A model for evaluating sharing policies for network-assisted HTTP adaptive streaming

Kleinrouweler, Jan Willem; Cabrero, Sergio; van der Mei, Rob; Cesar, Pablo

DOI
10.1016/j.comnet.2016.03.023

Publication date
2016

Document Version
Accepted author manuscript

Published in
Computer Networks

A Model for Evaluating Sharing Policies for Network-assisted HTTP Adaptive Streaming

Jan Willem Kleinrouweler[1], Sergio Cabrero[1], Rob van der Mei[1,2], and Pablo Cesar[1,3]

[1]Centrum Wiskunde & Informatica, Science Park 123, 1089 XG Amsterdam, The Netherlands
[2]VU University, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands
[3]Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

{j.w.m.kleinrouweler, s.cabrero, r.d.van.der.mei, p.s.cesar}@cwi.nl

Abstract

HTTP adaptive streaming (HAS) has become the dominant technology for streaming video over the Internet. It gained popularity because of its ability to adapt the video quality to the current network conditions and other appealing properties such as usage of off-the-shelf HTTP servers and easy firewall traversal. However, when multiple HAS players share a bottleneck link for streaming, the individual adaptation techniques in the players have difficulties to maintain a stable bitrate and fairly share the network resources. HAS-assisting network elements can solve these performance problems and allow execution of advanced policies for sharing the available bandwidth. Nonetheless, testing and evaluating new sharing policies is costly and time consuming. This motivated us to formulate a model that allows to differentiate between groups of users depending on the type of user or device, and that can describe the mean bitrate of the video streams and how often this bitrate is expected to change during playout. To show how our model can be used, we demonstrate two applications of our model. Furthermore, we validate the model based results against results obtained using our streaming testbed and proxy server based HAS-assistant. The results show that our model is highly accurate for both the mean bitrate and the number of changes in video bitrate. As such, our model is a useful tool for network administrators and internet service providers for evaluating the performance of sharing policies and for managing and provisioning video delivery networks.

Keywords

HTTP adaptive streaming; Video streaming; Performance; Capacity sharing; Markov model; Bitrate; Fairness; Instability

1 Introduction

HTTP adaptive streaming (HAS) has become the major technology for streaming over the Internet. In HAS, a video file is split up into segments typically with a duration between two and ten seconds. Each segment is encoded at multiple bitrates and resolutions. All video segments are placed on an HTTP server together with a manifest file. This manifest file describes the index, URL, bitrate and resolution of each segment. When a video player starts a stream, it first downloads the manifest file and then downloads the video segments in a bitrate or resolution that it sees fit. The major advantage of this technology is that it allows video players to adapt the video quality to the current network condition. Furthermore, because HTTP adaptive streaming is based on known Web technology, namely HTTP, content providers can leverage existing methods in distribution such as content delivery networks (CDNs) and caching. Moreover, the usage of HTTP tackles the issues with firewall and NAT traversal.

The reasoning behind HAS is that the video quality can be matched to both the available bandwidth and the type of device. On the one hand, this means that in situations where the available bandwidth becomes lower, buffer under-runs can largely be avoided by (temporarily) lowering the video quality. When more bandwidth becomes available the player can adapt the stream to a higher video quality to optimize the streaming experience for the user. On the other hand, since the intelligence is located at the player (i.e. the player selects the bitrate and resolution of the stream) it can take into account device capabilities, battery level, and data usage. This approach has its advantages over non-adaptive streaming as it is more robust in networks with unstable performance. However, it was found by several studies that the adaptation mechanism in the players suffer performance problems when multiple players share a bottleneck link [3, 2, 10]. The two most relevant problems are unfair sharing of the available bandwidth and instability. Instability refers to too often changing the video quality and it is identified to negatively impact the video watching experience [4, 7, 17].

HTTP adaptive streaming players adapt the video quality based on estimations of the available bandwidth. Most players use the download speeds of the previous segments as a measure for the bitrate of future video segments. However, partially due to the bursty nature of HAS traffic, it is difficult for the players to make accurate bandwidth estimations [10, 5]. More sophisticated adaptation algorithms with better heuristics and conservative switching between video profiles can lower the number unnecessary quality switches and improve fairness [12, 20, 11, 14, 15]. However, fixing the problems only in the player remains difficult because players have a limited view on what occurs in

the network. In the case where several video players share a bottleneck link, for example at home or on a hotel Wi-Fi network, the players are unaware of each other and thus cannot use this information while selecting a video bitrate. This eventually leads to a suboptimal distribution of the available bandwidth.

As an alternative to improving adaptation algorithms, several implementations have been proposed that use knowledge from devices in the network to assist video players in selecting the video quality. These implementations range from traffic shaping at the residential gateway [9], signaling players from a measurement proxy [16], using OpenFlow [6], and our implementation in the form of an HTTP proxy server [13]. Typically, these implementations target networks where the bottleneck link is in the local- or access network, and thus relatively close to the users. Network devices close to the players, such as home gateways or switches in the access network, have a good view of the traffic on the bottleneck link. They can share this view with the video players.

The solutions that solicit in-network devices for making adaptation decisions show promising results. For example, in previous work we showed that the number of changes in video quality can be reduced while improving the fairness between video streams [13]. However, the sharing policy did not take into account the types of streams or devices, and thus only represented fairness on a bitrate level instead of targeting an equal quality of experience while considering device specific factors. Fortunately, this can be resolved by improving the sharing policy. If an in-network device has an overview of both the streams and the device specific factors, then it becomes the most convenient point to make the adaptation decisions.

Changing the capacity sharing policy affects the streams' bitrates, and how often this bitrate will change during the playback of the video. In order to gain insights on the performance of a policy under different circumstances, it has to be evaluated. However, building testbeds to determine the performance of a policy is costly and time consuming. In previous work, we proposed a model that allows to accurately estimate the bitrate of the video streams and the bitrate stability [13][1], and as such give an estimation of the quality of experience (QoE) of the viewer. The QoE of the viewer improves when the bitrate of the video increases, while the number of switches in bitrate should be kept low [18, 8].

The contribution of this paper is three-fold. First, we extend our model to include player prefetching to become more accurate. Second, we show how our model can be used by demonstration our model in the form of two example applications. Third, we show that our model-based results are highly accurate when comparing them to the actual performance in a testbed with our proxy server.

Although the number of deployments of network-assisted HAS in currently low, we expect it to become more common as a result of the standardization of the Server and Network-Assisted DASH (SAND) architecture [19]. However, SAND only specifies the communication between the HAS assistant and the

[1]Reference [13] presents an early version and evaluation of the model. In this paper we extend this model and use it to evaluate different sharing policies.

players. This means that it is still up to the HAS assistant to decide how the available bandwidth must be shared among the players. Our model can be used to quickly evaluate a sharing policies for network-assisted HAS. As such, it is possible to evaluate a large number of sharing policies prior to deployment of the HAS assistant, and select the optimal policy for each network. We present our model as a useful tool for network administrators and internet service providers (ISPs), and encourage them to use it when developing new sharing policies, as well as using the model for managing and provisioning a HAS based video delivery network.

