal{Q}_v \subseteq \mathcal{Q}$	Available quality rates for video $v \in \mathcal{V}$
$\mathcal{S} \subset \mathcal{N}$	The VoD access server
$s_c \in \mathcal{S}$	The VoD access server for client c
$\mathcal{V}$	The set of available videos via VoD server $\mathcal{S}$
$v_c \in \mathcal{V}$	The video $v \in \mathcal{V}$ for which client c is requesting access

A. Definition of variables and assumptions

Figure 1 gives an overview of the problem variables and assumptions. Let us consider an access 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[8]. Note that typical access networks are using a logical tree for their delivery, although the underlying physical network is not a tree due to replication concerns. Every node $n \in \mathcal{N}$ has an incoming edge $e_{n^-} \in \mathcal{E}$ connecting it to its predecessor $n^- \in \mathcal{N}$ and a set of outgoing edges $\mathcal{E}_{\mathcal{N}^+} \subseteq \mathcal{E}$ connecting it to its successors $\mathcal{N}^+ \subset \mathcal{N}$. Every edge $e \in \mathcal{E}$ has an associated bandwidth capacity $\mathcal{B}_e$ reserved for HAS traffic.

The servers host a set of videos $\mathcal{V}$. Every video $v \in \mathcal{V}$ has an associated set of quality representations $\mathcal{Q}_v \subseteq \mathcal{Q}$. Moreover, every quality representation $q \in \mathcal{Q}$ has a bit rate β_q. Every client $c \in \mathcal{C}$ has an associated origin server $s_c \in \mathcal{S}$, a unique delivery path $\mathcal{E}_c \subseteq \mathcal{E}$ from that server, and a video $v_c \in \mathcal{V}$. The set of clients that have an edge $e \in \mathcal{E}$ as part of their delivery path $\mathcal{E}_c$, is represented by $\mathcal{C}_e \subseteq \mathcal{C}$. In summary, Table I lists the symbols introduced throughout this section.

B. Integer Linear Programming formulation

The problem 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 $q \in \mathcal{Q}_{v_c}$ is selected for client $c \in \mathcal{C}$, and 0 otherwise. The decision variables are subject to the following two constraints:

$$\forall c \in \mathcal{C}, \forall q \in \mathcal{Q}_{v_c} : a_{c,q} \in 0,1 \tag{1}$$

$$\forall c \in \mathcal{C} : \sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} = 1 \tag{2}$$

The above constraints state that the decision variables are boolean values and that only one quality representation can be selected per client.

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}, probability of packet loss p, delayed ACK number of b and average retransmission timeout T_0 can be approximated by the following [26]:

$$B(p) \approx \min \left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)} \right) \tag{3}$$

The maximum achievable TCP throughput for client c is thus limited by its window size $W_{max,c}$ and its RTT_c. Both parameters can be estimated or measured at the client and forwarded to the in-network control proxy. When p values are low, which is the case in fixed networks, the achievable throughput is primarily limited by the first term. To limit the overhead of acquiring packet loss probabilities for all clients, only the first part of the TCP estimator is considered. Therefore, the following constraint is added, limiting the end-to-end achievable throughput for each client:

$$\forall c \in \mathcal{C} : \sum_{q \in \mathcal{Q}_c} a_{c,q} \times \beta_q \leq \frac{W_{max,c}}{RTT_c} \tag{4}$$

When N TCP-connections use the same bottleneck link, *Altman et al.* state that the maximum aggregated achievable throughput that can be obtained is a factor of the link capacity B_e [27]:

$$B_{max} \approx \left(1 - \frac{1}{1+cN}\right) B_e \tag{5}$$

Where $c = \frac{1+d}{1-d}$ with d the fraction with which the send rate is decreased when congestion arises. We use this approximation of the maximum achievable throughput to limit the aggregated allocated rate of the different clients:

$$\forall e \in \mathcal{E} : \sum_{c \in \mathcal{C}_e} \sum_{q \in \mathcal{Q}_c} a_{c,q} \times \beta_q \leq \left(1 - \frac{1}{1 + c\,|C|}\right) B_e \tag{6}$$

As stated, the objective aims to maximize the global QoE. This is obviously a broad term that can be interpreted in

[8] An example is the Triple Play Service Delivery Architecture from Alcatel-Lucent (http://goo.gl/4aZVvf), which is used by over 50 operators worldwide (http://goo.gl/kHMY1b)

a multitude of ways. As such, a generic objective function is proposed that can be adapted to the service provider's optimization policy, represented by the function $F(\cdot)$:

$$\max \sum_{c \in \mathcal{C}} \sum_{q \in \mathcal{Q}_{v_c}} F(a_{c,q}) \tag{7}$$

For example, the provider could aim to maximize the total delivered bit rate, which can be translated into the following objective function:

$$\max \sum_{c \in \mathcal{C}} \sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q \tag{8}$$

The operator could also decide to optimize the fairness among the connected clients. This can be achieved by adopting proportional fairness, as proposed by *Kelly et al.* [28][29]. A vector of rates $A_c = (\sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q, c \in \mathcal{C})$ is proportionally fair if it is feasible, according to Equation (4) and (6) respectively, and if for any other feasible vector A_c^*, the aggregate of the proportional changes is zero or negative:

$$\sum_{c \in \mathcal{C}} \frac{A_c^* - A_c}{A_c} \leq 0 \tag{9}$$

According to *Wei et al.*, the fair bandwidth allocation can be represented by a local maximum of the logarithmic utility function [30]. Since this function is differentiable and strictly concave, it has only one maximum, which is therefore also the global maximum. Therefore, the objective of a proportionally fair bandwidth allocation can be expressed by:

$$\max \sum_{c \in \mathcal{C}} \log \Big(\sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q \Big) \tag{10}$$

Another objective could be to minimize the number of switches since they have a negative impact on overall QoE. This requires maintaining a history $\mathcal{H}_c$ of previous quality decisions for each client $c \in \mathcal{C}$ where $h_{c,q,t} = a_{c,q}$ at time t. For quality switches, not only the frequency of switching is important, but also the distance between quality selections affects the overall quality [5]. Therefore, to assess the impact of distance in quality, the variation in quality over the history $\mathcal{H}_c$ is taken into account. The following weighted sum is used to model the impact on switching behavior, where μ represents the average quality, σ introduces a penalty for quality switching and α_s represents a weighing factor used to emphasize either the impact of quality or the switching behavior:

$$\max \alpha_s \times \mu - (1 - \alpha_s) \times \sigma \tag{11}$$

Since the decision variables $a_{c,q}$ are binary variables, the calculation of the objective function can be simplified by calculating $\mu_{c,q}$ and $\sigma_{c,q}$ for each client c and it's associated quality range $\mathcal{Q}_c$. The quality rates are normalized with respect to the highest quality rate β_{max}.

