Improving Video Quality in Crowded Networks Using a DANE

Jan Willem Kleinrouweler
Centrum Wiskunde & Informatica
Science Park 123
1089 XG, Amsterdam, The Netherlands
j.w.m.kleinrouweler@cwi.nl

Britta Meixner
Centrum Wiskunde & Informatica
Science Park 123
1089 XG, Amsterdam, The Netherlands
britta.meixner@cwi.nl

Pablo Cesar
CWI, Science Park 123
1089 XG, Amsterdam
Delft University of Technology
Mekelweg 4
2628 CD, Delft, The Netherlands
p.s.cesar@cwi.nl

ABSTRACT

Dynamic Adaptive Streaming over HTTP (DASH) is a technology for delivering video content over the Internet. It provides an effective mechanism, which has been adopted by major content providers. Nevertheless, available DASH player implementations have a number of drawbacks such as performance problems on shared network connections, which lead to video freezes and frequent video quality changes. In this paper, we propose a method to reduce the performance problems that exist in networks with a large number of DASH players. These networks can be found in hotels, apartment complexes, and airports. In experiments with up to 600 simultaneously active players, we are able to reduce the number of DASH players with freezes by 95% (from 345 to 15) compared to throughput-based adaptation and by 75% (from 62 to 15) compared to BOLA using our DASH Assisting Network Element (DANE). In addition, we reduced the number of quality switches by 94% compared to throughput-based adaptation, and by 85% compared to BOLA.

CCS CONCEPTS

•Information systems →Multimedia streaming; •Networks →*Network management;*

KEYWORDS

Dynamic adaptive streaming over HTTP, HTTP adaptive streaming, Video streaming, Network assistance, Performance

ACM Reference format:
Jan Willem Kleinrouweler, Britta Meixner, and Pablo Cesar. 2017. Improving Video Quality in Crowded Networks Using a DANE. In *Proceedings of NOSSDAV'17, Taipei, Taiwan, June 20-23, 2017,* 6 pages.
DOI: http://dx.doi.org/10.1145/3083165.3083167

1 INTRODUCTION

Over-the-top video streaming is a popular application that accounts for a large share of Internet traffic. Estimates report that YouTube and Netflix account for 53% of downstream traffic on fixed links in North-America during peak hours in 2016 [17]. Both content providers use Dynamic Adaptive Streaming over HTTP (DASH) for delivering video content [15], which provides scalable distribution of video by reusing the current HTTP-based infrastructure. Using DASH, videos are split into segments which are encoded in different qualities. These segments are then requested by the DASH players. Content providers are able to simultaneously serve large numbers of users using Content Delivery Networks (CDNs). However, previous studies showed that DASH suffers performance problems on the receiving end, especially in settings with shared network connections or background traffic [2, 3, 12]. Performance problems manifest themselves to the user as video/playback freezes and frequently changing video quality (i.e. bitrate). These effects contribute to a lower Quality of Experience (QoE), resulting in viewers' disengagement and abandonment [7, 18]. However, many DASH players being simultaneously active in larger shared networks may become a standard scenario given the popularity of DASH-based video streaming for entertainment. Large shared networks can for example be found in hotels, apartment complexes, and airports. To the best of our knowledge, related work on DASH performance problems or potential solutions are so far mostly limited to networks which are used by a small number of DASH players (max. 150 concurrent players).

In this paper, we focus on larger networks that should facilitate many simultaneously active DASH players. We confirm performance problems (video/playback freezes and frequently changing video quality) of two established adaptation algorithms (throughput-based and Buffer Occupancy-based Lyapunov Algorithm (BOLA) [20]) in a real network with up to 600 concurrent DASH players. We propose a solution using network assisted DASH which seeks to facilitate more DASH players in a network and to improve the viewers' QoE. In our experiments, we show that the network assisted DASH solution outperforms the throughput-based algorithm and BOLA by reducing the number of DASH players with freezes by 95% at best. The number of quality switches is lowered by up to 94% at best. Thereby, we answer the following research questions:

(1) Do the same DASH streaming performance problems (stalling and video quality instability) exist in networks with up to 600 active players as they do in networks with a small number (up to 12) of active players?
(2) Can Network Assisted DASH improve the DASH streaming performance (reduce stalling, reduce instability, increase video bitrate, improve unfairness) in environments with a large number of players?

