Modeling stability and bitrate of network-assisted HTTP adaptive streaming players

Jan Willem Kleinrouweler*, Sergio Cabrero*, Rob van der Mei*†, Pablo Cesar*
*Centrum Wiskunde & Informatica †VU University
Amsterdam, The Netherlands Amsterdam, The Netherlands
Email: {j.w.m.kleinrouweler, cabrero, r.d.van.der.mei, p.s.cesar}@cwi.nl

***Abstract*—Viewers using HTTP Adaptive Streaming (HAS) without sufficient bandwidth undergo frequent quality switches that hinder their watching experience. This situation, known as instability, is produced when HAS players are unable to accurately estimate the available bandwidth. Moreover, when several players stream over a bottleneck link, their individual adaptation techniques may result in an unfair share of the channel. These are two detrimental issues in HAS technology, which is otherwise very attractive. To overcome them, a group of solutions are proposed in the literature that can be classified as network-assisted HAS. Solving stability and fairness only in the player is difficult, because a player has a limited view of the network. Using information from network devices can help players in making better adaptation decisions. In this paper we describe our implementation in the form of an HTTP proxy server. We show that with this proxy server both stability and fairness are improved. Furthermore, we present an analytical performance model that allows to accurately compute the number of changes in video quality and the bitrate of a video stream. The performance results of the HAS assisting proxy server, a real implementation, are used to validate our performance model. We believe that this model substantially eases future research of video delivery networks that use in-network HAS assistants. Using our model we are able to analyze stability and bitrate of HAS streams prior to real deployment, which saves time and money.**

I. Introduction

HTTP adaptive streaming (HAS) is becoming the dominant technology for video streaming over the internet. It quickly gained popularity because of its ability to adapt the video to the network conditions and other appealing properties such as simplifying firewall traversal and usage of normal Web servers. In HAS, a video file is split up into segments, typically with a duration between two and ten seconds. Each segment is encoded at several bitrates. A manifest file describes what the order and bitrate of the segments is. The video player processes this manifest file, and requests segments of a certain bitrate according to its own measurements of the available bandwidth.

In situations where the available bandwidth becomes lower, the video bitrate is lowered as well. This means that buffer under-runs can largely be avoided, and thus video playback is not interrupted. On the other hand, when more bandwidth becomes available, the player also adapts the video to a higher bitrate. Although this technology clearly has its advantages compared to non-adaptive streaming, it was found by several studies that the adaptation mechanisms in HAS players suffer performance problems when multiple players share a bottleneck network link [1][2][3]. The two most disturbing performance problems are *instability* and *unfairness*. Instability refers to the case when players switch too often between video qualities. In [4][5][6] it is shown that quickly changing between video profiles has a negative impact on the quality of the video watching experience. Unfairness means that some players receive at a high quality while other players receive at a low quality, even though it is possible for all players to stream at a quality that is somewhere in between.

The source for the performance problems can be found in the nature of HAS traffic and the bandwidth estimations of the players. HAS traffic is bursty: a segment is downloaded over TCP as fast as possible, and the download of the next segment is delayed to keep downloading and playback of segments aligned. Reference [2] describes how these ON-OFF download patterns create problems in estimating the available bandwidth when multiple players share a bottleneck network link. Wrong bandwidth estimations make that players switch back and forth between higher and lower quality profiles. When there is a change in available bandwidth it can take time before players stabilize at a certain bitrate, if players stabilize at all. And it does not mean that players select bitrates that are fair for all viewers in the network. Moreover, the problems described above go beyond video quality. Unstable and miscalculated use of network resources directly affects the network connection's stability and availability.

The multimedia community responds in two ways. One direction is to upgrade the players' adaptation mechanisms. More sophisticated algorithms with better heuristics and conservative switching between video profiles can lower the number of unnecessary switches and improve fairness. However, fixing these problems only in the players remains difficult because players have a limited view on the network. The alternative direction is about *using knowledge of devices within the network to assist players in making better adaptation decisions*. Several implementations have been proposed, including: traffic shaping at the residential gateway [7], signaling players from a measurement proxy [8], following the software defined network paradigm using OpenFlow [9], and our implementation that uses an HTTP proxy server as described later in this paper. Although the implementations are different, and each implementation has its own strengths and weaknesses, they take the same approach and have the same effect. By looking at HAS traffic at a *flow level*, HAS traffic becomes *predictable* and *manageable*.

