A

An SDN Architecture for Privacy-friendly Network Assisted DASH

Jan Willem Kleinrouweler, Centrum Wiskunde & Informatica
Sergio Cabrero, Centrum Wiskunde & Informatica
Pablo Cesar, Centrum Wiskunde & Informatica

Dynamic Adaptive Streaming over HTTP (DASH) is the premier technology for Internet video streaming. DASH efficiently uses existing HTTP-based delivery infrastructures implementing adaptive streaming. However, DASH traffic is bursty in nature. This causes performance problems when DASH players share a network connection or in networks with heavy background traffic. The result is unstable and lower quality video. In this article, we present the design and implementation of a so-called DASH Assisting Network Element (DANE). Our system provides target bitrate signaling and dynamic traffic control. These two mechanisms realize proper bandwidth sharing among clients. Our system is privacy friendly and fully supports encrypted video streams. Trying to improve the streaming experience for users who share a network connection, our system increases the video bitrate and reduces the number of quality switches. We show this through evaluations in our Wi-Fi testbed.

CCS Concepts: •**Information systems** → **Multimedia streaming;** •**Networks** → *Network management;*

Additional Key Words and Phrases: DASH, HTTP adaptive streaming, Video streaming, Network assistance, Wi-Fi, Performance

1. INTRODUCTION

Internet video streaming is an application that uses a substantial share of the network bandwidth. Streaming services, like YouTube and Netflix, are very popular [Sandvine, Inc 2016]. Combining this with the large number of Internet-enabled devices which are around us, we can conclude that users sharing a network connection for video streaming is no longer an exception. Dynamic Adaptive Streaming over HTTP (DASH) is the dominant technology for delivering video content. However, DASH underperforms in networks with multiple DASH players or with heavy background traffic [Akhshabi et al. 2011][Akhshabi et al. 2012]. This results in lower video bitrate and changes in video quality. Both negatively impact the user quality of experience (QoE) [Cranley et al. 2006][Hamberg and de Ridder 1999][Robinson et al. 2012]. In this article, we present our design and implementation of a DASH Assisting Network Element (DANE) that mitigates these issues.

Off-the-shelf DASH players compete for bandwidth with each other, and with other traffic on the network link [Akhshabi et al. 2011]. Networks with concurrent traffic offer little to no support for improving video streaming performance. TCP, which is used by DASH players for transport, requires large windows sizes to reach throughputs sufficient for video streaming. DASH players periodically download short video segments. This bursty on/off traffic pattern prevents the TCP window to grow. This results in DASH players streaming at suboptimal bitrates. State-of-the-art adaptation mechanisms for DASH players improve the stability of a stream, but they cannot in-

Author's addresses: J.W.M. Kleinrouweler, S. Cabrero, P.S. Cesar, Centrum Wiskunde & Informatica, Science Park 123, 1098 XG Amsterdam, The Netherlands; P.S. Cesar, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands.

DOI: http://dx.doi.org/10.1145/0000000.0000000

crease the throughput that is needed for high-quality video. Furthermore, bandwidth estimations remain challenging because players have a limited view of the network activity and cannot make assumptions on the underlying network infrastructure

To overcome these issues, we present an architecture for network assisted DASH in this article. With our architecture, we keep the DASH protocol stack simple and scalable for content providers. We add a websocket connection to an element that we call the Service Manager. The Service Manager divides available bandwidth in the network among DASH players and other services. Through target bitrate signaling and dynamic traffic control mechanisms, our system improves the QoE of users by increasing the video bitrate and reducing the number of quality switches.

We introduced an early version of our system in [Kleinrouweler et al. 2016]. This article extends [Kleinrouweler et al. 2016] with the following contributions:

— An improved implementation that is based on the new version of the DASH.js player (v2.2.0)[1]. We achieve up to 50% faster adaptation to the optimal bitrate for starting players.
— The Buffer Occupancy based Lyapunov Algorithm (BOLA) [Spiteri et al. 2016], which adapts the video bitrate is implemented in the DASH.js player. It is included in the evaluation.
— Two traffic control configurations are added to our network switch. We evaluate video streaming performance using a single DASH service queue and multiple client queues (i.e. each DASH player has its own queue). Both types of queues can be set up as rate limiting queue or minimum rate queue.
— The effect of the queuing configuration on the throughput for background traffic is evaluated.
— A bandwidth sharing policy taking devices type into account is implemented and analyzed.

The remainder of this article is organized as follows: Section 2 describes and discusses the related work. Section 3 outlines the design of our system. Section 4 describes the experimental setup. Section 5 provides a thorough evaluation of our system and shows its effectiveness. Section 6 summarizes this article and gives an overview of future work.