The remainder of this paper is structured as follows: In Section 2, we provide an overview of related work on DASH performance problems and network assisted DASH. Section 3 details the streaming testbed and experimental design. In Section 4, we present the performance evaluation results and show improvements using our network assisted DASH solution. Section 5 concludes the paper.

2 RELATED WORK

DASH[1] is the dominant technology for online video streaming. It provides players a manifest, which is a list of representations of the same video. The representations can differ in resolution and bitrate. The DASH player selects one of the representations based on the current network conditions, buffer level, or host device capabilities [19]. Although DASH is widely implemented, it has been identified that DASH suffers performance problems in shared networks. Huang et al. show that the video bitrate for one DASH player significantly decreases after starting a competing TCP flow [12]. In [3] the authors analyze how off-the-shelf DASH players react to changing network conditions and how video quality becomes unstable when two DASH players compete for bandwidth. Akhshabi et al. further break down this issue and explain how bandwidth estimations are invalidated as a result of on-off download patterns of DASH players [2]. They evaluate a network with up to 12 players and show that the number of active players affects the number of quality switches. Esteban. et al [8] analyze TCP behavior. They point out that the double feedback loop - both TCP and DASH adaptation algorithms react to changing network conditions - causes video quality instability. Freezes and frequent quality switches negatively impact the viewers' QoE [10, 18].

An established approach to counter DASH performance problems is to improve the adaptation algorithm in the player. However, DASH players have a limited view of the network. DASH Assisting Network Elements (DANEs) have a better view on network activity and can use this information to assist DASH players in adaptation. Houdaille and Gouache implement assistance through shaping the DASH flows [11]. Traffic shaping decreases the number of quality switches. In [5, 14] DASH traffic is routed through a proxy server. The proxy server changes the manifest files and the segment requests to move the players to the desired bitrate. Petrangeli et al. combine multiple proxy servers to manage three sub-networks [16]. Techniques from Software Defined Networking (SDN) are applied in [4, 6, 9, 13]. The DASH assistants are implemented as network controllers that provide traffic shaping and signaling target bitrates to DASH players. However, with a large number of players it is not feasible to maintain one traffic queue per player. It is unknown how many players can be combined into a queue. Furthermore, with many changing DASH players the overhead of control messages becomes large.

The number of DASH players used in the evaluations of the before mentioned related work does not exceed 12 players for papers on DASH performance problems, and 150 players for papers on network assisted DASH. An overview of the number DASH players used in related work is given in Table 1. In this paper we evaluate a network with up to 600 simultaneously active DASH players. We confirm DASH performance problems (freezes, quality switches), and show how our DANE can improve DASH streaming performance.

[1]In this section we refer to DASH as the technology, not the MPEG-DASH standard

Table 1: Overview of the number of DASH players used in related work

Topic	Reference	Max. # players
DASH performance problems	Huang et al., 2012 [12]	1
	Esteban et al, 2012 [8]	1, or more
	Akhshabi et al., 2011 [3]	2
	Akhshabi et al., 2012 [2]	12
Network assisted DASH	Houdaille and Gouache, 2012 [11]	2
	Georgopoulos et al, 2013 [9]	3
	Kleinrouweler et al, 2016 [13]	4
	Kleinrouweler et al, 2015 [14]	25
	Bouten et al, 2012 [5]	50
	Bentaleb et al, 2016 [4]	50
	Petrangeli et al, 2015 [16]	90
	Cofano et al, 2016 [6]	150

3 EXPERIMENTAL SETUP

In this section we describe the network assisted DASH implementation, the custom DASH player used in out experiments, the testbed setup, and the experiment settings.

3.1 DASH Assisting Network Element

We use our implementation (first introduced in [13]) of a DANE to configure traffic control and assist DASH players. The DANE consists of three components: a Network Bridge (NB), a Network Controller (NC), and a Service Manager (SM).