The solutions that use in-network information to assist HAS players in performing the adaptation show promising results. Fairness is improved because the network devices assign each player a fair bandwidth, and stability improves

because unnecessary switches caused by the bursty nature of HAS traffic are avoided. These results make it interesting to further study this class of solutions. For example, to improve the fair sharing policies and determine which policy works best in which situation, or to deploy these solutions in bigger network architectures using more network devices. However, changing the sharing policy does have an effect on stability and bitrate of the players. And, combining the players, it has an effect on the utilization of the network channel. Building testbeds, to determine the performance of a policy in a certain architecture, is a costly and slow process. Therefore, to support and quicken this type of research, we propose an analytical performance model that allows to accurately compute both stability (i.e. how often does a video player changes the video quality) and the bitrate of a video stream. These two factors have been identified to play a major role in the *quality of experience* of the viewer.

The paper is organized as follows: in Section II our implementation of an in-network device that can assist HAS players in adaptation is presented; in Section III we describe the performance model; Section IV gives details on the testbed, in Section V we evaluate both the performance of our implementation, and the accuracy of the performance model; Section VI discusses our findings; and Section VII concludes this paper.

II. HAS PROXY

To reduce the number of unnecessary quality switches and improve fairness, the adaptation mechanism has to be enhanced. Several solutions for improving the adaptation mechanism in the player have been proposed in the literature [10][11][12][13][14]. However, these implementations are a trade-off between flexibility on the one hand, and conservativeness on the other. A too flexible mechanism can result in overestimation of the available bandwidth, and causes frequent switching between quality profiles. A streaming solution that responds too late can end up with occasional stalling and unfair sharing of the network connection's bandwidth. To overcome these problems the HAS player should look at streams of other HAS players on a flow level. To do so, the players could make use of network devices that have a broader view on the network compared to the view of player itself. These devices can then assist the player in selecting an appropriate bitrate. Several implementations of this concept, that can be classified as *network-assisted HAS*, have been proposed [7][8][9][15]. In this section we provide details of our implementation. The performance results of this implementation are used to validate the model in Section III.

A. Flow level monitoring

In our implementation we have the adaptation assistant in the form of an HTTP proxy server. The proxy server can be placed in the gateway, or another similar network device between player and server. All traffic with TCP destination port 80 is routed to the proxy server. This means that HTTP traffic is transparently forwarded to the proxy server and does not require any setup at the client. At the proxy server, HTTP traffic is monitored to detect starting and stopping players.

A starting player is detected by the proxy by the download of the manifest file. The proxy server will also process the manifest to obtain the characteristics of the stream. These characteristics are later used for fair sharing of the bandwidth. A player stops when the download of the last segment in the stream is completed. However, since users oftentimes stop a stream before it is finished, it is considered stopped after a period of inactivity. In steady-state mode, segments are requested with intervals equal to the duration of the segments. This means that if the download of a segment takes shorter then its duration, there will be a period of inactivity. The proxy server marks a player as stopped when this period of inactivity exceeds the duration of a segment.

B. Adaptation assistance

Since the proxy server has an overview of all HAS traffic on the shared connection, and it is aware of the characteristics for the streams, it is the designated device to make the division of the available bandwidth. In principle the proxy server assumes that every device demands the highest bitrate that is possible. Therefore it divides the bandwidth by the number of active players, and assigns each player the highest bitrate that is lower or equal to this fair share. Every time a new player starts or an existing player stops, the fair share is recomputed.

If the proxy server detects a player requesting a segment in a bitrate that is different from the assigned bitrate, it corrects this 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 HTTP header to the response informing the player about the rewrite. This is done so it does not throw off the bandwidth estimation in the player.

There are two occasions where rewriting is unwanted behavior. First, when the player is not able to reach the assigned bitrate. For example, when there is another bottleneck on the path from server to player. In this case the player signals the proxy server that it is not able to reach the target bitrate via an extra HTTP header. The second case is that a player requests a lower bitrate than the assigned bitrate, even though the assigned bitrate can be reached. An example reason for this is that the device that is used is not capable to display a higher quality video, or for bandwidth saving reasons. In this case the player also instructs the proxy server not to assign a higher bitrate, but with the difference that the proxy server is allowed to divide the remaining bandwidth over the other players.