$$\forall c \in \mathcal{C}, \forall q \in \mathcal{Q}_v : \mu_{c,q} = \frac{1}{|\mathcal{H}_c| + 1} \left(\frac{\beta_q}{\beta_{max}} + \sum_{h_{c,t} \in \mathcal{H}_c} \sum_{q \in \mathcal{Q}_c} \frac{h_{c,q,t} \times \beta_q}{\beta_{max}} \right) \tag{12}$$

In this way, the use of quadratic terms in the objective function is avoided. The penalty σ for switching between qualities can be calculated as follows:

$$\forall c \in \mathcal{C}, \forall q \in \mathcal{Q}_v : \sigma_{c,q} = \sqrt{\frac{1}{|\mathcal{H}_c| + 1}} * \sqrt{\left(\frac{q \times \beta_q}{\beta_{max}} - \mu_{c,q} \right)^2 + \sum_{h_{c,q,t} \in \mathcal{H}_c} \sum_{q \in \mathcal{Q}_c} \left(\frac{h_{c,q,t} \times \beta_q}{\beta_{max}} - \mu_{c,q} \right)^2} \tag{13}$$

The total objective can then be expressed as:

$$\max \sum_{c \in \mathcal{C}} \sum_{q \in \mathcal{Q}_v} a_{c,q} \times (\alpha \times \mu_{c,q} + (1 - \alpha) \times \sigma_{c,q}) \tag{14}$$

IV. Algorithms

A. Centralized ILP

The ILP formulation described in Section III-B can be used to optimize the quality assignments using a centralized controller. It requires as input the knowledge of the delivery network topology $(\mathcal{N}, \mathcal{E})$, link constraints $\mathcal{B}_e$, the set of clients $\mathcal{C}_e$ for which the VoD traffic traverses an edge $e \in \mathcal{E}$ and the characteristics of these clients $(W_{max,c}, RTT_c)$. Solving said ILP formulation will yield a set of optimal quality assignments $a_{c,q}$ for each client c and quality level q. These assignments are optimal in the sense that they maximize the objective $\sum_{c \in \mathcal{C}} \sum_{q \in \mathcal{Q}} F(a_{c,q})$ subject to the constraints described in Equations (1), (2), (4) and (6). As we assume there is a constant bitrate reserved for HAS traffic on each edge, the *Centralized* optimization is executed each time a newly joined client requests a manifest file or if a client becomes inactive by leaving the delivery network.

B. Distributed ILP

The number of constraints for the centralized ILP grows with an increasing depth of the service delivery topology tree. Consider a topology tree with k child nodes per node and l levels (thus $l = log_k|C| + 1$), the total number of edge constraints is then equal to $\sum_{i=0}^{l-1} k^i$ which can be written as $\frac{1-k^l}{1-k}$. This leads to an exponentially increasing model size with the number of levels in the delivery tree, affecting the calculation time. Since our approach would be deployed in an operational setting, the decision process should be able to determine quality allocations in real-time. Therefore, we propose a distributed approach, where each proxy locally determines the optimal allocation constrained by the local edge

capacities and where the global solution is an aggregation of these local solutions. The advantages of this approach are threefold. First, each node only needs to have local information on the properties of the upstream edge e_{n^-} and the video flows for clients $\mathcal{C}_n$ traversing this node. Second, since each proxy locally solves the optimization problem, the number of constraints does not increase with the tree size, leading to small local ILP models. Third, proxies residing at the same level in the topology tree can optimize their local ILP models in parallel, leading to faster global optimization.

The distributed ILP algorithm uses a bottom-up approach for the distributed solution where each node n locally optimizes the allocation problem and forwards the solution to its predecessor n^- in the delivery tree. The solution at node n is constrained by the limitations of it's successors $\mathcal{N}_{n^+}$. These limitations are determined by the combination of the optimal solutions $a_{c,q}^{n^+}$ of each node $n^+ \in \mathcal{N}_{n^+}$. For each client $c \in \mathcal{C}_n$, $sq_{n,c}$ determines the maximum quality a client is able to receive according to the successors of n. The local ILP formulation is then constrained by (1) and (2) as before but only for $c \in \mathcal{C}_n$, while the following determines that the selected quality for a client c is not allowed to violate the limitations determined by the successors in $sq_{n,c}$:

$$\forall c \in \mathcal{C}_n : \sum_{q \in \mathcal{Q}_{v_c} : q > sq_{n,c}} a_{c,q} = 0 \tag{15}$$

Furthermore, the total consumed bandwidth for the allocation is not allowed to exceed the HAS capacity $\mathcal{B}_{e_{n^-}}$ for the edge e_{n^-}, connecting n to its predecessor n^-.

$$\sum_{c \in \mathcal{C}_n} \sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q \leq \mathcal{B}_{e_{n^-}} \tag{16}$$

The calculation time can be limited by taking advantage of some specific properties of the problem. First, since the distributed optimization is only performed when a client joins or leaves the service delivery network, only the proxies $p \in \mathcal{P}_c$ on the delivery path for client c are required to perform local optimization. For other proxies, the previous optimal solutions remain valid since there are no changes in (15) and (16). This limits the number of local optimizations during the execution of the distributed algorithm to the number of levels $l = \log_k |C| + 1$. Second, the solutions determined by the predecessors of n are optimal and since these solutions are independent, their combination is optimal. This means that if the combination of the solutions $a_{c,q}^{n^+}$ is feasible, there is no need to perform local optimization. In the best case, where there is only one bottleneck in the network, we have thus reduced the number of local optimization steps to 1. For the worst case scenario, where the upstream bottleneck becomes tighter at every node, the maximum number of local optimizations is limited by $l = \log_k |C| + 1$ as before.

There is a communication overhead when using the distributed optimization, which requires sending a list of clients for which the video traffic traverses the node and the decision on the quality level per client. Since this list (or any combination made of it by any node) contains at most n entries, with n the number of connected clients, the number of required bits per client entry is $\lceil \log_2 n \rceil$ to uniquely identify the clients and $\lceil \log_2 q \rceil$ to identify their selected quality with q the highest number of available qualities. For each level in the topology, the information exchanged will thus be equal to $n\lceil \log_2 n \rceil + \lceil \log_2 q \rceil$ bits and since $log_k n + 1$ is the number of levels, the total information exchange is $n \log_k n(\lceil \log_2 n \rceil + \lceil \log_2 q \rceil)$. If there are 1000000 clients in the network and k is equal to 10, then the total communication overhead is 16.5 MB per optimization. If the lowest quality representation is 1Mbps, the total amount of traffic flowing through the network per second is in the order of Tbps, leading to a negligible communication overhead for the optimization.