The *Network Bridge* is built from PC hardware using two Ethernet interfaces. The interfaces are bridged together and perform packet forwarding and traffic control. We implement traffic control with Linux `tc` using Hierarchical Token Buckets (HTBs)[2]. The *Network Controller* is software that configures traffic control on the NB. It provides a programming interface that the SM uses to reserve bandwidth for DASH players. By default the NC configures one traffic queue, but it can set up additional queues dedicated to DASH traffic. We extended the Network Controller so it can be configured to create any number of queues for DASH traffic. It will automatically balance active DASH flows over different queues. DASH queue rates determine the minimum throughput for traffic in that queue (i.e. a throughput guarantee is given). One queue is designated for control messages between the DASH players and the DANE. Prompt delivery of control messages is essential for network assisted DASH to be effective. The queue is overprovisioned at a rate of 50 Mbit/s, to make sure that control messages are delivered without delay. The queue is configured to use available bandwidth for DASH traffic if it is not needed for control messages.

The *Service Manager* is the entity that assists the DASH players. A schematic overview of the interactions involving the SM is shown in Figure 1. DASH players request assistance from the DANE by connecting to the SM via WebSocket and report the representations from the DASH manifest (Figure 1, step 1). The SM takes the reported representations and divides the available bandwidth among the players. In our experiments, we assume that all devices have the same form factor and priority. Therefore, we equally divide the available bandwidth among DASH players. The SM communicates the target representations to the NC (Figure 1, step 2) which will

[2]http://luxik.cdi.cz/~devik/qos/htb/ (last accessed: February 21, 2016)

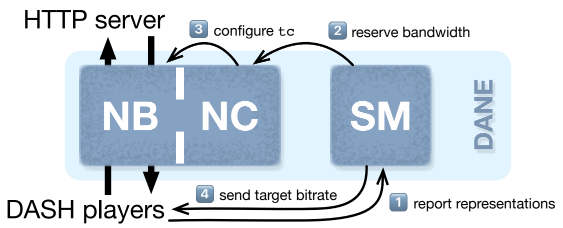

Figure 1: DANE components and interactions

configure traffic control on the NB (Figure 1, step 3). The target representations are also sent to the DASH players (Figure 1, step 4).

We implemented the SM in JavaScript within Node.js[3], which is able to handle a large number of simultaneous WebSocket connections. We restricted the number of updates from SM to DASH players and NC to one update per two seconds limiting the number of update messages caused by changing DASH players. The original implementation from [13] had to be improved, because update messages would not arrive in time or were outdated at their arrival.

3.2 Headless DASH player

We use the low-performance computing device Raspberry Pi as a platform for our players. To be able to start multiple DASH players on a single Raspberry Pi, we created a custom headless version of the DASH.js player using Node.js. Video decoding and displaying consume the most CPU resources which is not a problem for a single player on a device. However, we want to investigate the network transmission part, which is the bottleneck in the described setting. For this reason decoding and displaying are not implemented in our player. Playback of a video segment is simulated by a timer with the length of the video segment. Disabling decoding and rendering does not change the networking behavior of the player. As a consequence, our player gives comparable results to the DASH.js player. To reduce memory usage, the player maintains a buffer of small fake video segments. After downloading a video segment, it is discarded and we only store metadata (i.e. the size and duration of the video segment) in the buffer. With this approach we are able to run 20 DASH players on a single Raspberry Pi while keeping CPU usage below 60% and memory usage below 720 MB (out of 1 GB available), ensuring that the hardware is not the limiting factor in our experiments. Running more players at the same time showed increasingly unstable results. Given our testbed with 30 Raspberry Pis and 20 DASH player instances, we can emulate up to 600 simultaneously active DASH players.

The headless player implements three adaptation algorithms: throughput-based, BOLA, and assisted adaptation. Throughput-based and BOLA (i.e. the adaptation- and abandonment rules) are implemented as in the DASH.js v2.2.0[4] player. For assisted adaptation we use our custom adaptation rule that works as follows: DASH players receive quality recommendations from the SM. If the buffer level is higher than 10 seconds and BOLA indicates a video quality equal or higher than the recommended quality, the recommended quality is used. Otherwise the DASH players selects

[3]https://nodejs.org/en/ (last accessed: February 20, 2017)
[4]https://github.com/Dash-Industry-Forum/dash.js/ (last accessed: February 23, 2017)