2. RELATED WORK

DASH is the primary technology for Internet video streaming and large content providers. For example, YouTube and Netflix implement this technology in their players [Lederer 2015]. DASH provides a manifest (i.e. list of representations of the same video) to the player, where representations differ in bitrate and resolution [Sodagar 2011]. The player selects one of the available representations in the general DASH architecture. DASH players base their decisions on current network conditions, the current buffer level, or take the capabilities of the host device into account. However, off-the-shelf DASH players show that they have difficulties selecting a bitrate when multiple DASH players share a bottleneck link [Akhshabi et al. 2011][Akhshabi et al. 2012][Huang et al. 2012]. In this case, the players suffer instability and unfair network resource sharing. The on/off download patterns for DASH segments cause both over- and underestimations of available network resources. The double feedback loop – both DASH player and TCP respond to changes in bandwidth – eventually leads to quality oscillations and unfairness [Esteban et al. 2012]. The throughput for video segments does not match the throughput of other services when a DASH player shares a network connection with other bandwidth demanding applications.

[1] https://github.com/Dash-Industry-Forum/dash.js (last accessed January 17, 2017)

Lower video bitrate and frequent changes in bitrate lower the overall quality of experience (QoE) and diminish user engagement [Cranley et al. 2006][Dobrian et al. 2013][Robinson et al. 2012]. Maximizing the video bitrate while keeping the number of switches to a minimum increases the QoE for the viewer [Hoßfeld et al. 2015]. Several proposals that improve the performance of adaptation algorithms in DASH players appeared in the literature [Jiang et al. 2012][Liu et al. 2011][Miller et al. 2012][Jarnikov and Özçelebi 2011][Spiteri et al. 2016]. However, as we will show in this article, changing the adaptation algorithm in the player has limited effect in crowded networks. The stream will still end up having a too low bitrate. The model where DASH players are fully in charge of selecting the video quality limits the ability to configure network resource sharing among clients and services. In this article, we will show how centralizing this decision process could lead to a more meaningful (i.e. assign more network resources that require more) sharing of network resources.

As an alternative to changing the adaptation logic in the DASH player, network elements can be used to assist players in selecting a video representation. So-called DASH assisting network elements (DANEs) [Thomas et al. 2015], have better knowledge of the network capabilities and a better view of the current network activity. Based on this information, DANEs can support DASH players to make more informed decisions when selecting a video representation. In Houdaille et al. [2012], DASH traffic is implicitly supported by limiting the flows of DASH streams to prevent players from selecting too high bitrate video segments. Others route DASH traffic through a DASH-aware proxy server [Bouten et al. 2012][Kleinrouweler et al. 2015]. The proxy servers alter the content of the manifest and segment requests to move players to the desired bitrate. Petrangeli et al. [2015] utilize a chain of proxy servers to address networking infrastructures with multiple bottlenecks. Georgopoulos et al. [2013] present an OpenFlow-enabled system with an orchestrating module that signals the representation that DASH players should select. The drawback of these implementations is that they use intrusive means for detecting DASH traffic and require the DANE to inspect the manifest to obtain the DASH characteristics. Depending on who owns the DANE (user, Internet service provider, or content provider) this is an invasion of the users' privacy, as the DANE has full knowledge of what a user is watching.

Our system (first presented in [Kleinrouweler et al. 2016]) works on a "need to know" basis, where DASH players only communicate essential information to the DANE. Moreover, as we maintain an out-of-band communication channel to the DASH player and do not touch the content of the original video flow, our implementation is fully compatible with encrypted streams (i.e. HTTPS). Our system also takes background traffic into account. The implementations mentioned above are evaluated in networks with only DASH traffic, and thus cannot guarantee that their adaptation assistance is effective in environments with background traffic. Unfair distribution of bandwidth between background traffic and DASH streams results in lower video quality [Huang et al. 2012]. We will demonstrate that providing Quality of Service (QoS) support in the network is essential when implementing a DANE.

Type and format of messages between DASH player and DANE used in our system are similar to those specified in the Server and Network Assisted DASH (SAND) proposal [Thomas et al. 2015]. Our system fits into this design as it implements a DANE. It could easily be extended to comply with the SAND message format once standardized. However, SAND only specifies the message exchange but leaves bandwidth sharing policies and enforcement of these policies up to the DANE implementors.

3. SYSTEM DESIGN

We designed our system to be deployed in a Software Defined Networking (SDN) environment. In SDN, network control is separated from the data forwarding plane. We follow this approach in our four layers architecture, as shown in Figure 1. The layers (from bottom to top) are: assistance enabled DASH players, programmable network hardware, a Network controller, and a DASH-aware Service Manager. Together the components carry out stable and high-quality DASH streaming.

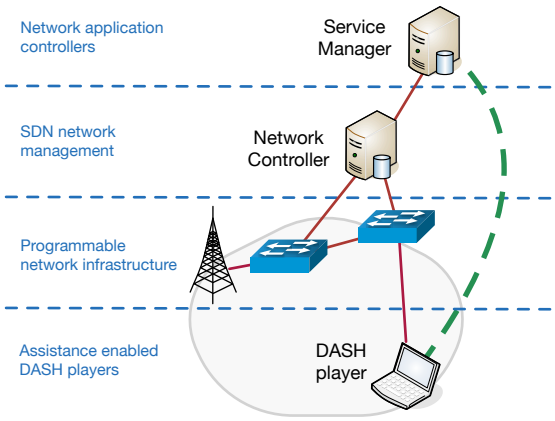


Fig. 1: Overview of our SDN-based architecture. The dashed line between the DASH player and the Service manager marks the communication channel for sending adaptation assistance messages.