C. Sharing policy

Although the implementation as HTTP proxy server is effective, the players do stream at the bitrates they are assigned to, the fairness policy is limited and simple. Currently, fairness means streaming at a similar bitrate. However, this does not necessarily give the same experience to all users since video quality is also dependent on device characteristics. In future versions of the proxy server the policy should be improved. In fact, we introduce the performance model in this paper to encourage experimenting and evaluation of different policies.

III. PERFORMANCE MODEL

The advantage of using the proxy is that players knows when a new player starts or an existing player stops, instead of detecting this through a change in available bandwidth using

bandwidth estimations. By using our proxy, or a different implementation of the same concept, the adaptation is managed and works following a predefined policy. In the performance model we use this same flow level view, and model starting and stopping players instead of the download of individual segments.

A. Starting and stopping players

We model starting and stopping players in the form of an Markov process. The Markov process is shaped in the same way as the Erlang multi-rate loss model. Although transport of HAS streams is done over TCP, which makes it elastic traffic, it is modeled as having a fixed duration. The reason for this is that the duration of a video cannot change, and that downloading of the HAS segments has to keep up with the playback in order to avoid interruptions in playback. To cope with this, the adaptation mechanism tries to make the download of a video segment take as long (or a little shorter) as the duration of this segment. If the network connection becomes loaded, and the download of a segment would take longer than its duration, the job size is decreased (selecting a lower bitrate). When the network connection becomes less loaded, it works the other way around. Therefore, HAS traffic is elastic in bandwidth, but not in time.

The state space of the Markov process is denoted by $\mathcal{S}$. Each state represents the number of players that is active. In theory every stream can have different characteristics (e.g. available bitrates and resolutions). However, in practice it is not uncommon that video streams are hosted by a few providers such as YouTube[1] or Netflix[2]. Each provider applies its own encoding scheme to newly added videos, which results in video streams coming from the same provider to have the same characteristics. Therefore, in our model we group players that stream videos with the same characteristics. The number of players of group k is denoted by n_k. Each state in the Markov process is described by a vector $(n_1, n_2, \ldots, n_K)$, where K is the number of groups that is considered. The state space is finite, because is does not make sense to have more video players then the network could handle. The state space is defined of all states $(n_1, n_2, \ldots, n_k)$ that satisfy the following condition:

$$\sum_{k=1}^{K} n_k \cdot B_{k_{min}} \leq C, \tag{1}$$

where C is the network capacity and $B_{k_{min}}$ the lowest available bandwidth for streams of group k. Not having more players than the network connection can handle is also enforced in the proxy: when the proxy detects the start of a player when there is no network bandwidth left, it will deny the player of service to ensure smooth playback of the other players.

Transitions between states are based on the rate that players of a certain group start and stop. The rate that players of group k start is denoted by λ_k. The rate that players of group k stop depend on the number of players n_k and the average duration of video streams in the group β_k. The rate that transitions $n_k \rightarrow n_k - 1$ occur is n_k/β_k. An advantage of having the Markov process shaped in the same way as the Erlang multi-rate loss model is, that for that model it is well known that a stationary distribution exists in the following product form:

$$\pi(n_1, n_2, \ldots, n_k) = \frac{1}{G} \prod_{k=1}^{K} \frac{(\lambda_k \beta_k)^{n_k}}{n_k!}, \tag{2}$$

where

$$G = \sum_{n \in \mathcal{S}} \prod_{k=1}^{K} \frac{(\lambda_k \beta_k)^{n_k}}{n_k!}. \tag{3}$$

Furthermore, for the Erlang multi-rate loss model it is known that it is insensitive to the distribution of the service times, in our case that means the durations of the video streams β_k. A summary of input parameters for our model can be found in Table I.

TABLE I. MODEL PARAMETERS

Parameter	Description
C	Capacity of the network connection
λ_k	Rate of the Poisson process at which users start the video streams
β_k	Mean duration
B_k	Available profiles
$T_{segment_k}$	Segment size

B. HAS bitrate and bitrate switches

The bitrate of a video stream, and how often this bitrate changes, also depends on the policy that divides the bandwidth between the players. Since one of the cornerstones of managing HAS traffic is to treat traffic with similar characteristics in the same way (i.e. enforce fairness), players that belong to the same group will get the same bitrate assigned. This makes that a policy is a function that takes the combination of players, their available bitrates and the network capacity as input, and outputs a vector $(q_1, q_2, \ldots, q_K)$ where q_k is the bitrate that is assigned to players of group k.