Algorithm 1: Assisted adaptation rule

```
Inputs  : $b \leftarrow$ current buffer level
          $q_{dane} \leftarrow$ target quality from DANE
          $q_{bola} \leftarrow$ quality determined by BOLA
          $followDane \leftarrow true$ if previous segment got
              quality level from DANE, $false$ else
Outputs : $q_{segment} \leftarrow$ quality for next segment
          $followDane \leftarrow true$ if next segment gets
              quality level from DANE, $false$ else
1  $q_{segment} = \mathtt{min}(q_{dane}, q_{bola})$
2  if $b \geq 10$ seconds then
3      if $q_{bola} \geq q_{dane} \vee followDane$ then
4          $q_{segment} = q_{dane}$
5          $followDane = true$
6      else
7          $followDane = false$
8  else
9      $followDane = false$
10 return $q_{segment}, followDane$
```

the quality as determined with the BOLA algorithm. Assisted adaptation has the effect that in most cases the quality recommendation from the SM is adopted by the DASH players. In cases of starting player or players with unexpected lower network performance, the assisted adaptation rule relies on BOLA to provide a better quality recommendation than the SM. The pseudo-code for the assisted adaptation rule is given in Algorithm 1. The adaptation rule is executed before each segment download.

3.3 Testbed

We evaluate the throughput-based, BOLA, and assisted adaptation algorithms in a wired streaming testbed. The testbed consists of three network switches, an HTTP server, a PC that functions as our DANE, and 30 Raspberry Pis hosting DASH players. Figure 2 shows the network diagram for our experiments. The network switches, DANE, and HTTP server are connected via 1 Gbit/s Ethernet. The Raspberry Pis are connected via 100 Mbit/s Ethernet, because of Raspberry Pi network interface limitations. The secondary switches connect to five Raspberry Pis each, to prevent internal bottlenecks (i.e. five Raspberry Pis together cannot download more than 500 Mbit/s on a shared 1 Gbit/s link). The primary switch is configured as a plain Layer-2 switch (not prioritizing or limiting the network performance of the ten Raspberry Pis connected to the secondary switches). Hard- and software settings are listed in Table 2.

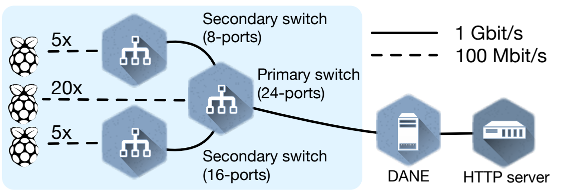
5x
20x
5x
Secondary switch (8-ports)
Primary switch (24-ports)
Secondary switch (16-ports)
1 Gbit/s
100 Mbit/s
DANE
HTTP server

Figure 2: Network diagram showing the experimental setup

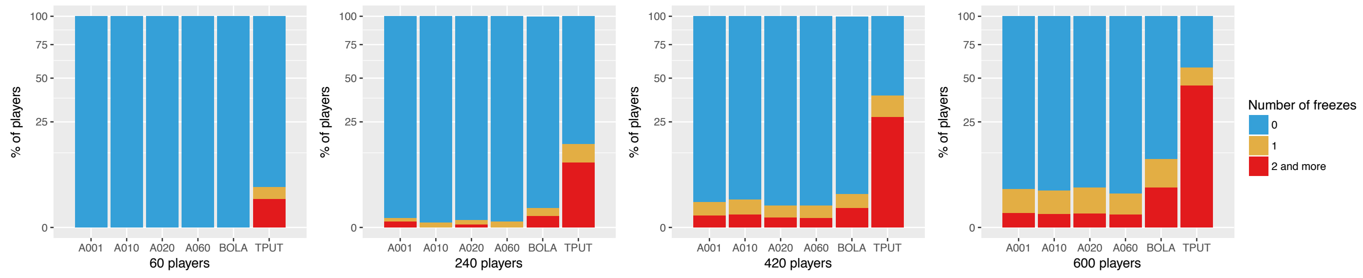

Figure 3: Percentage of DASH players that experiences none, one, and two or more freezes. (Y-axis has squared scale)