C. Relaxed Distributed Linear Program (LP) Formulation

Solving the distributed ILP optimally in a single node can however lead to execution times in the order of seconds when the number of VoD flows crossing that node becomes large. We can increase the execution speed at the expense of a sub-optimal solution by moving from an Integer LP formulation to a Relaxed LP formulation by relaxing the boolean constraints on the variables $a_{c,q}$ in (1) by only requiring $a_{c,q}$ to belong to the interval $[0, 1]$:

$$\forall c \in \mathcal{C}, \forall q \in \mathcal{Q}_{v_c} : 0 \leq a_{c,q} \leq 1 \tag{17}$$

This relaxation can be solved in polynomial time but at the cost of optimality. The variables $a_{c,q}$ do not longer unambiguously define which quality each client is allowed to download, therefore a heuristic is required to transform the optimal floating point solution into an integer solution. Algorithm 1 shows an overview of the heuristic procedure. First, the clients of the solution matrix A are sorted according to two criteria: first on the proximity of the floating point solution to the integer solution and subsequently on the contribution to the objective (line6). The proximity of a client solution is defined as $P_c = \min_{q \in \mathcal{Q}_{v_c}}(1 - a_{c,q})$, where $P_c = 0$ indicates that the floating point client solution can be immediately transformed into an integer client solution (line 2). The contribution to the objective is calculated as $\sum_{q \in \mathcal{Q}_{v_c}} F(a_{c,q})$ and gives an indication of the weight of a single client in the global objective (line 3). Second, the maximum quality each client is allowed to download determined by the solution $a_{c,q}$, is set to $mq_c = \max_{q \in \mathcal{Q}_{v_c}}(a_{c,q} > 0)$ (line 4). This assures that the limitations of the successors $\mathcal{N}_{n^+}$ are not violated which could lead to an infeasible solution further down the delivery tree. Third, for each client $c \in \mathcal{C}_n$ w.b.e calculate their contribution to the constraint as $\sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q$ and add it to the budget B (line 9). We then optimize the following problem for client c (line 12 to 19):

$$\max \quad \sum_{q \in \mathcal{Q}_{v_c}} F(a_{c,q}) \tag{18}$$

$$\text{subject to} \quad \sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q < B \tag{19}$$

$$a_{c,q} \in \{0, 1\} \forall q \in \mathcal{Q}_{v_c}. \tag{20}$$

Algorithm 1: Overview of the heuristic to transform a floating-point solution to an integer solution

for all $c \in \mathcal{C}$ **do**
  $Proximity_c \leftarrow \min_{q \in \mathcal{Q}_{v_c}} (1 - a_{c,q})$
  $Contribution_c \leftarrow \sum_{q \in \mathcal{Q}_{v_c}} F(a_{c,q})$
  $mq_c \leftarrow \max_{q \in \mathcal{Q}_{v_c}} (a_{c,q} > 0)$
end for
$Sort(a_{c,q}, Proximity_c, Contribution_c)$
$sol \leftarrow 0$
for all $c \in \mathcal{C}$ **do**
  $B \leftarrow B + \sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q$
  $MaxObj \leftarrow 0$
  $setq \leftarrow 0$
  for all $q \in \{0, mq_c\}$ **do**
    if $B \geq \sum_{q \in \mathcal{Q}_{v_c}} a_{c,q} \times \beta_q$ **and** $\sum_{q \in \mathcal{Q}_{v_c}} F(a_{c,q}) > MaxObj$ **then**
      $MaxObj \leftarrow \sum_{q \in \mathcal{Q}_{v_c}} F(a_{c,q})$
      $setq \leftarrow q$
    end if
  end for
  $sol_{c,setq} \leftarrow 1$
  $B \leftarrow B - \beta_{setq}$
end for

V. Performance Evaluation

A. Experiment Setup

A VoD HAS scenario was implemented by using an NS3 based simulation framework, capable of the transmission of HAS video [25]. This framework has been extended with support for QoE management at both the servers and the proxies. For the HAS Clients we used the *AVC MSS* algorithm, which is based on the implementation of an open source version of the algorithm of the Microsoft Smooth Streaming (MSS) video player[9] and is extensively described by *Famaey et al.* [11]. The server-based traffic shaping method proposed by *Akhshabi et al.* was also implemented in combination with *AVC MSS* and is referred to as *AVC MSS Rate Controlled* [24]. This approach tries to reduce quality oscillations when multiple clients compete for bandwidth, by dynamically adjusting the shaping rate when oscillations are detected. For further implementation details we refer to the paper by *Akhshabi et al.* [24]. Furthermore, an additional client heuristic was implemented, which downloads each segment using the QoE management quality decision and checks if these decisions are feasible, given the measured throughput at the client. If the measurements indicate that the proposed quality is not achievable, the proposed client heuristic will select the highest sustainable quality based on the local throughput measurements. We refer to the aforementioned heuristic as *AVC Steered*. Both client heuristics dispose of a buffer of 10 second, thus accommodating space for 5 segments. Both client implementations use persistent connections to avoid the impact incurred by the setup and teardown of TCP-connections. The congestion window at the server was limited to avoid unfair sharing of the bottleneck bandwidth [31]. The configured congestion window allows transmitting segments at a rate that is two times bigger than the maximum bitrate of the stream.

[9]Source available from https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming

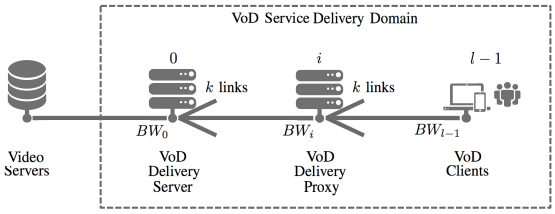


Fig. 2: Network topology, modeling a typical video service delivery network.

TABLE II: Overview of the quality layers for the Big Buck Bunny video

Quality Layer Index	Average Bitrate (kbps)	Average PSNR (dB)	Average SSIM
0	300	32.04	0.8746
1	427	32.72	0.8861
2	608	34.41	0.9108
3	866	35.71	0.9249
4	1233	36.88	0.9387
5	1636	37.64	0.9469
6	2436	40.07	0.9608

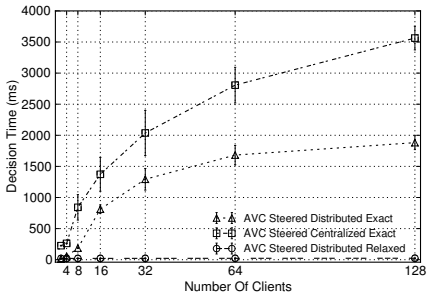
As discussed before, we modeled the delivery network as a tree-based topology, where a video server $\mathcal{S}$ streams videos to a set of clients $\mathcal{C}$. The number of branches on each level was set to k, leading to $l = log_k|\mathcal{C}| + 1$ levels in the tree. For each level the available HAS bandwidth was varied depending on the bottleneck factor BF. For each level the bandwidth $BW_i = (k * BF)^{l-i-1} * BW_{l-1}$, meaning that if $BF = 0.8, l = 4, k = 5, BW_{l-1} = 4Mbps$, the bandwidth for a edge at level 1 is $BW_1 = 64Mbps$. Figure 2 gives a graphical overview of the service delivery network topology. Clients are started using a Weibull startup process with shape 2.5 and mean of $300s$. The average RTT for each client c is set to $RTT_c = 40ms$ unless stated otherwise [32]. VoD Service Delivery Domain The Big Buck Bunny video[10] was encoded at 7 different quality rates and divided into 200 segments with an average duration of 2 seconds. Table II gives an overview of the different quality layers, their associated bitrates, average Peak Signal-to-noise Ratio (PSNR) and Structural Similarity (SSIM) values. During the evaluations we used the SSIM metric introduced by *Wang et al.*, which is motivated by the assumption that human visual perception is highly adapted for extracting structural information. It has been shown to have a high correlation with image [33] and video quality [34].