If the average number of players in group k is defined as:

$$\mathbb{E}[N_k] = \sum_{x \in \mathcal{S}} \pi(x) n_k^x, \tag{4}$$

then the average bitrate of streams in group k becomes:

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

where n_k^x is the number of players of group k, and q_k^x is the bitrate that is assigned to players of group k in state x in the Markov process.

The number of quality switches relates to how often the Markov process transitions between states. If the assigned bitrate for a group of players differs between two states, then a quality switch is made when the process transitions between those states. However, HAS players can only switch bitrates in between segments, not during download of a segment. Therefore, to mimic this behavior of the HAS player in our model, we observe the Markov process with intervals of $T_{segment_k}$ seconds. The probabilities that the process transitions from state x to state y in $T_{segment_k}$ seconds can be retrieved via

[1] http://www.youtube.com
[2] http://www.netflix.com

uniformization of the Markov process. The probability for a transition $x \to y$ in $T_{segment_k}$ seconds is denoted by $P_{x,y}$.

Since the duration of the videos is not necessarily equal (in fact the Markov process assumes the durations to be exponentially distributed), we express the number of changes in video quality not as an absolute number but as a rate: *number of bitrate switches per second*. The expected number of bitrate switches in a stream is defined as:

$$\mathbb{E}[Q_k] = \sum_{x,y \in \mathcal{S}} \frac{\pi(x) \cdot P_{x,y} \cdot n(x \to y)}{T_{segment_k} \cdot \mathbb{E}[N_k]}, \quad (6)$$

where $n_k(x \to y)$ is the number of players of group k that switch bitrates when transitioning from state x to state y. Only players that are active before and after the transition $x \to y$ can make a switch. A newly started player will already select the new bitrate, and a player that stops does not have to make a switch any more. Furthermore, switches are only made when in x a different bitrate is assigned for players of group k than in y. The number of players that switch bitrates when transitioning $x \to y$ then becomes:

$$n_k(x \to y) = min(n_k^x, n_k^y) \cdot min(1, |q_k^x - q_k^y|). \quad (7)$$

The overall (i.e. for all groups together) average bitrate and bitrate switch-rate can be found via a weighted average, weighted by the average number of players for each group $\mathbb{E}[N_k]$.

C. Defining groups

In the description above the argument for different groups is that videos can be encoded using different encoding schemes, however it is not limited to encoding schemes only. Grouping can also be used to express device heterogeneity, where different groups represent different devices and the sharing policy can make decisions based on what is best for each device type. Another example of using groups is to express premium users, where different groups represent the different plans. The idea behind dividing players into groups is that players within the same group are treated equally by the sharing policy, but players from different groups can be treated different.

IV. Testbed

The testbed consists of 17 PCs, two routers, and three servers that are connected as illustrated in Figure 1. The shared bottleneck link is the network connection between the two routers. It is set to 8 mbit/s to represent a network connection that is insufficient to support many players streaming a the highest bitrate. The round-trip delay between the servers is respectively 10ms, 20ms, and 40ms. All devices are implemented as virtual machines, in the form of lightweight Linux containers. Setting the network capacity and delay of the network connections is done using netem[3]. Our proxy server is installed on the router that is closest to the client machines. Although the bottleneck link is set to 8 mbit/s, we configured the proxy server with a channel capacity of 6.8 mbit/s, thus having a 15% percent margin. The safety margin is required for real-time streaming over TCP. Theoretical research [16] indicates that the network capacity should be twice the bitrate of the video streams, though this is overdone. Others [17] found that a 15% margin is sufficient for HTTP streaming. We follow this 15% margin for HAS.

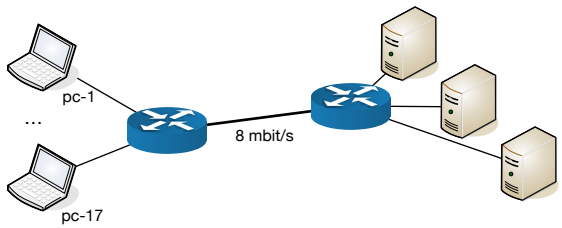

Fig. 1. Testbed architecture

Video players are started at free clients according to a Poisson process. Each client can start a single instance of the HAS player, and therefore the number of players that can be active at the same time is limited to 17. Starting more players would cause the connection to be congested. A player that starts when there are already 17 other players active is denied service. We use a custom HAS player that does not display the video to reduce CPU load. This player has a fairly aggressive adaptation mechanism that uses bandwidth estimations (weighted average $\alpha = 0.75$ of the throughput of the last two segments). The players stream one of two videos depending on the evaluated scenario. Video1 has a duration of 140.0 seconds and is encoded at 400, 720, 1020, 2300, and 4200 kbit/s. Video2 has a duration of 100.0 seconds and is encoded at 400, 800, 1200, 1600, 2200, 3000, and 4000 kbit/s. Both videos are split up into segment of 4.0 seconds using the HLS format. The manifests and the video segments are placed on the three Web servers that are running Apache 2.2.22. At the router closest to the server we record the HTTP requests using tcpdump.