Table 2: Testbed hardware/software overview

Device	Description	#
Raspberry Pi 3 model B, Raspbian Jessie Lite, nodejs 6.9.4	DASH player host	30
HP ProCurve 25106-24 (J9279A), 24 ports, firmware Y.11.49, configured as L2 switch	Main switch	1
NETGEAR ProSafe GS116, 16 ports	Secondary switch	1
NETGEAR ProSafe GS108, 8 ports	Secondary switch	1
PC with Intel Core i5-5250U (quad core) 1.6 GHz, 8GB RAM, Debian Linux 8, nodejs 6.9.4	DASH assistant and traffic shaper	1
PC with Intel Core i3-M350 (quad-core), 2.27 GHz, 4GB RAM, Debian Linux 8, nginx 1.10.1	HTTP server	1

3.4 Experiments

The video clip that we use in our experiments comes from the movie Sintel[5]. The video is prepared for DASH streaming using the MPEG-DASH Live Profile [1] in compliance with the guidelines of the DASH Industry Forum. The stream is available in 12 different representations[6], with a segment size of two seconds.

The evaluation of the adaptation algorithms and our network assisted DASH solution is spit into two experiments:

Experiment 1: Parallel start-up. In this experiment we start $n \in \{60, 240, 420, 600\}$ DASH players at the same time. This scenario resembles the start of a popular stream (e.g. a video going viral or an important live event such as a big soccer match). We perform 20 runs per setting to account for the small variations between the runs. Each DASH player streams a video of 180 seconds.

Experiment 2: Poisson process start-up. DASH players are started following a Poisson process with arrival rate $\lambda \in \{0.9, 1.9, 2.9\}$ and a video duration of 180 seconds. This will give on average 150, 330, and 510 simultaneously active DASH players. The actual number of DASH players varies over time. The maximum number of active DASH players cannot exceed 600. Due to the use of a Poisson process, we perform one run of two hours for each arrival rate λ. After two hours the mean number of players is converged with a variation of less than 0.05 players per 10 seconds.

[5]https://durian.blender.org (last accessed: February 20, 2017)
[6]296Kbit/s@240p, 395Kbit/s@240p, 493Kbit/s@360p, 732Kbit/s@360p, 971Kbit/s@480p, 1.458Kbit/s@480p, 1.934Kbit/s@720p, 2.878Kbit/s@720p, 3.779Kbit/s@1080p, 5.544Kbit/s@1080p, 7.234Kbit/s@1440p, 10.563Kbit/s@1440p

4 RESULTS

In this section we present the results for the experiments. We look at video freezes, bitrate, and quality switches in each experiment. A freeze is an interruption in playback and occurs when the buffer in the DASH player is empty. In our analysis we count the number of freezes per player. We report the performance of the adaptation algorithms in terms of freezes as percentages of players that experienced none, one, and two or more freezes while streaming the video. We do not consider the duration of a freeze in our evaluation. The video bitrate is determined for each player as the mean of the bitrates of the video segments for that player. The standard deviations (denoted by σ) indicate the variations of mean bitrates between the players. Each time that two consecutive segments have a different bitrate is defined as a switch. The number of switches is counted independent from the size of the switch (meaning that differences of more than one quality level according to the DASH manifest are counted as one switch). We use the acronyms *"TPUT"* for throughput-based adaptation, *"BOLA"* for BOLA, and *"Axxx"* for assisted adaptation, where *xxx* stands for the number of traffic shaping queues that we configured.

4.1 Parallel start-up

We start the evaluation with the experiments where many DASH players were started in parallel. This experiment represents popular events such as the start of a live stream. The setting poses extra difficulty on the DASH players because the large number of buffering DASH players put an extra demand on the network compared to players that have a full buffer. This difficulty is visible when looking at freezing players as shown in Figure 3. TPUT shows a high number of freezes, even when only 60 players are active. The percentage of players that experienced a freeze went up to 54.8% for 600 concurrent players. When using BOLA, the number of freezes was lower, but still 10.5% of the player froze in the setting with 600 simultaneously active players. This indicates that freezes are a performance problem for DASH adaptation algorithms, when a large number of players starts streaming at the same time. The number of players that experiences a freeze increases with the total number of players. We were able to the reduce the percentage of freezing players to 2.6% for 600 started players using our DANE. DANE assisted players will not request video segments in a bitrate higher than recommended (see Algorithm 1, line 1), and thus reduce competition for bandwidth between players that causes freezes.