B. Implementation details

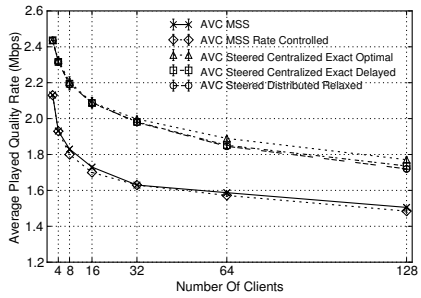
The IBM CPLEX[11] solver was used to implement and solve the proposed binary ILP-problems for both the centralized and distributed algorithm, as well as the relaxed distributed LP-problem. Since NS3 is an event-based simulation framework,

[10]Big Buck Bunny available from http://www.bigbuckbunny.org/

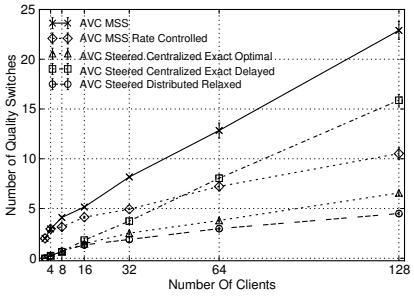
[11]IBM ILOG CPLEX Optimizer: http://www.ibm.com/software/integration/optimization/cplex-optimizer/



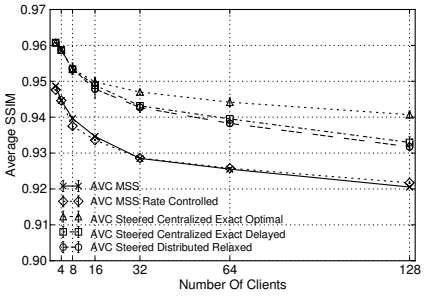
(a) Impact on average decision time



(b) Impact on average played quality bitrate



(c) Impact on average number of quality switches



(d) Impact on average SSIM

Fig. 3: Impact of number of clients, using a topology with $k = 2$, $BW_{l-1} = 3Mbps$, $BF = 0.9$

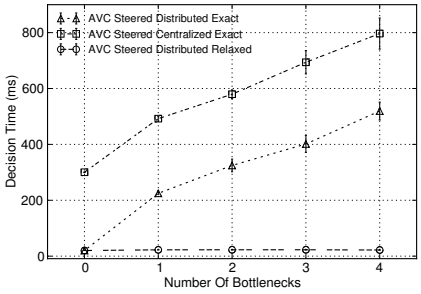


Fig. 4: Impact of number of bottlenecks in the topology on the average decision time, using a topology with $k = 5$, $BW_{l-1} = 2Mbps$, $|\mathcal{C}| = 125$ and $l = 4$

the execution times of the optimizations were measured as t_{exec} and used to schedule the release of the calculated configuration at $t_{current} + t_{exec}$. We executed the different experiments using two modes: *Delayed* ensuring that the configurations only become available when optimization is finished and *Optimal* which is agnostic to execution times and installs the configuration immediately.

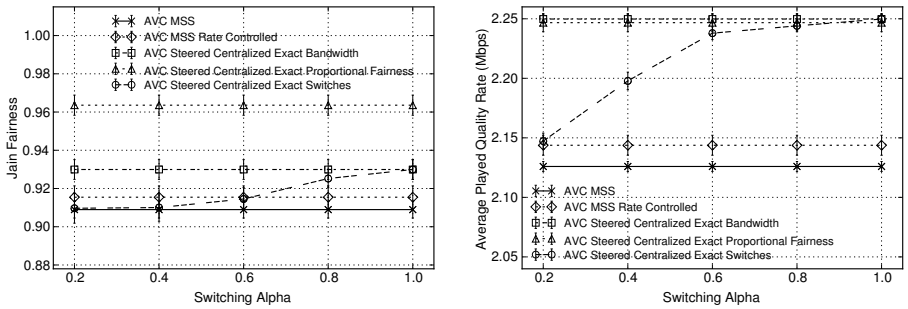
However, since executions can take several seconds, clients can join during this period, leading to two implications for the system: first, since there is no immediate quality configuration available, the *Steered Client* heuristic will not be able to select the optimal quality and second, new optimizations, which take into account more recent network configurations are delayed by the execution of the previous one. Both problems can be solved by performing a heuristic optimization first, allowing to quickly install a suboptimal quality configuration and replace it with the optimal configuration when it becomes available. This also allows us to preempt a QoE optimization when additional requests lead to a changed environment and the optimal solution would be outdated. The heuristic optimization checks if the previous limitations in combination with the additional client are feasible for each edge e, if not the client qualities for $c \in \mathcal{C}_e$ are lowered by one level until the solution is feasible again.

C. Evaluation Details

The performance of the centralized ILP, distributed ILP and relaxed distributed LP was evaluated in terms of service assurance, quality delivery and oscillations. Also the impact of the different approaches on the decision time was quantified. The network size, the number of bottlenecks, the optimization objective, Round Trip Time (RTT) and number of servers were varied. During the evaluations, the maximization of the video bitrates, defined by (8), was used as a network provider's objective. We refer to the centralized and distributed ILP optimization as *Centralized Exact* and *Distributed Exact* respectively, while the relaxed optimization is indicated as *Distributed Relaxed*. All experiments use the quick optimization heuristic, providing a preliminary decision on the selected qualities, which is updated when the optimization calculation is finished. Therefore these results are installed with a delay and are referred to as *Delayed* decisions. When it is explicitly stated that the configurations are installed based on the *Optimal* selection, we show the results that could be achieved with the optimal configurations in the presence of network variations and buffering. All of the following results are averaged over $n = 10$ iterations, with the graphs showing the 95% confidence intervals $[\bar{X} - 1.96\frac{\sigma}{\sqrt{n}}, \bar{X} + 1.96\frac{\sigma}{\sqrt{n}}]$ [35].