The remaining of this paper is organized as follows. In Section 2 our HAS-assisting network element in the form of an HTTP proxy server is introduced, as well as our streaming testbed. Section 3 presents the performance model that describes the mean bitrate and number of bitrate changes. In Section 4 two types of sharing policies are evaluated and the model-based results are validated against results achieved using our streaming testbed. Section 5 concludes this paper.

2 Network-assisted HAS

The adaptation algorithms in HTTP adaptive streaming players are designed in such a way that they will provide the user with the highest possible video quality. This approach relies on best effort and adaptation decisions are made from the viewpoint of the player. This means that HAS players can be considered to be selfish, because players only try to maximize the bitrate for their stream and do not consider other traffic or other players. Instead of simply sharing the available bandwidth, HAS players have to compete with each other for their own share. As a result, the share of the bandwidth that is available to a player can vary. Variations in bandwidth available to a player cause changes in video bitrate, even though theoretically these changes should not have to occur when the available bandwidth is fairly divided among the players. A higher number of changes in video quality lowers the quality of experience of the viewer. Furthermore, since HAS players by default are not aware of each other, they cannot take into account other players and their characteristics when deciding on a bitrate. When competing for bandwidth each player is equal, where for instance it would have been better for devices with smaller screens to make room for devices with larger screens.

2.1 HAS proxy server

In network-assisted HAS, the problem described above is countered by including network devices that have a broader view of the use of the bottleneck network link. These so called HAS-assisting network elements are aware of the active streaming players through monitoring the network traffic. When players signal the network element with their requirements (these can both be minimum and maximum requirements) and characteristics, the HAS-assisting network element

can take these factors into account when dividing the available bandwidth. The major difference from regular HAS is that adaptation decisions are not made individually by the players, but by an overseeing network element while receiving support from the players.

We implemented a HAS-assisting network element in the form of an HTTP proxy server. The proxy server can be installed on routers and gateways that are relatively close to the players and the bottleneck link that has to be guarded. In practice this means that the proxy server approach can be applied to networks where the bottleneck is the local network, the household's or company's internet connection, or a link in the access network of the ISP. Figure 1 illustrates these three scenarios and indicates placement of the proxy server. For scalability reasons the proxy server should be placed as close to the players as possible, but such that it processes the video traffic from all players on the shared bottleneck link. For both scenarios (A) and (B) the proxy server would preferably be located at the (Wi-Fi) router, even though the bottleneck is situated at opposite sides of the proxy server. In scenario (C) the proxy server has to be placed further upstream in a neighborhood station or switch.

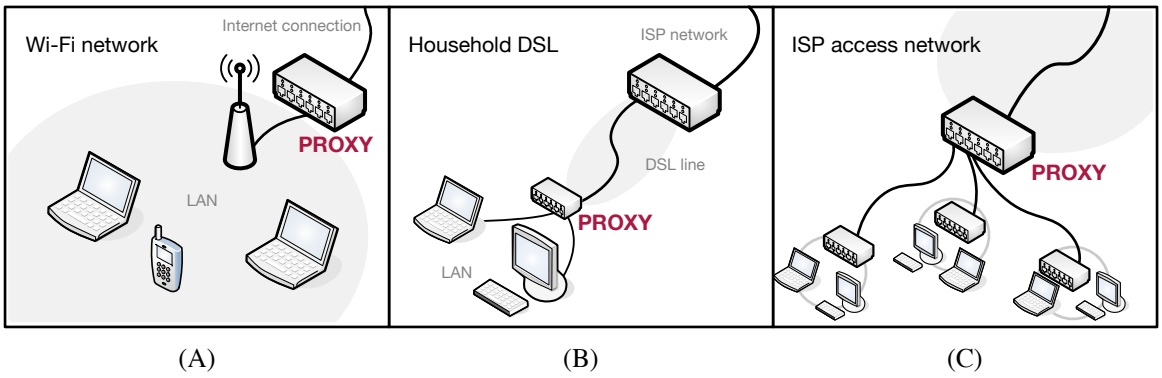

Figure 1: Use cases for HAS-assisting network elements and indication of proxy placement. Bottlenecks (marked in gray) are: (A) the Wi-Fi network, (B) the DSL line, and (C) the ISP access network.

All HTTP traffic, i.e. TCP traffic with destination port 80, is transparently forwarded to the proxy server[2]. HTTP traffic is monitored in order to detect starting and stopping video players. When starting a stream, a HAS video player first downloads the manifest file that contains the details of the stream, such as the available bitrates, resolutions and the URLs of the video segments. The proxy server will listen for manifest downloads that indicate starting players. Like the player, the proxy server will also process the manifest file. The proxy server uses the manifest to obtain characteristics of the stream that it uses when

[2]In theory it is also possible to proxy encrypted traffic that uses HTTPS, however for this to work the the user must trust the proxy server and the player should accept the proxy's certificate. This requires user interaction and/or configuration.

dividing the available bandwidth among the players. Furthermore, the proxy server can track the video players' activities based on the HTTP request for video segments. In general, a video player has stopped a stream when the last segment in the stream has been downloaded. However, since users oftentimes stop a stream before it is finished, the proxy server marks a player as stopped after a certain period of inactivity. To set a value for this timeout we make use of the periodic behavior of HAS players. In steady-state mode, segments are requested with intervals equal to the duration of the segment. If the download of a segment takes shorter than the duration of a segment, there will be a period of inactivity.

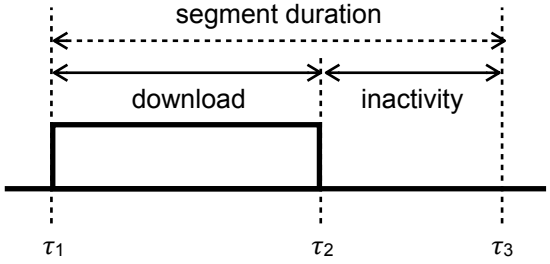

Figure 2: Period of download activity followed by a period of inactivity

As depicted in Figure 2 the start of the download of a segment is marked τ_1, and the segment download is finished at τ_2. If $T_{segment}$ is the segment duration, then the maximum period that a player can be inactive before requesting the next segment is:

$$I_{max} = \max(T_{segment} - (\tau_2 - \tau_1), 0) + 1. \quad (1)$$

The inactivity period is unlikely to be higher than the duration of the segment minus the download time. In case the download takes longer than the segment duration there will no period of inactivity. To be certain a player is finished and to cope with small variations in periodicity a margin of one second is added.

Detection of starting and stopping players happens through monitoring the traffic. By inspecting the `User-Agent` field in the HTTP header, the proxy server can obtain a rough estimate of the type of device. However, the HAS players have to communicate more accurate and detailed information to the proxy when more advanced sharing policies are considered. Players that are modified to work with the HAS assistant can do this through in-band signaling via additional fields in the HTTP header. This allows players to signal the proxy server before the start of the stream as part of the request for the manifest file, and during the stream in requests for the video segments. The proxy server can provide players with additional information via the HTTP response.