The URL in each request is formatted in such a way that is contains an identifier for the player, an identifier for the video, the index of the segment, and the requested bitrate. Through analysis of the HTTP traces, we compute the following three performance metrics:

- Number of switches in bitrate per stream;
- Unfairness: an adaptation of Jain's fairness index [18]:

$$unfairness = \sqrt{1 - \frac{(\sum_{i=1}^{n} q_i)^2}{n \cdot \sum_{i=1}^{n} {q_i}^2}},$$

  where n is the number of players and q_i is the bitrate for each player [10]. A higher value means more unfairness;
- Average bitrate, and for each available bitrate the number of segments that are requested in that bitrate.

V. Evaluation

A. HAS proxy performance

To demonstrate the effectiveness of our implementation, we compare the performance of HAS players in a testbed with,

[3] http://linuxfoundation.org/collaborate/workgroups/networking/netem

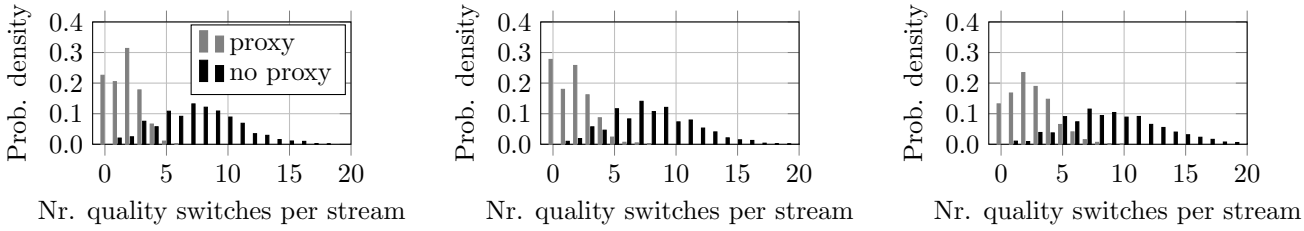

Fig. 2. Comparison of number of bitrate switches with the proxy server enabled and disabled for $\lambda_{0.020}$ (left), $\lambda_{0.030}$ (middle), and $\lambda_{0.045}$ (right).

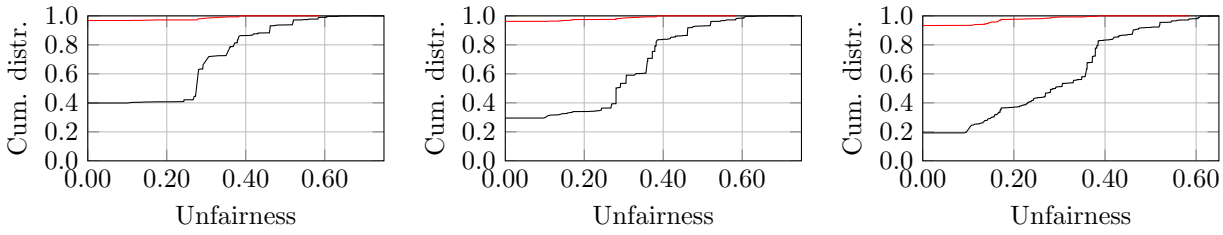

Fig. 3. Comparison of unfairness with the proxy server enabled and disabled for $\lambda_{0.020}$ (left), $\lambda_{0.030}$ (middle), and $\lambda_{0.045}$ (right).