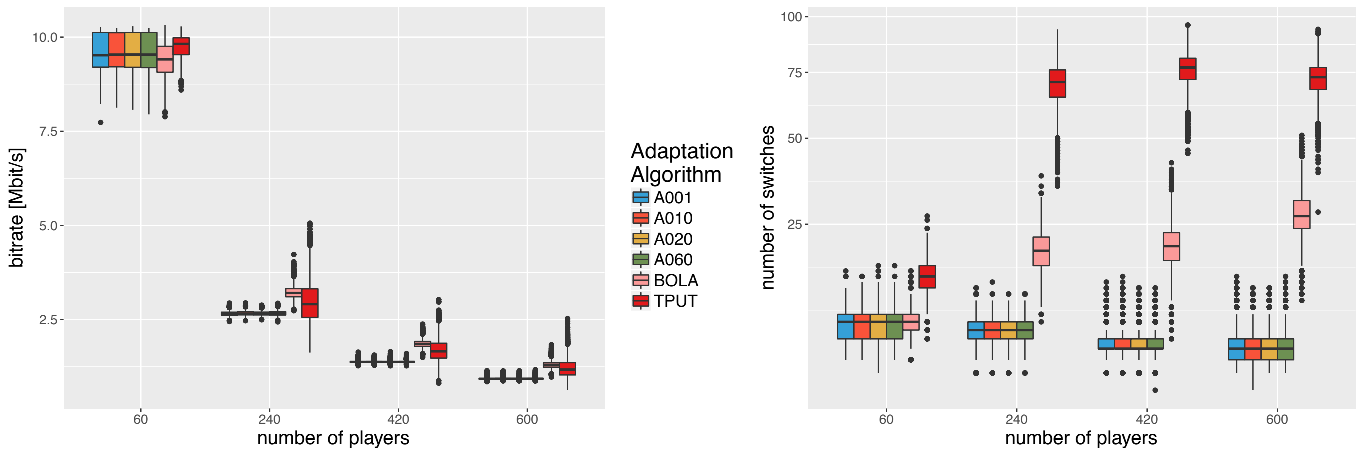

Figure 4: Distribution of mean video bitrates (left) and number of quality switches (right). (Y-axis denoting the number of quality switches has squared scale)

The distribution of mean video bitrates is shown in Figure 4 (left). The results show that the mean video bitrate is generally lower when using network assistance compared to TPUT and BOLA. For example, with 240 players the bitrate for A060 is 17% lower compared to BOLA and 10% lower compared to TPUT (A060: 2.66 Mbit/s ($\sigma = 0.06$), BOLA: 3,22 Mbit/s ($\sigma = 0.17$), TPUT: 2,95 Mbit/s ($\sigma = 0.53$)). The reason why the video bitrate is lower when using assisted adaptation is twofold. First, the SM maintains a 20% safety margin when assigning bitrates to DASH players. As a result, at least 20% of the bandwidth remains free. However, through experimentation we found that this safety factor is necessary for stable streaming (reduce freezes and number of quality switches). The second reason for the lower bitrate is the fact that the SM assigns all DASH players the same bitrate. In future versions of our implementation, this could be improved by assigning a subset of DASH players a higher bitrate to increase network utilization. For example, players with a higher priority could get a higher bitrate assigned.

The number of quality switches is shown in Figure 4 (right). Both BOLA and TPUT cause a high number of quality switches when the number of players increases. BOLA switches on average 27.60 ($\sigma = 5.37$) times for 600 concurrent players . TPUT switches on average 72.54 ($\sigma = 6.44$) times. This means that for our three minute video clip (90 segments of two seconds), 81% of consecutive segments had a different bitrate. Frequent quality switches are thus also a DASH performance problem in network with up to 600 players. We significantly reduced the number of quality switches using assisted adaptation. For 240 players with A060, the mean number of switches was 4.11 ($\sigma = 1.32$), a reduction of 85% compared to BOLA and 94% compared to TPUT. The number of switches for A060 decreases with the number of players, in contrast to the other algorithms. More players result in a lower recommended bitrate from the SM. It takes the DASH players less switches at start-up to reach this bitrate.