The proxy server uses the information on how many players are active, the characteristics of the streams and the types of devices and their capabilities to divide the bandwidth among the players. How the proxy server divides the available bandwidth depends on the active policy. Currently we have implemented

three policies in the proxy server. The first policy equally divides the available bandwidth among the players and for each player selects the highest bitrate that is lower or equal to the fair share. The second policy classifies devices based on screen size and resolution, and targets each device type accordingly. The third policy makes a distinction between regular and premium users and ensures higher quality video for premium users.

If the proxy server detects a player requesting a segment in a bitrate that is different from the bitrate that it selected for this player, it corrects the request by rewriting it into a request for the same segment but in the correct bitrate. When the proxy server performs a rewrite, it will add an additional field to the HTTP response informing the player about the rewrite. This allows the player to act accordingly in the decoding and rendering pipeline. In some occasions the forced rewriting of requests by the proxy server is unwanted. For instance, when a player is not able to stream at the selected bitrate due to other limitations in bandwidth on the path between server and client, or when buffer levels are critically low. In these cases a player can request the proxy server not to rewrite the request.

2.2 Streaming testbed

To evaluate the performance of the proxy server under a certain policy, we installed the proxy server in our streaming testbed. All devices in the testbed are implemented as lightweight virtual machines with their own process- and network stack. The capacity and delay of the connections between the devices is set by means of network emulation. The CORE network emulator [1] is used to configure the setup as depicted in Figure 3. The VMs run on a GNU Debian Linux 6 host that has a 2.83 Ghz Intel Core 2 Quad CPU and 8 GB ram.

The testbed consists of 20 virtual PCs, two routers and three servers. The bottleneck link is the network connection between the two routers. The capacity of this link is limited to 8 mbit/s to represent a network connection that is not sufficient for multiple players streaming at the high bitrates. The round-trip delay between the clients and each of the servers is set to 10, 20, and 40 ms respectively.

The router closest to the client machines (router-1) is made into a HAS-assisting network element by installing our proxy server. Although the bottleneck link has a capacity of 8 mbit/s, the proxy server is configured with a maximum channel capacity of 6.8 mbit/s, thus having a 15% safety margin. The safety margin is included to allow for lightweight background traffic and provide the video players extra capacity to maintain sufficient buffer levels. In the experimental runs in this paper there is no background traffic present.

Video players are started at free clients according to a Poisson process. Each client can hold a single instance of a video player, however the maximum number of active video players at the same time is limited to 17. In all tested scenarios the lowest available video bitrate is 400 kbit/s. To ensure sufficient bandwidth for uninterrupted streaming, an 18th player that would cause a too high demand on the network is denied service by the proxy server.

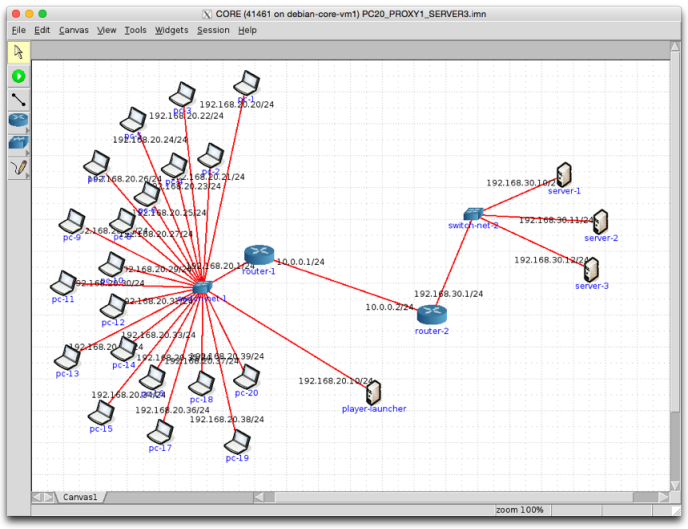

Figure 3: Streaming testbed using lightweight virtual machines and emulated network connections

At the client machines a stripped-down version of our custom HAS player is used. The supports signaling information about the device to the proxy server. Decoding and rendering of the video is disabled to reduce the CPU load and memory usage of the virtual machines. The players use an 8 second buffer, unless otherwise stated. The videos that are used in the evaluations are segmented with a duration of 4.0 seconds and described in a manifest file according to the HTTP Live Streaming (HLS)[3] format. The manifest files and the video segments are placed on the three off-the-shelf Apache 2.2.22 HTTP servers.

The URLs of the video segments are formatted such that they contain an identifier of the player, an identifier of the video, the index of the segment, and the requested video bitrate. For each evaluated setting, twelve hours of HTTP traces are collected using tcpdump. Through analysis of the HTTP traces we can obtain the mean bitrate of the streams and the number of times the bitrate is changed during playback.

3 Performance model

The key difference between adaptation algorithms in the player and the bandwidth division algorithm in the proxy server is that the proxy server bases it decisions on a flow level view instead of on the individual downloaded segments. Without the proxy server, players enter the bandwidth competition for every

[3]https://developer.apple.com/streaming/

segment, and the outcome of the competition can be different any time. For the proxy server we expect that the total capacity of the bottleneck network link is constant and that the available bandwidth is only re-divided among the players when a new stream starts or a current stream stops. An advantage of this is that the network usage becomes more predicable as it is predefined what happens when a player of a certain type starts or stops. This behavior can also be leveraged in model based performance analyses. At the core, it has to be described how many players of each type are active and what the streaming bitrate is for each of the players. Based on that, the mean bitrate of the streams can be retrieved. By observing how often new players arrive or current players stop, and thus observing how often the division of the available bandwidth changes, we can obtain the number of quality changes in a stream.

3.1 Starting and stopping players

One of the characteristics of a policy is that a policy can distinguish between different types of players. To keep the policies concise and easy to execute in the proxy server, devices are grouped based on their type. The idea behind grouping players is that players in the same group are treated equally by the policy, where players in different groups can be treated differently. This implies that all players in a group will get the same video bitrate assigned by the policy.

The process of starting and stopping players is captured in a Markov process. Let K be the number of different groups considered by the policy and n_k denote the number of active players is group k, then each state in the process is described by a vector $(n_1, n_2, \ldots, n_K)$. We assume that HAS-assisting network elements do not allow more video players than the network allows for. Although our proxy server was not initially intended for access control, it can take this role and prevent streaming interruptions caused by a too heavy demand on the network. This implies that the state space of the Markov process is finite. The state space $\mathcal{S}$ is defined as all states with non-negative integer valued entries $(n_1, n_2, \ldots, n_K)$ that satisfy the following condition:

$$\sum_{k=1}^{K} n_k \tilde{B_k} \leq C, \tag{2}$$

where $\tilde{B_k}$ is the lowest available bitrate for players in group k, and C is the capacity of the bottleneck link.