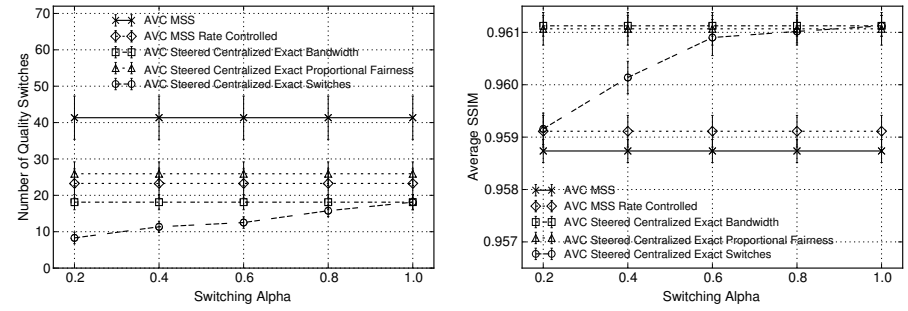
D. Impact of Number of Clients

In this section, we motivate the deployment of in-network quality adaptation algorithms for HAS delivery networks and quantify the impact of the delivery tree size on the in-network adaptation performance. For the following experiments we set the number of child nodes k for each node to 2 and varied the number of clients $|\mathcal{C}|$ in the interval $[2, 128]$, the bandwidth



(a) Impact on Jain Fairness

(b) Impact on average played quality bitrate



(c) Impact on average number of quality switches

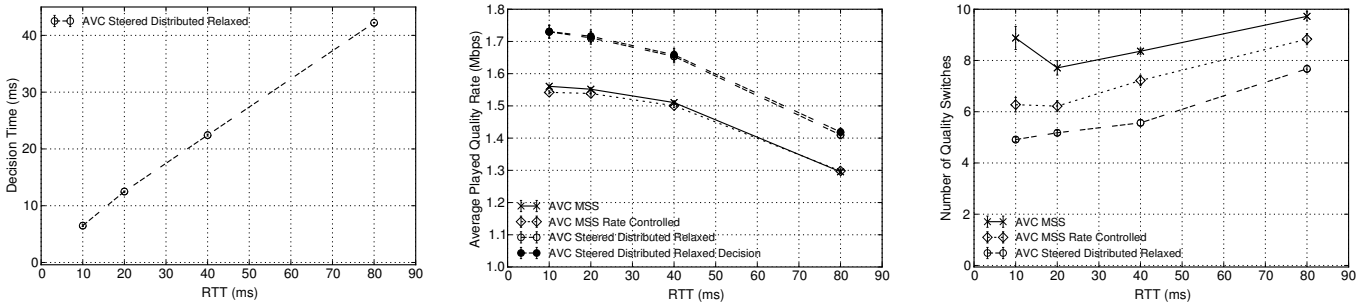
(d) Impact on average SSIM

Fig. 5: Impact of *Switching Alpha* α_s, using a topology with $k = 5$, $BW_{l-1} = 4Mbps$, $BF = 0.8$, $|\mathcal{C}| = 125$ and $l = log_k|\mathcal{C}| + 1 = 4$

on the first link BW_{l-1} is set to $3Mbps$, while the bottleneck factor is set to $BF = 0.9$, leading to a bottleneck at the server $\mathcal{S}$ of approximately $184Mbps$.

Figure 3(a) illustrates the impact of the number of clients on the QoE optimization execution times. For both the *Centralized Exact* and *Distributed Exact* optimization, the execution times show a logarithmic increase with the number of clients and thus a linear increase with respect to the number of levels in the topology tree. For 7 levels, the average optimization time for the *Centralized Exact* algorithm is $3563.34ms$, for the *Distributed Exact* algorithm it is $1881.75ms$, while the *Distributed Relaxed* algorithm only requires $23.52ms$ (including an average one-way delay of $20ms$ for forwarding local solutions). These results indicate that only the distributed heuristic approach is feasible for medium to large size problems.

The impact of the in-network quality management on the average bitrate, the number of switches and average quality in terms of SSIM is shown in Figure 3(b), 3(c) and 3(d) respectively. The results show a significant improvement on the average played bitrate over traditional client-based heuristics ranging from 14% to 23%, while the number of switches can be reduced with a factor of 1.5 to 5. Since client-based heuristics have only a local view on the network situation, they require safety measures to avoid buffer starvations and quality oscillations, resulting in underestimations of the available throughput and thus underutilization of the available bandwidth. Combining a client-driven approach with server-based traffic shaping allows maintaining the same quality as with a client-driven approach, while reducing the number of switches up to a factor 2.5. In-network quality adaptation is able to react more quickly to changing network environments and allows to fully utilize the available bandwidth. The *Distributed Relaxed* optimization is able to reduce the number of switches with a factor 5 compared to the traditional client-based heuristic and with a factor 2.5 when traditional client-based approaches are combined with server-based rate shaping. The *Centralized Exact Delayed* optimization is able to achieve a slightly higher quality than the *Distributed Relaxed* optimization, but shows an increased number of switches which can be accounted to the longer execution times causing the clients to choose suboptimal quality configurations while waiting for the optimal configuration. The *Centralized Exact Optimal* optimization shows the theoretical configuration that could be achieved in absence of the long execution times. These results show a penalty of about 3% in terms of average quality rate when using the *Distributed Relaxed* optimization due to suboptimal solutions attained by the rounding heuristic. As shown in Figure 3(d), also the achieved average SSIM, is slightly lower for the *Distributed Relaxed* optimization. The average number of switches is slightly higher for the *Centralized Exact Optimal* optimization when compared to the *Distributed Relaxed* heuristic. The *Bandwidth* optimization objective does not take into account the switching penalty explicitly, while the *Distributed Relaxed* optimization reuses local solutions which are still feasible to calculate the next optimal solution, inherently minimizing the difference between two subsequent solutions. For further discussion on how this switching penalty could be included during the optimization,



(a) Impact on average buffer starvations

(b) Impact on average played quality bitrate

(c) Impact on average number of quality switches

Fig. 6: Impact of the *RTT (ms)*, using a topology with $k = 5$, $BW_{l-1} = 3Mbps$, $BF = 0.8$, $|\mathcal{C}| = 125$ and $l = log_k|\mathcal{C}| + 1 = 4$

we refer to Section V-F.

The aforementioned results confirm the advantages of adopting in-network quality adaptation: First, the average played quality can be improved compared to the quality delivered by traditional client heuristics. Second, also the number of quality switches can be significantly reduced. Furthermore, the *Distributed Relaxed* heuristic is able to calculate a suboptimal configuration at low execution cost, making the approach viable for real-time delivery systems. The fast calculation of the optimal configuration also leads to fewer quality switches, since the configurations can be installed quickly and there is no need to install suboptimal temporary configurations as is the case with both *Centralized Exact Delayed* optimization.