The differences between the different queuing configurations (A001-A060) in terms of freezes, bitrate, and quality switches are small. We performed a Kruskal-Wallis test[7] ($\alpha = 0.05$) with multiple comparison posthoc test ($\alpha = 0.05$) to determine the differences. We summarize the results as follows: There are no significant differences between the number of queues with regards to freezes. There are also no significant differences with 60 simultaneous players with regards to video bitrate and quality switches. With regards to averages for video bitrate and number of switches for 240 players and more, there are slight differences between settings A001 and A060, in favor of A060. Nevertheless, the practical values show small differences and the number of queues thus has only limited effect.

4.2 Poisson process start-up

In the second experiment, we start DASH players following a Poisson arrival process to target scenarios where players gradually start and stop. This setting targets environments such as hotels where people continuously watch video clips. We observe a lower percentage of players that freeze during playback for this setting. For example, for $\lambda = 2.9$ with an average of 510 concurrent players, TPUT caused freezes for 13% of the players. For BOLA the percentage of freezing players was below 1%. This means that freezes are a performance problem in networks with a large number of DASH players depending on which adaptation algorithm is used. With assisted adaptation (A060), we eliminated freezes ($\leq 0.01\%$). For $\lambda \leq 1.9$, the percentages of freezing players is below 1% for all algorithms.

We list the mean bitrates for the three different arrival rates in Table 3. In terms of mean bitrate, we observe a comparable performance for the three adaptation algorithms. During the experiments there is always a fair share of the DASH players that is buffering. Those players put a larger demand on the network compared to players in steady-state mode. We expect that buffering players limit steady-state players in selecting higher-bitrates. With assisted adaptation, the safety margin of 20% can be used by buffering players, thus not affecting the video quality of steady-state players.

[7] numerical data, non-normal distribution, unpaired

Table 3: Bitrates and switches for Poisson process start-up

	Adaptation algorithm	Mean bitrate Mbit/s (σ)	Mean nr. switches (σ)
$\lambda = 0.9$	A060	3.91 (0.43)	**6.20 (2.97)**
	BOLA	**5.17 (0.71)**	20.80 (4.11)
	TPUT	4.81 (0.90)	47.81 (9.38)
$\lambda = 1.9$	A060	2.11 (0.29)	**5.02 (1.30)**
	BOLA	**2.38 (0.33)**	17.97 (3.92)
	TPUT	2.17 (0.34)	54.21 (7.96)
$\lambda = 2.9$	A060	1.50 (0.25)	**7.57 (4.32)**
	BOLA	**1.54 (0.24)**	21.40 (4.83)
	TPUT	1.36 (0.21)	63.27 (7.89)

With regards to the number of switches the performance is similar to the experiments where the DASH players start at the same time. This means that the performance issue of quality fluctuations also exists in the scenario where DASH players gradually come and go. Using our DANE we on average reduce the number of quality switches by 65% compared to BOLA, and 88% compared to TPUT for $\lambda = 2.9$. This shows that network assisted DASH is not only able to provide more stable streaming in scenarios where players start in parallel, but also in scenarios where the number of simultaneously active DASH players continuously changes.

5 CONCLUSION

Online video streaming using DASH is a popular application that accounts for a large share of today's Internet traffic. However, it has been identified that DASH suffers video freezes and frequent quality switches on shared network connections. In this paper, we demonstrate that these performance problems not only exist in well studied small networks, but also in networks with a large number of simultaneously active DASH players. Through experiments with up to 600 concurrent players, we have shown that two established adaptation algorithms, throughput-based and BOLA, cause freezes and high numbers of quality switches. Two distinct scenarios were evaluated: players starting at the same time and players randomly starting with a Poisson process. Using our DASH assisting network element we were able to significantly reduce the number of playback freezes and quality switches. As such, we demonstrated that network assisted DASH is an effective solution that results in an improvement of the viewers' Quality of Experience even in networks with larger numbers of players.