Transitions between states are linked to the arrivals of new players and the termination of active players. We assume that players of group k arrive according to a Poisson process with intensity λ_k. In HAS video streaming, the download of video segments has to keep up with the playback. Therefore, the video segments are typically chosen such that the time to download a segment is equal to (or slightly shorter) than the duration of that video segment. Adapting the video bitrate means that the job size (the number of bytes in the video stream) changes accordingly to the load on the network, and that the time that video players are active in the network is tightly linked to the duration of the

video. Therefore, the rate for transitions $n_k \to (n_k - 1)$ is n_k/β_k, where β_k is the mean duration of videos in group k. The Markov process at the base of our model is equal to the Erlang multi-rate loss model, for which it is well-known that a stationary distribution exists with the following product form solution:

$$\pi(n_1, n_2, \ldots, n_k) = \frac{1}{G} \prod_{k=1}^{K} \frac{(\lambda_k \beta_k)^{n_k}}{n_k!}, \tag{3}$$

where G is the normalization factor:

$$G = \sum_{\underline{x} \in \mathcal{S}} \prod_{k=1}^{K} \frac{(\lambda_k \beta_k)^{n_k(\underline{x})}}{n_k(\underline{x})!}. \tag{4}$$

The Erlang multi-rate model has the advantage of being insensitive to the distribution of service times. In our model the service time refers to the durations of the video streams β_k. A summary of the notation for our model can be found in Table 1.

Table 1: Model notation

Notation	**Description**
Input	
C	Capacity of the network connection
λ_k	Rate of the Poisson process at which group-k users start the video streams
β_k	Mean duration of the video streams for group k
B_k	Available bitrates for video streams for group k
$T_{segment_k}$	Segment size used in the video streams for group k
Intermediate	
$\mathcal{S}$	State space of the Markov process
$\pi(\underline{x})$	The probability that the Markov process is in state $\underline{x}$
$n_k(\underline{x})$	The number of group-k players in state $\underline{x}$
$q_k(\underline{x})$	The bitrate of group-k players in state $\underline{x}$
$\gamma(\underline{x} \to \underline{y})$	The number of group-k players that change video bitrate when transitioning from state $\underline{x}$ to state $\underline{y}$
Output	
$\mathbb{E}[N_k]$	The expected number of players of group k
$\mathbb{E}[Q_k]$	The expected number of bitrate switches for players of group k
$\mathbb{E}[B_k]$	The expected bitrate for players of group k

3.2 Streaming bitrate and bitrate switches

The bitrate of a video stream, and how often this bitrate changes, depends on how the policy divides the bitrates among the players. From an abstract

level, a policy is a function that takes the capacity of the network, the number of players in each group, and the available bitrates for each player as input, and outputs for each group k a video bitrate q_k while taking into account the streams, devices and users. Because in each state $\underline{x} \in \mathcal{S}$ the number of players of each group is different, the policy has to be computed for all states in $\mathcal{S}$. We use the notation $n_k(\underline{x})$ to denote the number of players of group k in state $\underline{x}$, and $q_k(\underline{x})$ to denote the bitrate for players of group k in state $\underline{x}$.

Given the number of players and the bitrates selected by the policy the mean bitrate can be straightforwardly obtained. If the mean number of players in group k is defined as:

$$\mathbb{E}[N_k] = \sum_{\underline{x} \in \mathcal{S}} \pi(\underline{x}) n_k(\underline{x}), \tag{5}$$

then the mean bitrate for the streams in group k becomes:

$$\mathbb{E}[B_k] = \frac{1}{\mathbb{E}[N_k]} \sum_{\underline{x} \in \mathcal{S}} \pi(\underline{x}) n_k(\underline{x}) q_k(\underline{x}). \tag{6}$$

The number of quality switches relates to how often the Markov process transitions between states. If the selected bitrate for a group of players is different between two states, then a bitrate switch is potentially made when the process transitions between those states. The intuition behind determining the number of bitrate switches is that by observing the number of transitions between states with different selected bitrates, we can obtain the number of switches in video quality. However, HAS players can technically only switch in between segments, and not during the download of a segment. The reason for this is that requested video quality is part of the HTTP request, and only when making a new request a new bitrate can be selected. Therefore, to include this behavior of the HAS player in the model, we observe the Markov process with intervals equal to the segment duration. The probabilities that the process transitions from state $\underline{x}$ to state $\underline{y}$ in $T_{segment}$ seconds can be retrieved via uniformization of the continuous time Markov chain, by conditioning on m. If $P^m_{\underline{x},\underline{y}}$ is the probability that the Markov process transitions from state $\underline{x}$ tot state $\underline{y}$ in m steps, and if b is the uniform rate parameter, then the probability that a transition $\underline{x} \to \underline{y}$ occurs in $T_{segment}$ seconds becomes:

$$P_{\underline{x},\underline{y}} = e^{-bT_{segment}} \sum_{m=0}^{\infty} \frac{(bT_{segment})^m}{m!} P^m_{\underline{x},\underline{y}} \qquad \text{for } \underline{x}, \underline{y} \in \mathcal{S}. \tag{7}$$

The duration of the videos is variable, therefore we express the number of switches in video quality not as an absolute number but as a rate: number of bitrate switches per second. The expected bitrate instability rate is defined as:

$$\mathbb{E}[Q_k] = \frac{1}{T_{segment}\mathbb{E}[N_k]} \sum_{\underline{x},\underline{y} \in \mathcal{S}} \pi(\underline{x}) P_{\underline{x},\underline{y}} \gamma_k(\underline{x} \to \underline{y}), \tag{8}$$

where $\gamma_k(\underline{x} \to \underline{y})$ is the number of players in group k that make a bitrate switch on the transition $\underline{x} \to \underline{y}$. Note that the bitrate instability rate is defined from the viewpoint of single player. Players in group k only make a bitrate switch when the bitrate in state $\underline{x}$ is different from the assigned bitrate in state $\underline{y}$. Furthermore, the number of players that make a switch is limited to the players that are both active in $\underline{x}$ and $\underline{y}$. A player that is started will already stream at the selected bitrate and does not have to make a switch. Similarly, a player that terminated a stream cannot make bitrate switches anymore. The number of players that make a bitrate switch on the transition $\underline{x} \to \underline{y}$ then becomes:

$$\gamma_k(\underline{x} \to \underline{y}) = \begin{cases} 0 & \text{if } q_k(\underline{x}) = q_k(\underline{y}), \\ \min(n_k(\underline{x}), n_k(\underline{y})) & \text{if } q_k(\underline{x}) \neq q_k(\underline{y}) \end{cases} \tag{9}$$

Equations 5, 6 and 8 are defined per group k to allow for a more detailed evaluations. The overall mean number of players, mean bitrate, and expected bitrate instability rates can be found via a weighted average, weighted by the mean number of players for each group:

$$\mathbb{E}[N/B/Q] = \frac{\sum_{k=1}^{K} \mathbb{E}[N_k] \cdot \mathbb{E}[N_k/B_k/Q_k]}{\sum_{k=1}^{K} \mathbb{E}[N_k]}. \tag{10}$$

3.3 Inclusion of player prefetching

The model presented above describes the steady-state behavior of HTTP adaptive streaming players. In practice, HAS players first enter the prefetching phase before going into the steady-state phase. This behavior does effect the streaming bitrate, but is not yet included in the model described above. In this section we describe how we can improve the accuracy of our model by taking player prefetching into account.