E. Impact of Number of Bottlenecks

As indicated in Section IV-B, the number of bottlenecks has an impact on the behavior of the *Distributed Exact* optimization. The topologies for these experiments were created by introducing bottlenecks in a top-down manner and setting the bottleneck factor BF_l to 1 if there is no bottleneck at level l and $BF_l = 0.8$ otherwise. Figure 4 confirms a linear increase in execution time for both the *Exact* and *Relaxed* optimization with an increasing number of bottlenecks. The *Centralized Exact* optimization however, takes $300ms$ to execute, even in the absence of a bottleneck, while the *Distributed* optimization is only performed when the configuration assigning maximum quality to each client becomes infeasible, leading to an execution time of on average $20ms$, consisting solely out of the delay introduced by forwarding the local solutions.

F. Impact of Optimization Objective

An operator can optimize different policies when offering a HAS streaming service such as maximizing the total bitrate over all streams (Equation (8)), maximizing the proportional fairness across the streaming sessions (Equation (10)) or maximizing the QoE as a weighted sum of the total bitrate and bitrate variations (Equation (14)). Decreasing the factor α_s puts the emphasis on decreasing the impact of switches, while increasing α_s increases the impact of bitrate optimization. To quantitatively evaluate the fairness degree of the different optimization schemes, the Jain's Fairness Index is used [36]. This index states that if a system allocates resources to $|\mathcal{C}|$ users, where user c receives a rate allocation b_c, the fairness index of the system is defined to be:

$$J = \frac{(\sum_{c \in \mathcal{C}} b_c)^2}{|\mathcal{C}| \sum_{c \in \mathcal{C}} {b_c}^2} \tag{21}$$

Figure 5(a) shows the impact of these different optimization goals on the average Jain Fairness in function of the switching factor α_s. Optimizing the total bitrate allocation is able to achieve a fairness index closer to 1 then *AVC MSS*, indicating a fairer distribution of the available throughput among the clients. Adding rate shaping at the server allow increasing the fairness for *AVC MSS* at a slightly increased quality and a reduction with a factor 1.7 in terms of number of switches. When optimizing for proportional fairness, the in-network optimization is able to increase the fairness index at the cost of slightly decreasing the average quality as indicated in Figure 5(b) and 5(d) and increasing the average number of switches from 18 to 26. This indicates the trade-off between maximizing fairness and total bitrate allocation. When the impact of the number of switches increases (by decreasing α_s), the fairness index drops. This can be attributed to the fact that the in-network optimization will prefer to retain the same quality level for each client at any moment in time, leading to a number of clients downloading the highest quality, while newly arrived streaming sessions are assigned a lower quality, decreasing the total fairness. As Figure 5(c) shows, the in-network adaptation is able to reduce the number of quality switches compared to *AVC MSS* for all optimization goals. Explicitly minimizing the number of switches allows further reduction of d the number of switches from 43% to 30% at the cost of decreasing quality.

G. Impact of Delay

The in-network optimization uses an approximation of the achievable throughput as an upper bound for each link. Since RTT has the biggest impact on the available throughput, we varied the RTT and evaluated the impact on streaming performance. Figure 6(b) show how both the client-side and

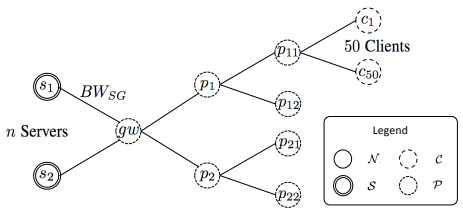
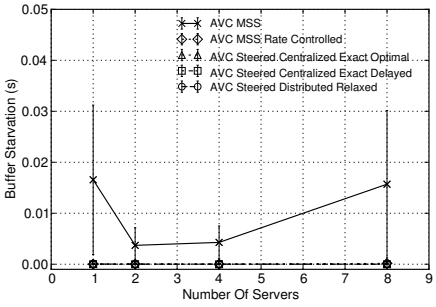
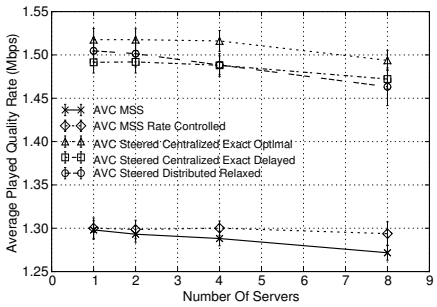


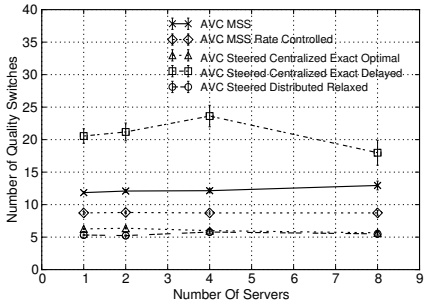
Fig. 7: Network topology, modeling a typical video service delivery network with multiple servers.



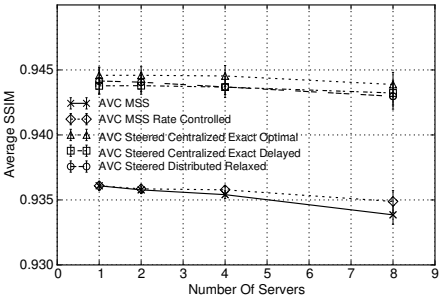
(a) Impact on average buffer starvations



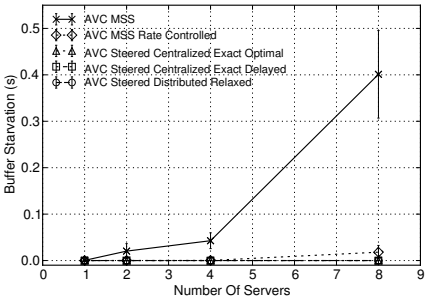
(b) Impact on average played quality bitrate



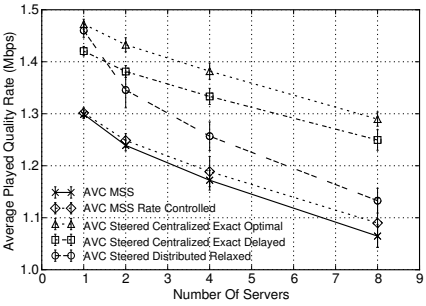
(c) Impact on average number of quality switches



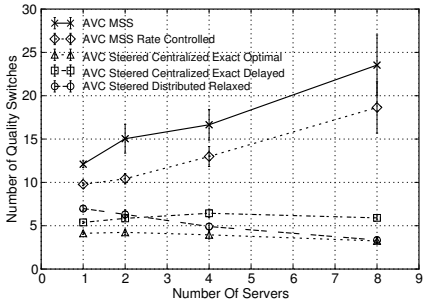
(d) Impact on average SSIM



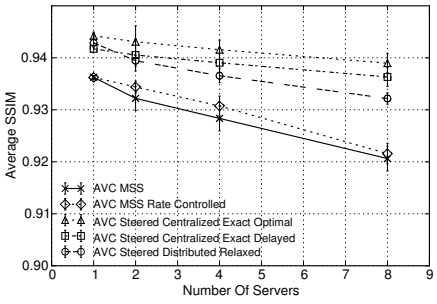
(e) Impact on average buffer starvations