In this paper, we put our focus on the interplay between the DASH players. In future work, we will extend our evaluations with a part of the nodes in the testbed that will generate background traffic. Furthermore, we will look into differentiation of DASH players by different form factors and capabilities (e.g. screen size and resolution, battery level). With regards to network assisted DASH, the bandwidth sharing algorithms in the Service Manager should be able to divide the bandwidth giving a comparable QoE to different users. The Service Manager has to react fast to accommodate for the changing DASH players. We will also work on the development of a heuristic that is based on objective QoE models.

ACKNOWLEDGEMENTS

Parts of this work were carried out during the tenure of an ERCIM 'Alain Bensoussan' Fellowship Programme.

REFERENCES

[1] 2014. ISO/IEC 23009-1 Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats. (2014).
[2] Saamer Akhshabi, Lakshmi Anantakrishnan, Ali C Begen, and Constantine Dovrolis. 2012. 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*. ACM Request Permissions, New York, New York, USA, 9–14.
[3] Saamer Akhshabi, Ali C Begen, and Constantine Dovrolis. 2011. An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*. ACM, New York, NY, USA, 157–168.
[4] Abdelhak Bentaleb, Ali C. Begen, and Roger Zimmermann. 2016. SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking. In *Proceedings of the 2016 ACM on Multimedia Conference (MM '16)*. ACM, New York, NY, USA, 1296–1305.
[5] N. Bouten, J. Famaey, S. Latr, R. Huysegems, B. D. Vleeschauwer, W. V. Leekwijck, and F. D. Turck. 2012. QoE optimization through in-network quality adaptation for HTTP Adaptive Streaming. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*. 336–342.
[6] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo. 2016. Design and Experimental Evaluation of Network-assisted Strategies for HTTP Adaptive Streaming. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 3, 12 pages.
[7] Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, and Hui Zhang. 2013. Understanding the Impact of Video Quality on User Engagement. *Commun. ACM* 56, 3 (March 2013), 91–99.
[8] Jairo Esteban, Steven A. Benno, Andre Beck, Yang Guo, Volker Hilt, and Ivica Rimac. 2012. Interactions Between HTTP Adaptive Streaming and TCP. In *Proceedings of the 22Nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '12)*. ACM, New York, NY, USA, 21–26.
[9] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. 2013. 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*. ACM Request Permissions, New York, New York, USA, 15–20.
[10] Tobias Hofeld, Michael Seufert, Christian Sieber, Thomas Zinner, and Phuoc Tran-Gia. 2015. Identifying QoE optimal adaptation of {HTTP} adaptive streaming based on subjective studies. *Computer Networks* 81 (2015), 320 – 332.
[11] Rémi Houdaille and Stéphane Gouache. 2012. Shaping HTTP adaptive streams for a better user experience. In *MMSys '12: Proceedings of the 3rd Multimedia Systems Conference*. ACM Request Permissions, New York, New York, USA, 1–9.
[12] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. 2012. Confused, timid, and unstable: picking a video streaming rate is hard. In *IMC '12: Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM Request Permissions, New York, New York, USA, 225–238.
[13] Jan Willem Kleinrouweler, Sergio Cabrero, and Pablo Cesar. 2016. Delivering Stable High-quality Video: An SDN Architecture with DASH Assisting Network Elements. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA, Article 4, 10 pages.
[14] Jan Willem Kleinrouweler, Sergio Cabrero, Rob van der Mei, and Pablo Cesar. 2015. Modeling Stability and Bitrate of Network-Assisted HTTP Adaptive Streaming Players. In *27th International Teletraffic Congress (ITC 27)*. Ghent, Belgium.
[15] Stefan Lederer. 2015. Why YouTube & Netflix use MPEG-DASH in HTML5. Availble online https://bitmovin.com/status-mpeg-dash-today-youtube-netflix-use-html5-beyond/ (Last accessed February 8, 2017). (Februari 2015). https://bitmovin.com/status-mpeg-dash-today-youtube-netflix-use-html5-beyond/
[16] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2 (Oct. 2015), 28:1–28:24.
[17] Sandvine, Inc. 2016. Global internet phenomena report. Available online https://www.sandvine.com/trends/global-internet-phenomena/ (last accessed February 8, 2017). (2016).
[18] R. K. Sitaraman. 2013. Network performance: Does it really matter to users and by how much?. In *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*. 1–10.
[19] I Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *Industry and Standards* (2011).
[20] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9.