During the prefetching phase, video segments are requested immediately after the finishing downloading the previous segment, i.e. without the period of inactivity after the segment download. A buffer is maintained during streaming to prevent interruptions in playback caused by small variations in the available network capacity. Depending on the type of stream this buffer size can vary. For live streams the buffer size is kept small because it is important that the play-out point is close to the actual broadcast. For video-on-demand (VoD) this timing requirement can be relaxed and larger buffers with better resilience agains video interruptions are more common.

The size of the buffer in the player has an effect on the mean bitrate of the videos when our HAS proxy is used. This effect is illustrated in Figure 4, where three players with different buffer sizes are compared. PlayerA uses a buffer of two video segments or 8 seconds. PlayerB and PlayerC have buffer sizes of 16 seconds and 24 seconds respectively. All players are evaluated in our streaming

testbed and stream the same video with a duration of 144.0 seconds encoded at 400, 720, 1020, 2300 and 4200 kbit/s. Figure 4 also includes the model based results for comparability.

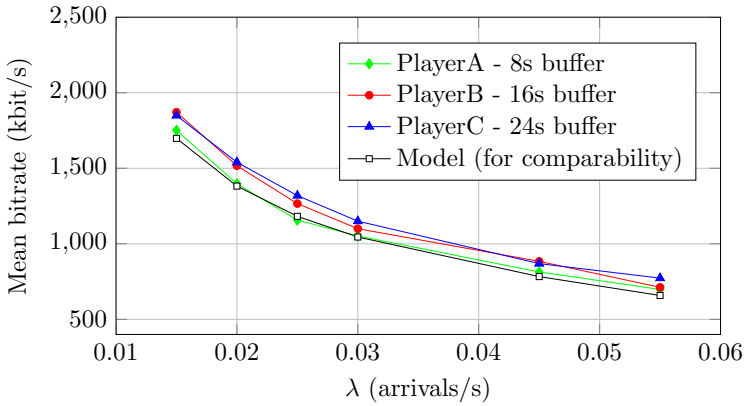

Figure 4: The effect of the players' buffer sizes on the mean bitrate of HAS streams. Larger buffers result in a higher mean bitrate.

The results show that the mean bitrate increases when players with a larger video buffer are used. The reason for this is that players with larger buffers are shorter active in the network. During prefetching, the time that players are active in the network is less than the duration of video that is downloaded during that time. In the steady-state phase, the network activity equals the duration of downloaded video. The difference between not including and including player prefetching is illustrated in Figure 5.

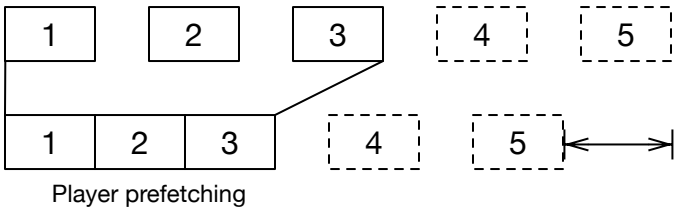

Figure 5: Prefetching causes the player to be active in the network for a shorter period of time. No prefetching (top) versus prefetching (bottom).

This also explains why our model is accurate for players with small buffers, but shows an underestimation of the mean video bitrate for players with larger buffers. To account for the prefetching behavior in the model, it has to be determined how much time is spent in the buffering phase, and how much time is spent in the steady-state phase. An estimation of the mean time that it takes

to download a single video segment can be found as:

$$T_{download} = \frac{\mathbb{E}[B] \cdot T_{segment}}{C/\mathbb{E}[N]} = \frac{\sum_{x \in \mathcal{S}} \pi(\underline{x}) n(\underline{x}) q(\underline{x}) \cdot T_{segment}}{C}. \tag{11}$$

Based on the time that it takes to download a single segment, it can be estimated how many segments need to be downloaded to reach a certain buffer level.

A common buffer strategy in HAS players is as follows. The player starts by downloading one segment of video. Then, it starts playback while prefetching (i.e. requesting video segments without inactivity period in between segment downloads) until the player reaches a certain buffer level. During prefetching, the inflow into the buffer is $T_{segment}$ seconds, and the outflow from the buffer is $T_{download}$ seconds. To reach a certain buffer level *Buff* so that the player can go into steady-state mode, $\lceil \alpha + 1 \rceil$ segments have to be downloaded:

$$\begin{aligned} \mathit{Buff} - \mathrm{T}_{\mathrm{segment}} &= (T_{segment} - T_{download})\alpha \\ \alpha &= \frac{\mathit{Buff} - \mathrm{T}_{\mathrm{segment}}}{T_{segment} - T_{download}} \end{aligned} \tag{12}$$

The download of the first video segment – while there is no outflow from the buffer – is accounted for by subtracting $T_{segment}$ from the total buffer level in Equation 12, and increasing α by one to come to the total number of video segments that is required to be downloaded to reach *Buff*. The number of segments is rounded up, $\lceil \alpha + 1 \rceil$, because moving from prefetching to steady-state phase can only occur in between segments, but not during segment downloads.

The effective service time β_{eff} that video players are active in the network, given a certain video length β_{video} and $\beta = \beta_{video}$, then becomes:

$$\beta_{\mathit{eff}} = \lceil \alpha + 1 \rceil (T_{download} - T_{segment}) + \beta_{video}. \tag{13}$$

The effective service time β_{eff} is lower than the actual service time β that is used in the model. Therefore, to obtain a more accurate mean bitrate, β has to be lowered to match β_{eff}. However, $T_{download}$ and α are dependent on β and lowering β will thus affect β_{eff}. The intersection $\beta = \beta_{\mathit{eff}}$ is found by iteratively lowering β while keeping β_{video} constant. A comparison of the model based mean bitrate, the corrected model based mean bitrate, and the actual mean bitrate of players with a 24 second buffer (PlayerC) is displayed in Figure 6. The results show that including prefetching info the model results in better accuracy when players with larger buffers are used, making it more broadly applicable.