(f) Impact on average played quality bitrate



(g) Impact on average number of quality switches



(h) Impact on average SSIM

Fig. 8: Impact of multiple servers with balanced load and the *Number of clients*, using a topology with $BW_{l-1} = 2Mbps$, $BF = 0.8$ and $|\mathcal{C}| = 200$ ((a), (b), (c)) and (d) and Impact of multiple servers with unbalanced load and the *Number of clients*, using a topology with $BW_{l-1} = 2Mbps$, $BF = 0.8$ and $|\mathcal{C}| = 200$ ((e), (f), (g) and (h))

in-network assisted quality decision based clients suffer from a quality degradation when delay increases. For low RTT (10 ms) we are able to increase the quality by 10.8% by deploying in-network quality decisions, while for higher RTT's (80ms), the gain decreases to 8.9%. This slight performance decrease can be attributed to the fact that an approximation of the achievable throughput is used. As discussed in previous work [12], HAS quality decreases quickly when RTT's increase due to the subsequent download-request cycles. Using HTTP pipelining can reduce the negative impact of the RTT on quality, by eliminating the idle time between two successive downloads. Figure 6(a) illustrates the impact of RTT on the decision times. Since the *Distributed* optimization requires a bottom up propagation of intermediary solutions, the decision times are also impacted by increasing delays. Figure 6(b) shows that for increasing delay, the performance of the in-network decisions slightly decreases when compared to the optimal decision, due to the network delay increasing the decision time.

H. Impact of Multiple Servers

To analyze the impact of multiple servers on the in-network optimization, we modeled a typical video service delivery network with multiple servers as illustrated in Figure 7. We modified the *Distributed optimization* to also support this type of topologies by first performing two types of bottom-up optimizations, one taking no limitations as input and a second optimization taking into account the limitations of

performing a local optimization between *Gateway* and servers first. Both approaches yield feasible solutions, out of which the most optimal one is selected. We varied the number of servers n and scaled the link bandwidths (BW_{SG}) to reflect this situation. We created 20 content items and distributed the clients over these content items using a Zipf distribution with $\alpha = 0.81$ to mimic the Internet popularity [37]. The content items were then assigned to the different server instances to evenly distribute the load among them. Figure 8(a) illustrates the average buffer starvation in seconds, showing how the in-network optimization is able to deliver the video stream without buffer starvations, whereas *AVC MSS* suffers some minor frame freezes due to competing behavior. The average quality is shown in Figure 8(b) and 8(d) which indicates that the in-network optimization is able to yield a higher average quality than *AVC MSS*. Adding rate shaping at the server, allows increasing the quality for *AVC MSS* when the number of servers increases. Up to 4 servers, the *Distributed Relaxed* optimization is able to outperform the *Centralized Exact Delayed* optimization due to the installation delay of the former approach, which was discussed earlier. The *Distributed Relaxed* decision however is suboptimal compared to the *Centralized Exact Optimal* decision. This is caused by the rounding heuristic and the distributed approach, which does not propagate multiple identical solutions which could yield more optimal solutions upwards the stream. Propagating multiple similar solutions, could benefit the distributed approach, at the cost of increased complexity and network communication. With respect to the number of switches, the *Distributed Relaxed* approach outperforms the *AVC MSS* algorithm, *AVC MSS* with rate shaping and *Centralized Delayed* optimization with approximately a factor between $[2.1, 2.3]$, $[1.5, 1.7]$ and $[3.3, 4.1]$ respectively.

During the previous experiments the load was evenly distributed across the different servers. During the following experiments, the content items were assigned to the different server instances following the pareto principle: 80% of the content items are stored on 20% of the servers. This results in an unevenly distributed request pattern across the servers. The *Distributed approach* is not able to achieve the same performance as the *Centralized optimization*. This can be attributed to the fact that an optimal resource allocation upstream of the gateway does not necessarily translate into a feasible and optimal solution downstream of the gateway and vice versa. Figure 8(f) shows the impact of adding multiple servers with uneven load distribution on the average achieved quality rate. This shows how the performance of the *Distributed optimization* degrades when the number of servers increases due to the suboptimal quality decisions. For 8 servers, the performance of the *Distributed optimization* drops to about 88% of the optimal solution. Figure 8(e) and (g) show the impact of increasing the number of servers on the buffer starvations and quality switches respectively. Both in-network approaches are able to maintain a fluent playout, while for *AVC MSS*, the average freeze time is about $0.4s$. For 8 servers, the average number of switches for *AVC MSS* amounts to 23, which can be reduced to 19 when applying server-based rate shaping. Enabling the *Centralized* in-network approach allows reducing the number of switches to 6. The *Distributed* approach even further decreases the average number of switches to 3.3, but at the cost of reduced quality compared to the *Centralized* optimization, as was mentioned before. This shows that an unbalanced spread across the content servers impacts the performance of the *Distributed* approach compared to the *Centralized optimization*.

VI. Conclusion

In this paper, we proposed an in-network QoE management for VoD HTTP Adaptive Streaming in a managed network environment. The adaptation algorithms enable the network providers to control the quality selection at the client. This allows them to increase the average played quality with at least 14% compared to traditional client-based heuristics. Moreover, due to the in-network management, the number of quality oscillations can be reduced with a factor 5 and with a factor 2.5 when traditional client-based approaches are combined with server-based rate shaping. We also discussed different variants of the in-network QoE management: an optimal Centralized ILP, a Distributed ILP and a relaxation of the Distributed algorithm. The impact of the number of clients, the Absolute Gap for the integer optimization and the number of bottlenecks were quantified. The impact of the delayed installation of the configurations showed that, even though the *Distributed Relaxed* optimization yields suboptimal configurations, the immediate installation of these configurations allows them to yield higher average quality at a significantly lower number of switches compared to the *Exact* optimization algorithms.

Acknowledgments

Niels Bouten is funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT).The research was performed partially within the iMinds MISTRAL project (under grant agreement no. 10838). This work was partly funded by Flamingo, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme. This work was carried out using the Stevin Supercomputer Infrastructure at Ghent University, funded by Ghent University, the Hercules Foundation and the Flemish Government department EWI.

References

[1] X. Wang, "Network-Assistance and Server Management in Adaptive Streaming on the Internet," in *Proceedings of the Fourth W3C Web and TV Workshop*, 2014.

[2] N. Bouten, S. Latré, W. Van de Meerssche, B. De Vleeschauwer, K. De Schepper, W. Van Leekwijck, and F. De Turck, "A Multicast-Enabled Delivery Framework for QoE Assurance of Over-The-Top Services in Multimedia Access Networks," *Journal of Network and Systems Management*, vol. 21, no. 4, pp. 677–706, 2013.