TABLE II. PROXY SERVER PERFORMANCE COMPARISON

	$\lambda_{0.020}$	$\lambda_{0.030}$	$\lambda_{0.045}$
Proxy enabled:			
Players	1,631	2,422	
Bitrate switches	2,761	4,197	
Avg. switches per second (per stream)	0.01209	0.01238	
Avg. unfairness	0.0099	0.0104	
Avg bitrate (kbit/s)	1543	1198	
Proxy disabled:			
Players	1,674	2,470	3,783
Bitrate switches	12,592	19,787	34,336
Avg. switches per second (per stream)	0.05373	0.05722	0.06483
Avg. unfairness	0.2107	0.2485	0.2607
Avg. bitrate (kbit/s)	2242	1743	1208

and without, our proxy server enabled. At the clients, video players start streaming Video1 according to a Poisson process with the following rates: $\lambda = 0.020$, $\lambda = 0.030$, and $\lambda = 0.45$. For each condition a 24 hour run is done both with the proxy server enabled and disabled. The performance results are listed in Table II.

The results show that for $\lambda_{0.020}$ the average number of switches is lowered from 0.0537 to 0.0121 switches per second, a reduction of 77%. For $\lambda_{0.030}$ there is a similar reduction, and for $\lambda_{0.045}$ there are xx% less switches in quality. The distribution for the number of switches per player are shown in Figure 2. For $\lambda_{0.020}$ there were no players that switch more than 6 times when the proxy was enabled, compared to 18 when the proxy was disabled.

With our proxy enabled the average unfairness is xx% lower for $\lambda_{0.045}$, it goes down from 0.261 to 0.xx. With a lower number of active players, $\lambda_{0.020}$, the average unfairness is 95% lower. By looking at the cumulative distribution of unfairness over time in Figure 3, it can be seen that enabling the proxy server makes players receive the same bitrate more than 93% of the time (i.e. an unfairness value of zero). This is between 2.4 and 4.9 as much time compared to having the proxy disabled.

The average bitrate with the proxy server enabled is around 30% lower. For example, at $\lambda_{0.020}$, the bitrate without enabling the proxy is 2242 kbit/s, where it is 1543 kbit/s with the proxy enabled. Part of this can be explained by the fact that we avoid congesting in the proxy server by maintaining a 15% safety margin. The other part is because enforcing fairness typically comes at the price of a lower utilization.

B. Model validation

The model is validated by comparing it to performance results of the HAS proxy, a real implementation running in the testbed described above. Since the HAS proxy operates with a 15% safety margin on the capacity, we configure the model to use a capacity of $C = 6.8$ mbit/s. The fairness policy function matches the capacity dividing policy that is implemented in our HAS proxy.

First we compare our model with experiments using a single group of traffic that is playing Video1. The input parameters for model match the characteristics of the video, thus $\beta_1 = 140$, $B_1 = \{400, 720, 1020, 2300, 4200\}$ and

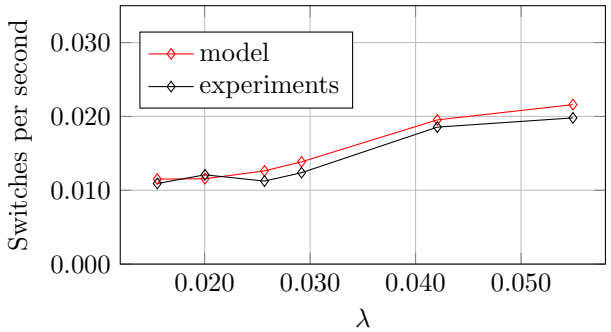

Fig. 4. Comparison of the modeled quality switch rate and the performance measured in experiments for a single traffic group.

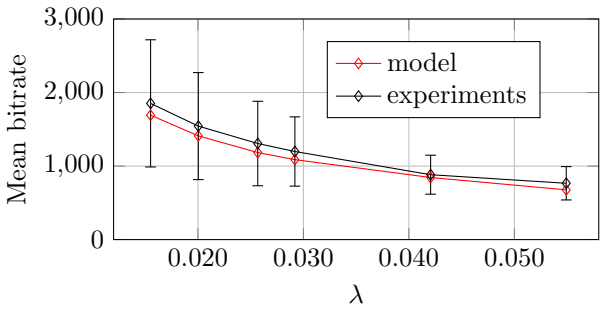

Fig. 5. Comparison of the modeled average bitrate and average bitrate obtained trough experimental runs for a single traffic group.