4 Capacity sharing policies

At the proxy server the bandwidth that is available for video streaming is divided among the players, according to a policy. In this section we perform a model-based evaluation of two example sharing policies. These examples are

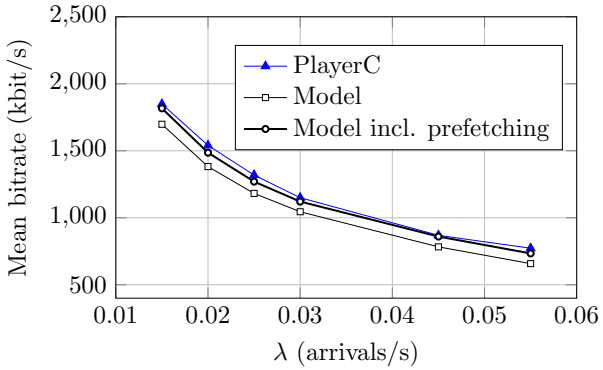

Figure 6: Comparison of the model-based mean bitrate, corrected model-based mean bitrate, and the mean bitrate achieved in experiments. The model including prefetching shows higher accuracy for the player with a large buffer.

to show sensitivity of our model to changes in the sharing policy, as well as to demonstrate how the model can be applied. The first example compares a policy that takes all devices as equal, to a policy that takes into account the screen size and resolution of the devices. The second example compares two policies that include priority or premium users. In addition to demonstrating how our model can be used for policy evaluations, the model-based results are validated by comparing them against results that are obtained using our streaming testbed.

4.1 Example: Device heterogeneity

Mobile devices have become more powerful and are fully enabled for streaming videos using their Internet connection. It is not uncommon that smartphone and tablet devices are used for video streaming. Traditional devices, like television sets, nowadays also come with a network connection and they offer the same streaming services that are available on the PC. Together these devices create an interesting mix of different screen sizes and potentially different usage patterns. Because of the different screen sizes it is not fair to equally divide the available bandwidth among the players, since this would not yield an equal quality of experience.

Georgopoulos et al. describe how different bitrates and resolutions can be compared among devices with different form factors [6]. In our examples we will use the same groups of devices, video profiles and video quality mapping. The first group is smartphone sized devices that stream a 360p video of 60 seconds, encoded at 400, 600 and 1000 kbit/s. The second group represents tablet viewers that stream a 720p video of 120 seconds, encoded at 400, 600, 1000, 1500, and 2000 kbit/s. The third group is large screen devices that stream a 1080p video of 180 seconds, encoded at 400, 600, 1000, 1500, 2000, 4000 kbit/s.

Each player will report its screen size to the proxy server, via the signaling

mechanism. This way, the proxy server can take different device types into account. Based on the screen resolution and the available bitrates, a device-aware quality mapping is created and listed in Table 2. Depending on the number of players with each resolution, the proxy server selects a quality level from Table 2 that fits the capacity of the channel. For example, the test if the current active players would fit the capacity of the network given quality level 2 would be:

$$\#360p * 600 + \#720p * 1500 + \#1080p * 2000 \leq C \tag{14}$$

For quality levels 1-3 the perceived video quality is similar for the different device resolutions. From level four and up it is not possible to maintain the same perceived video quality. However, when the network capacity allows it, the bitrate of the 1080p streams is higher than those of the 720p streams, and the bitrate of the 720p streams is higher than those of the 360p streams.

Table 2: Device-aware video quality mapping (in kbit/s)

Quality level	**360p**	**720p**	**1080p**
1	1000	2000	4000
2	600	1500	2000
3	400	1000	1500
4	400	600	1000
5	400	400	600
6	400	400	400

Figure 7 shows the model-based comparison between the policy that equally divides the available bandwidth among the players (Policy1) and the policy that takes the devices' resolutions into account as defined in Table 2 (Policy2). Players are started according to three independent Poisson processes with arrival intensities between $\lambda = 0.0025$ and $\lambda = 0.0030$ for 360p and 720p devices, and between $\lambda = 0.00125$ and $\lambda = 0.0150$ for 1080p devices. The arrival rate λ in Figure 7 is the combined arrival rate for the three independent Poisson processes.

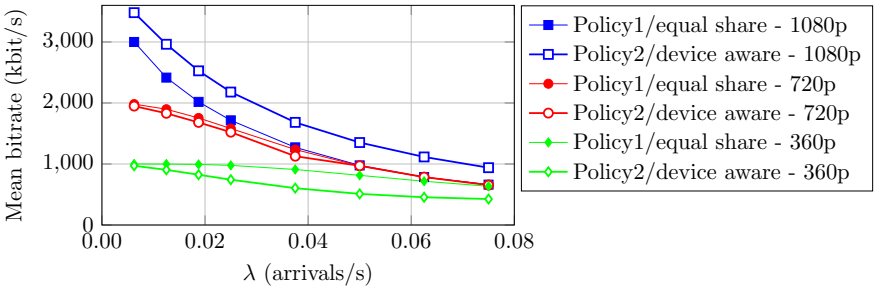

Figure 7: Model-based comparison of mean bitrates of a non-device aware and a device-aware sharing policy

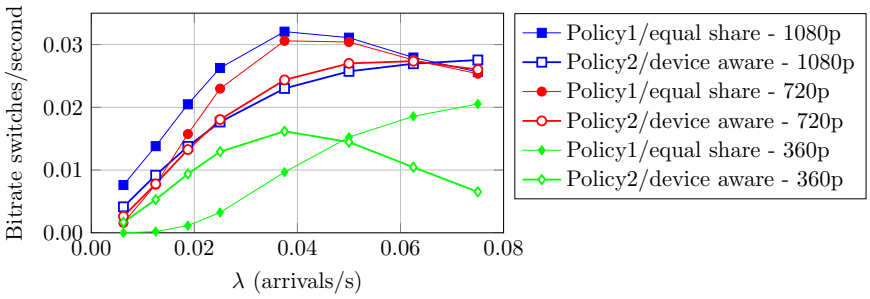

Figure 8: Model-based comparison of quality switches for a non device-aware and a device-aware sharing policy

For tablet devices the two policies do not show a difference in mean bitrate. However, it can be observed that the small screen devices are set to lower bitrates to make room for the big screen devices. This is a result of the quality level mapping from Table 2. Differences between different type of devices also shows in the number of bitrate switches in Figure 8. The biggest difference in the number of switches between the two policies is for 360p devices. This class shows opposite behavior for the two policies. Under Policy1 the small screen devices can stream at the highest available bitrate of 1000 kbit/s under low arrival rates. The fair share of bandwidth is likely to be above the highest bitrate. Therefore, 360p devices do not have to make a switch. When the load on the network becomes higher, the equal share of the available bandwidth is lower than 1000 kbit/s and requires the small screen devices to make a switch. This results in a higher number of switches for higher arrival rates.

Under Policy2, this effect is reversed. At the lower arrival rates, the 360p type devices have to make room for the large screen devices and thus switch to a lower bitrate. This is again a result from the quality mapping in Table 2. At high arrival rates the small screen players are likely to already stream at the lowest available bitrate and cannot make a switch anymore, resulting in a lower bitrate instability rate.

Figures 9 and 10 show the comparison of the model-based results with the results that we obtained through experimentation using our streaming testbed with the device aware policy (Policy2) installed on the proxy server. The results show that our model is highly accurate for both the mean bitrate of the video players as well as the bitrate instability rate.