[3] S. Akhshabi, A. C. Begen, and 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.

[4] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and 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.

[5] D. C. Robinson, Y. Jutras, and V. Craciun, "Subjective Video Quality Assessment of HTTP Adaptive Streaming Technologies," *Bell Labs Technical Journal*, vol. 16, no. 4, pp. 5–23, 2012.

[6] S. Benno, J. O. Esteban, and I. Rimac, "Adaptive streaming: The network HAS to help," *Bell Labs Technical Journal*, vol. 16, no. 2, pp. 101–114, 2011.

[7] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and 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.

[8] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP – Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems (MMSys)*, 2011, pp. 133–144.

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

[10] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality Selection for Dynamic Adaptive Streaming over HTTP with Scalable Video Coding," in *Proceedings of the 3rd Multimedia Systems Conference (MMSys)*, 2012, pp. 149–154.

[11] J. Famaey, S. Latré, N. Bouten, W. Van de Meerssche, B. De Vleeschauwer, W. Van Leekwijck, and 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.

[12] N. Bouten, S. Latré, J. Famaey, F. De Turck, and 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.

[13] N. Bouten, S. Latré, W. Van de Meerssche, K. De Schepper, B. De Vleeschauwer, W. Van Leekwijck, and 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.

[14] T. Schierl, C. Hellge, S. Mirta, K. Grneberg, and 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.

[15] S. Latré and F. De Turck, "Joint In-network Video Rate Adaptation and Measurement-Based Admission Control: Algorithm Design and Evaluation," *Journal of Network and Systems Management*, vol. 21, no. 4, pp. 588–622, 2013.

[16] Y.-M. Hsiao, S.-W. Yeh, J.-S. Chen, and 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.

[17] W. Pu, Z. Zou, and C. W. Chen, "Video adaptation proxy for wireless Dynamic Adaptive Streaming over HTTP," in *Proceedings of the International Packet Video Workshop (PV)*, 2012, pp. 65–70.

[18] T. Wirth, Y. Sánchez, B. Holfeld, and T. Schierl, "Advanced downlink lte radio resource management for http-streaming," in *Proceedings of the ACM International Conference on Multimedia (MM)*, 2012, pp. 1037–1040.

[19] S. Parakh and A. Jagannatham, "Game theory based dynamic bit-rate adaptation for H.264 scalable video transmission in 4G wireless systems," in *Proceedings of the International Conference on Signal Processing and Communications (SPCOM)*, 2012, pp. 1–5.

[20] T. Sutinen, J. Vehkaperä, E. Piri, and M. Uitto, "Towards ubiquitous video services through scalable video coding and cross-layer optimization," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, 2012.

[21] H.-C. Lee and S.-M. Guu, "On the Optimal Three-tier Multimedia Streaming Services," *Fuzzy Optimization and Decision Making*, vol. 2, no. 1, pp. 31–39, 2003.

[22] C. Hsu and M. Hefeeda, "Optimal bit allocation for fine-grained scalable video sequences in distributed streaming environments," in *Proceedings of ACM/SPIE Multimedia Computing and Networking Conference (MMCN)*, 2007, pp. 650 402–650 402.

[23] M. Hefeeda and C.-H. Hsu, "Rate-distortion Optimized Streaming of Fine-grained Scalable Video Sequences," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 4, no. 1, pp. 2:1–2:28, 2008.

[24] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and 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.

[25] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. Vleeschauwer, W. Leekwijck, and F. 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.

[26] J. Padhye, V. Firoiu, D. Towsley, and 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.

[27] E. Altman, D. Barman, B. Tuffin, and M. Vojnovic, "Parallel TCP Sockets: Simple Model, Throughput and Validation," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–12.

[28] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, 1997.

[29] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[30] W. Li, S. Wang, Y. Cui, X. Cheng, R. Xin, M. Al-Rodhaan, and A. Al-Dhelaan, "AP Association for Proportional Fairness in Multirate WLANs," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 191–202, 2014.

[31] T. Kupka, "On the HTTP Segment Streaming Potentials and Performance Improvements." Ph.D. dissertation, University of Oslo, 2013.

[32] L. Plissonneau and E. Biersack, "A Longitudinal View of HTTP Video Streaming Performance," in *Proceedings of the Multimedia Systems Conference (MMSys)*, 2012, pp. 203–214.

[33] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[34] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 121–132, 2004.

[35] M. Hazewinkel, "Confidence Estimation," in *Encyclopedia of Mathematics*. Springer, 2001.

[36] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," DEC-TR-301, Digital Equipment Corporation, Tech. Rep., 1984.

[37] L. A. Adamic and B. A. Huberman, "Zipf's law and the Internet," *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

Niels Bouten obtained his M.Sc. degree in computer science in June 2011 from Ghent University, Belgium. In August 2011, he joined the Department of Information Technology at Ghent University, where he is currently active as a Ph.D. student. His main research interests are the application of autonomic network management approaches in multimedia delivery. He is involved in FP7 NoE Flamingo.

Steven Latré is a fulltime professor affiliated with the Department of Mathematics and Computer Science of the University of Antwerp and the iMinds Future Internet Department. His main research interests are the use of autonomic network management approaches with a special focus on Quality of Experience optimization and federations. He is in the program committee of several conferences and regular reviewer for conferences and journals. He is, and has been, involved in several European projects, including FP7 IP Muse, CELTIC RUBENS, FP7 STREP ECODE, FP7 STREP OCEAN and FP7 NoE Flamingo.

Jeroen Famaey is post-doctoral researcher at the Department of Mathematics and Computer Science, University of Antwerp and affiliated as a senior researcher with the iMinds Future Internet department. He received his M.Sc. degree in Computer Science from Ghent University in 2007 and a Ph.D. in Computer Science Engineering, on federated management of multimedia streaming services, from the same university in 2012. His research interests include multimedia streaming services, federated and inter-domain network management, network and cloud resource management, and applied optimization theory and machine learning.

Werner Van Leekwijck obtained a Master of Science degree in Electrical Engineering from the University of Leuven in 1990, and an additional Master degree in Knowledge and Information Technology from the University of Ghent. He joined Alcatel in 1990 and he has held various technical and management positions in research and product development in the area of IP multimedia technology and networks. Currently, he is a senior research manager in Bell Labs in Antwerp, and is responsible for future multimedia delivery architectures.

Filip De Turck is a full-time professor affiliated with the Department of Information Technology of Ghent University and the iMinds Future Internet Department. Filip De Turck is author or co-author of approximately 340 papers published in international journals or in the proceedings of international conferences. His main research interests include scalable software architectures for telecommunication network and service management, performance optimization and evaluation and design of new telecommunication services.