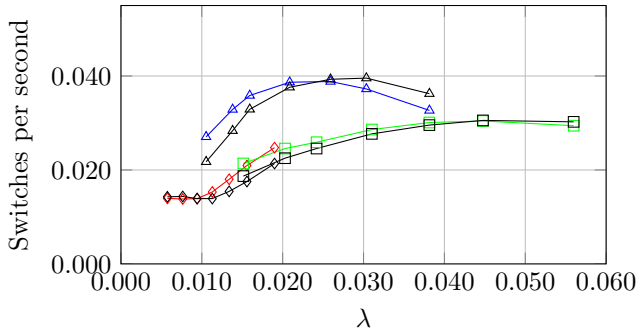

Fig. 6. Comparison of the modeled quality switch rate and the performance measured in experiments for two traffic groups.

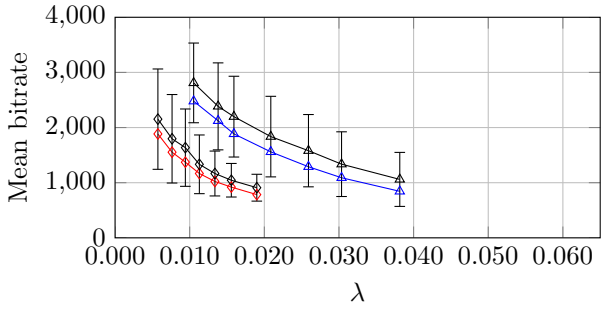

Fig. 7. Comparison of the modeled average bitrate (red/blue) and average bitrate obtained trough experimental runs (black) for traffic group 1 (diamond) and group 2 (triangle).

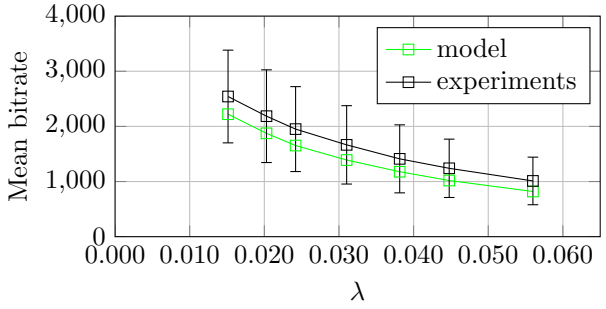

Fig. 8. Comparison of the modeled overall average bitrate and the average bitrate measured in experiments.

$T_{segment_1} = 4.0$. The arrival rate is varied from $\lambda_1 = 0.015$ to $\lambda_1 = 0.055$. Figure 4 shows the comparison between the model and the experiments for the rate of switches per second.

The results show that we can accurately estimate the number of switches with our model. The biggest difference between in switch rate between the model and the experiments is for $\lambda_1 = 0.055$, where the difference is 9%. To put this into perspective, for every 556 seconds of video that is played out, the model counts one more switch than was measured in the experiments.

The comparison between the model and the experiments in terms of average video bitrate is shown in Figure 5. The error bars show the standard deviation for the measured average bitrates of the players. For all tested arrival rates the model estimates the average bitrate slightly lower than what was observed in the experiments. The difference between model and experiments is on average 8.8%.

To demonstrate the accuracy of our model when using multiple traffic groups, we define two groups. The players in the first group stream Video1, and the players arrive with rates varying from $\lambda_1 = 0.005$ to $\lambda_1 = 0.019$. The second group of players stream Video2, thus with $\beta_2 = 100.0$, $B_2 = \{400, 800, 1200, 1600, 2200, 3000, 4000\}$ and $T_{segment_2} = 4.0$. The tested arrival rates for the second group of players are between $\lambda_2 = 0.010$ and $\lambda_2 = 0.038$.

The two videos cover more or less the same range of available bitrates, however they are distributed differently. Furthermore, Video2 has two more bitrates available. In each tested setting, the arrival rate for players of group 2 was twice as high as the arrival rate for players of group 1. The comparison of quality switch rates is given in Figure 6. The coloured lines show the modelled switch rate, where the black lines show the switch rates that were measured in the experiments. The points marked with a diamond express the switch rates for players of group 1, the switch rates for group 2 are indicated by a triangle. The overall switch rates, that is the average of switch rates for all players, are denoted by squares.

The average bitrates of the streams for players in group 1 and 2 are shown in Figure 7. The overall average bitrates are shown in Figure 8. Similar to the single traffic group setting, the modelled bitrate is is lower than the average bitrate than was observed in the experiments. On average the model estimated the bitrate 16% lower compared to the bitrates achieved using our HAS proxy.