4.2 Example: Premium users

The second example of sharing policies that we demonstrate in this paper are policies that differentiate between regular and premium users. The existence of premium users in a video delivery network can come from different reasons. For example, some devices are considered more important because they are being watched by multiple persons, or some users pay more for Internet access and

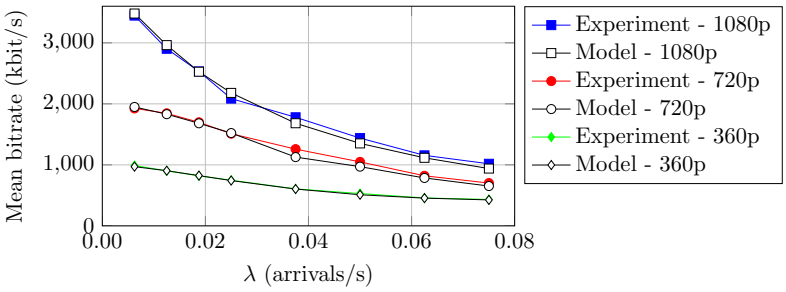

Figure 9: Model-based mean bitrate versus mean bitrate achieved in experiments for a device-aware sharing policy

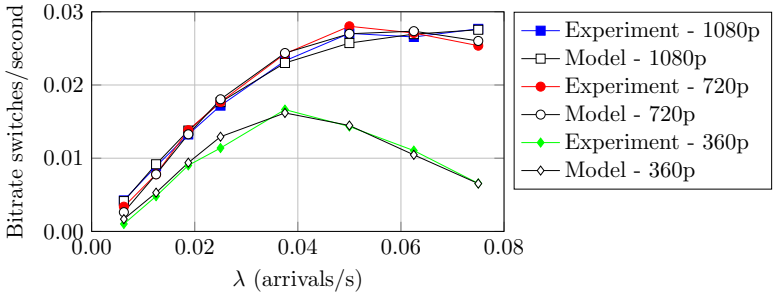

Figure 10: Model-based instability rate versus instability rate achieved in experiments for a device-aware sharing policy

therefore assume a higher video quality.

In our policies we consider two groups of users: regular users and premium users. Premium users can expect that the video quality of their stream will be higher than those of the regular users when the network allows for it. The first policy, PolicyA, gives the group of premium players the highest possible bitrate regardless of the bitrate for the regular players. The second policy, PolicyB, takes the same approach of selecting the highest possible bitrate for premium players, but will never select more than two bitrate-steps lower for regular users.

A video with a duration of 140 seconds encoded at 400, 720, 1020, 1600, 2300, and 4200 kbit/s is used in the evaluation. Both primary and regular players stream the same video. Regular players are started following a Poisson process with arrival rates between $\lambda = 0.0025$ and $\lambda = 0.0300$, primary players are started with arrival rates between $\lambda = 0.00125$ and $\lambda = 0.0150$. This results in an environment where there are on average twice as many regular players as there are primary players. Figure 11 shows the comparison of the two policies for primary users in terms of mean bitrate. Figure 12 shows the difference between the two policies in terms of number of bitrate switches.

The results show that PolicyB if more friendly towards regular players and

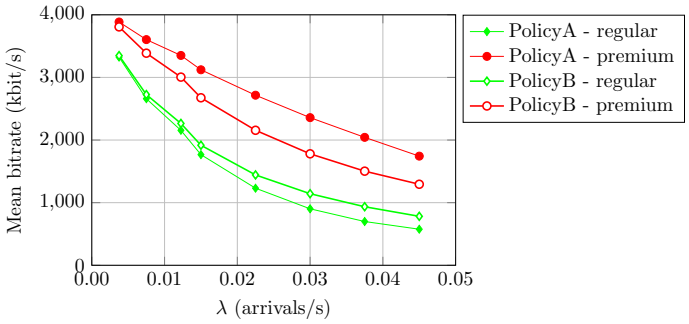

Figure 11: Model-based comparison for mean bitrates of two sharing policies for premium users

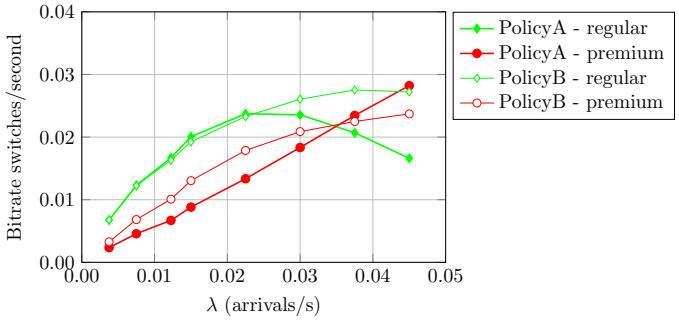

Figure 12: Model-based comparison for bitrate instability rates of two sharing policies for premium users

the difference between primary and regular players under PolicyB is smaller. This can be observed by the two lines representing Policy A to be closer to each other, compared to the two lines representing Policy B. However, the impact of switching from PolicyA to PolicyB is bigger for premium users compared to the gain for regular when looking at absolute bitrates. The differences between the two policies also shown in the comparison of the number of switches in video bitrate. Under PolicyA, premium users are kept longer on the high bitrates and thus require less switches at lower arrival rates. Regular users are the first to switch to lower bitrates when the network becomes loaded. At higher arrival rates it is more likely that regular players are already at the lowest bitrate, resulting in less quality switches.

For our proxy server we decided to implement PolicyA because it yields the highest bitrate for premium users. In the experimental runs a video stream with the same characteristics as in the comparison above is used. Figure 13 shows that the mean bitrates of the streams during the experimental runs are close to the model-based mean bitrate. Similarly, the model-based bitrate instability

shows to be highly accurate when comparing them to the number of bitrate switches achieved using the streaming tested, as displayed in Figure 14.

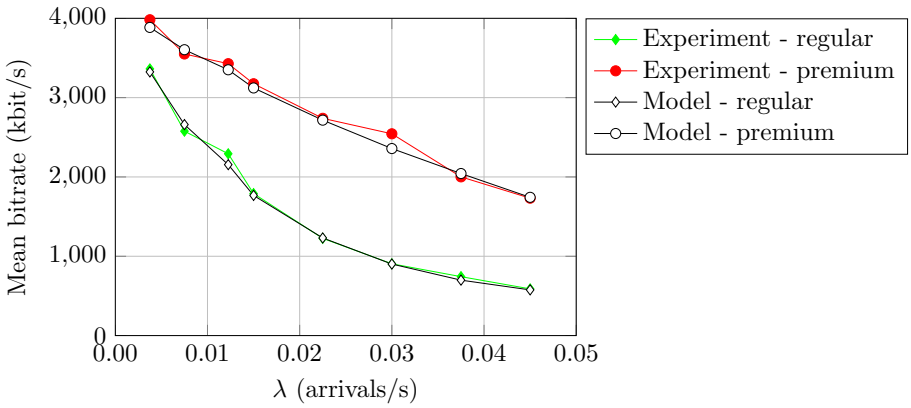

Figure 13: Model-based mean bitrate versus mean bitrate achieved in experiments for a premium device policy

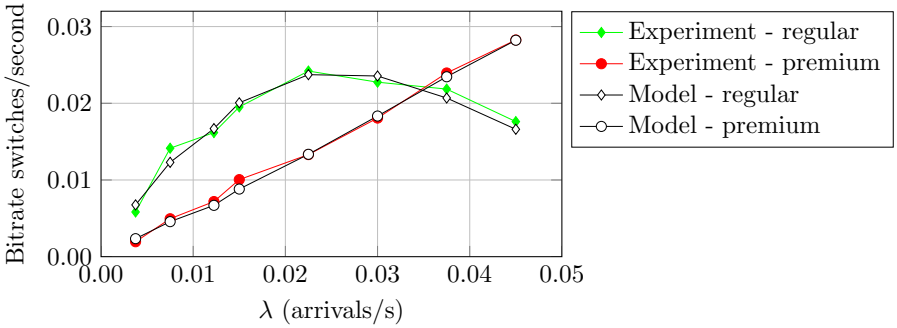

Figure 14: Model-based instability rate versus instability rate achieved in experiments for a premium device policy

From both examples in this section we can conclude that our model is sensitive to changing the sharing policy while and remains highly accurate.

5 Conclusion

Video streaming over the Internet is getting extremely popular. With the rise of handheld devices such as smartphones and tablets it is no longer an exception that multiple users share a network connection for video streaming. However, when this network connection contains a bottleneck that prohibits HTTP adaptive streaming players to stream at the highest bitrates, it is important to think

about how the capacity of the shared link should be shared among the players in order to provide an optimal viewing experience. Traditionally, HAS players are "selfish" in trying to achieve the highest possible video bitrate without taking into account the existence of other players in the network. With the introduction of HAS-assisting network elements, network connections can be shared more stable and fair, and policies that take into account various user and device specific factors can be executed.

Developing new sharing policies for network-assisted HAS requires the policies to be thoroughly tested and evaluated. However, simulation and experimental runs are time consuming and error prone. This motivated us to formulate a Markov model that can describe the performance of network-assisted HAS under a certain policy. The model presented in this paper allows to classify different types of players or streams, and to estimate the mean bitrate and number of changes in video bitrate for each class of players. The usage of our model is demonstrated by means of a model-based evaluation of two types of sharing policies: device heterogeneity and premium users. The model-based results are validated agains experimental runs using our streaming testbed and HAS-assisting proxy server. The results show to be highly accurate for both the mean bitrate and the bitrate instability.

As such, our model is a useful tool that can be used in the development of sharing policies, as well as for managing and provisioning video delivery networks. Given our model, a large number of configurations can be evaluated to come to the optimal configuration given a network setting.

Depending on where the bottleneck is located in the network, the model is aimed at network administrators and internet service providers. Network administrators can use the model as support while configuring HAS-assisting network elements such as our proxy server. ISPs can gain insights on HAS traffic requirements on a larger scale. Furthermore, they can use it for planning and provisioning a dedicated video-on-demand service over their IP network.

Future research efforts will focus on applicability and accuracy of our model in larger architectures with multiple bottleneck links, and how to express requirements and dependencies in the policy formulation. Furthermore, we will investigate the possibilities of online usage of our model in the proxy server, such that our proxy server, or multiple instances of the proxy server together, can dynamically update their internal sharing policies to provide an optimized viewing experience based on the current usage of the network.

References

[1] J. Ahrenholz. Comparison of CORE network emulation platforms. Technical report, Networked Syst. Technol., Boeing Research & Technology, Seattle, WA, USA, 2010.

[2] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis. What happens when HTTP adaptive streaming players compete for bandwidth? In *NOSSDAV '12: Proceedings of the 22nd international workshop on Network and Operating*

System Support for Digital Audio and Video, pages 9–14, New York, New York, USA, June 2012. ACM Request Permissions.

[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*, pages 157–168, New York, NY, USA, 2011. ACM.

[4] N. Cranley, P. Perry, and L. Murphy. User perception of adapting video quality. *International Journal of Human-Computer Studies*, 64(8):637–647, 2006.

[5] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and 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*, pages 21–26, New York, NY, USA, 2012. ACM.

[6] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. Towards network-wide QoE fairness using openflow-assisted adaptive video streaming. In *FhMN '13: Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, pages 15–20, New York, New York, USA, Aug. 2013. ACM Request Permissions.

[7] R. Hamberg and H. de Ridder. Time-varying Image Quality: Modeling the Relation between Instantaneous and Overall Quality. *SMPTE Journal*, 108(11):802–811, 1999.

[8] T. Hossfeld, M. Seufert, C. Sieber, and T. Zinner. Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming. In *Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on*, pages 111–116. IEEE, Sept. 2014.

[9] R. Houdaille and S. Gouache. Shaping HTTP adaptive streams for a better user experience. In *MMSys '12: Proceedings of the 3rd Multimedia Systems Conference*, pages 1–9, New York, New York, USA, Feb. 2012. ACM Request Permissions.

[10] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In *IMC '12: Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 225–238, New York, New York, USA, Nov. 2012. ACM Request Permissions.

[11] D. Jarnikov and T. Özçelebi. Client intelligence for adaptive streaming solutions. *Signal Processing: Image Communication*, 26(7):378–389, 2011.

[12] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *CoNEXT '12: Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108, New York, New York, USA, Dec. 2012. ACM Request Permissions.

[13] J. W. Kleinrouweler, S. Cabrero, R. van der Mei, and P. Cesar. Modeling Stability and Bitrate of Network-Assisted HTTP Adaptive Streaming Players. In *27th International Teletraffic Congress (ITC 27)*, Ghent, Belgium, Sept. 2015.

[14] C. Liu, I. Bouazizi, and M. Gabbouj. Rate adaptation for adaptive HTTP streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 169–174. ACM, 2011.

[15] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz. Adaptation algorithm for adaptive streaming over HTTP. In *Packet Video Workshop (PV), 2012 19th International*, pages 173–178. IEEE, 2012.

[16] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang. QDASH: a QoE-aware DASH system. In *MMSys '12: Proceedings of the 3rd Multimedia Systems Conference*, pages 11–22, New York, New York, USA, Feb. 2012. ACM Request Permissions.

[17] D. C. Robinson, Y. Jutras, and V. Craciun. Subjective Video Quality Assessment of HTTP Adaptive Streaming Technologies. *Bell Labs Technical Journal*, 16(4):5–23, 2012.

[18] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *Communications Surveys Tutorials, IEEE*, PP(99):1–1, 2014.

[19] E. Thomas, M. O. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey. Enhancing MPEG dash performance via server and network assistance. In *The Best of IET and IBC*, pages 48–53. Institution of Engineering and Technology, 2015.

[20] Z. Yuan, H. Venkataraman, and G.-M. Muntean. ibe: A novel bandwidth estimation algorithm for multimedia services over ieee 802.11 wireless networks. In *Wired-Wireless Multimedia Networks and Services Management*, pages 69–80. Springer, 2009.