VI. Discussion

HAS was brought with the promise that there is no need for special QoS systems, because TCP would handle reliability and congestion avoidance. However, in absence of a QoS system, HAS players suffer from instability and unfairness. Though there is still no need for a full fledged QoS system, we show that managing HAS traffic with a simple, and lightweight, adaptation assistant located in a network element, does help improve stability and fairness. Our evaluation of the HAS proxy server shows that viewers experience up to XX% less quality switches and the average unfairness over time is reduced by XX%.

The policy that we used in the HAS proxy server is basic, and should be improved. However, it serves its purpose in

showing the link between the model and the experimental results. With our model we can accurately estimate the rate at which HAS players switch quality. Though, the modeled average bitrate is lower than average bitrate that was obtained through experimental runs. We expect that this issue is caused by the buffer management of the player that we used in the experiments. The player has a large buffer of 24 seconds, it can hold up to five video segments. When a player starts, it will continue downloading segments until the buffer reaches its maximum fill level. The effect of this is that players are slightly shorter active on the network connection than the duration of the video. The model will overestimate the active number of players at the same time, and thus present a lower average bitrate. We decided not to include buffer management since it is highly player specific. Furthermore, when smaller buffers are used this effect is reduced.

VII. Conclusion

Video streaming over the Internet is becoming more and more popular. It is no longer an exception that several users stream videos at the same time using the same network connection. Current HAS players exhibit two performance problems: instability and unfairness. In-network solutions that assist HAS players in making adaptation decisions, such as our proxy server, have shown to be effective. However, the results of adaptation decisions (i.e. how they are experienced by the viewers) depend on the chosen policy. Improvement of these policies requires further research. To support and quicken this research we introduced a performance model that can estimate the number of switches in quality and the resulting bitrate of the video streams, based on selected sharing policy.

Our model supports groups of players that can be treated differently by the sharing policy. These groups can for instance be used to differentiate between encoding schemes, devices types and capabilities, and premium users. We compared our model to a real implementation of an in-network adaptation assistant. The evaluation results show that our model can accurately estimate the number of times a video player switches quality, and it can give a reasonable estimation of the average bitrate of the streams.

In future research, we will evaluate new sharing policies with our model that can be used in the HAS proxy server, or in similar implementations. Besides that, we plan to improve the accuracy of our model, and extend the model to be used in architectures that contain multiple bottleneck links. Furthermore, we will investigate the possibilities of using this model as a basis for a model that supports managing HAS traffic in the presence of significant background traffic.

References

[1] 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*. New York, New York, USA: ACM Request Permissions, Jun. 2012, pp. 9–14.

[2] 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*. New York, NY, USA: ACM, 2011, pp. 157–168.

[3] 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*. New York, New York, USA: ACM Request Permissions, Nov. 2012, pp. 225–238.

[4] N. Cranley, P. Perry, and L. Murphy, "User perception of adapting video quality," *International Journal of Human-Computer Studies*, vol. 64, no. 8, pp. 637–647, 2006.

[5] R. Hamberg and H. de Ridder, "Time-varying Image Quality: Modeling the Relation between Instantaneous and Overall Quality," *SMPTE Journal*, vol. 108, no. 11, pp. 802–811, 1999.

[6] 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.

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

[8] 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*. New York, New York, USA: ACM Request Permissions, Feb. 2012, pp. 11–22.

[9] 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*. New York, New York, USA: ACM Request Permissions, Aug. 2013, pp. 15–20.

[10] 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*. New York, New York, USA: ACM Request Permissions, Dec. 2012, pp. 97–108.

[11] 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*. Springer, 2009, pp. 69–80.

[12] D. Jarnikov and T. Özçelebi, "Client intelligence for adaptive streaming solutions," *Signal Processing: Image Communication*, vol. 26, no. 7, pp. 378–389, 2011.

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

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

[15] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. D. Vleeschauwer, W. V. Leekwijck, and F. D. Turck, "QoE optimization through in-network quality adaptation for HTTP adaptive streaming," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 2012, pp. 336–342.

[16] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia streaming via TCP: An analytic performance study," *Transactions on Multimedia Computing, Communications, and Applications (TOMM*, vol. 4, no. 2, pp. 1–22, May 2008.

[17] A. Biernacki and K. Tutschku, "Performance of HTTP video streaming under different network conditions," *Multimedia Tools and Applications*, vol. 72, no. 2, pp. 1143–1166, 2014.

[18] R. Jain, D. Chiu, and W. Hawe, "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems," *arXiv.org*, Sep. 1998.