Communication between DASH player and DANE can be done through in-band and out-of-band messaging. In-band messaging piggy-bags adaptation instructions into the data flow, for example via additional HTTP headers. However, we have chosen for out-of-band messaging via an extra connection between DASH player and Service Manager. This design provides the benefits of privacy friendliness, support for encryption, and flexible placement of Network Controller and Service Manager. In our system, we separate the content of the stream (i.e. manifest and video segments) from the messages for adaptation assistance. Using this messaging scheme, the Service Manager does not inspect the content of the stream. Messages for adaptation assistance contain only essential information needed by the Service Manager to function. The Service Manager needs to know that a DASH player is streaming and the set of representations for the video. However, it does not need to know which video a user is watching. This results in an increase of privacy for the user.

Nowadays, many content providers serve their video content exclusively over HTTPS. With our out-of-band messaging scheme we support encrypted streams. The Service Manager does not need to touch the content of the stream. Different encryption keys can be used for video content and adaptation assistance messages. As a result, the privacy of the user increases even further.

We have the flexibility to position the Network Controller and Service Manager at different locations in the network. For example, in a home network the Wi-Fi router hosts both Network Controller and Service Manager. In an office network the Network Controller and Service manager can be located at separate servers. For larger scale

networks, like an ISP access network, offloading to the cloud is a possibility. The only requirement is that the DASH players can establish a connection with the Service Manager.

3.1. Programmable Network Hardware and Network Controller

Programmable network hardware (e.g. switches, routers, Wi-Fi access points) build the basis of our system. In SDN, programmable network elements have two functions: First, they collect device status and network statistics. Second, they execute sophisticated forwarding rules. The Network Controller is the central entity that monitors and configures the SDN network elements. It has a global view of the network and dynamically adds and deletes forwarding rules in the switches.

The Network Controller in our system provides transparent routing from the DASH player to the Service Manager. The DASH player may not know up front how to contact the Service Manager because it can be positioned in different places in the network. The DASH players open a connection to the Service Manager using a generic address. The Network Controller (transparently) routes this connection to the Service Manager. We use the standard protocol OpenFlow [McKeown et al. 2008] for the interface between Network Controller and network hardware.

The second responsibility of the Network Controller is traffic control configuration. Although OpenFlow has the `enqueue` message to put packets into queues, it lacks messages to create, modify, and destroy queues. Therefore, we implemented an additional interface on our switch that allows queue configuration. By default, the Network Controller configures a queue for background traffic. Background traffic can use the full bandwidth of the channel when there are no DASH players active. When additional queues for DASH traffic are configured, the size of the background queue is automatically adjusted.

The traffic control implementation in our switch offers two ways of creating queues to the Network Controller: rate limit based or minimum rate based. A rate limiting queue restrains the throughput for packets in the queue. Rate limiting queues cannot share unused bandwidth with other queues. In contrast, a minimum rate queue guarantees a minimum throughput for packets in that queue. This type of queue does impose a maximum throughput, and remaining bandwidth can be shared among queues.

In our system design, we distribute network management over the Network Controller and the Service Manager. The Network Controller focusses on forwarding, routing, and shaping packets in the network. However, it has no knowledge of DASH players. The Service Manager is aware of DASH and instructs the Network Controller to configure the network based on its DASH specific information. The advantage of this approach is a possible generalization to other services. If we would implement a service manager for another (non-DASH) service, this service could also benefit from assistance in the network.

3.2. DASH-aware Service Manager

DASH players connect to the Service Manager to receive adaptation assistance. This makes the Service Manager aware of active DASH players. The Service Manager combines knowledge of DASH players and network capabilities which it receives from the Network Controller. Based on this information, it divides the available bandwidth among the DASH players. Depending on its configuration, the Service Manager equally divides the bandwidth among players, or takes other parameters (e.g. device type) into account.

The adaptation actions from the Service Manager propagate in two ways: via target bitrate signaling on the control channel and through dynamic traffic control in the switch. DASH players use the target bitrate as guidance in their adaptation al-

gorithm. The traffic control mechanism allocates enough network resources for the managed link. DASH players now have enough network resources to stream at the target bitrate.

We provide two traffic control configurations in our system: per service and per client. Per service traffic control closely relates to differentiated services (DiffServ). This method creates a single queue for DASH traffic, while other traffic goes into a background queue. The size of the service queue depends on the number of active DASH players, and the bitrate assigned to each player by the Service Manager (i.e. the size of the queue is the sum of all bitrates). The Service Manager reconfigures the queue when a new DASH player starts or an active player stops.

The second traffic control configuration creates a dedicated queue per DASH player. In this mode, the Service Manager separates DASH from background traffic, and individual players from each other. Using this configuration, the Service Manager can better enforce policies where it assigns a different bitrate to a different client. It creates a new queue when a DASH player starts and destroys this queue after the player stops.

For both traffic control configurations, we add a safety margin to the size of the queue. Through experimentation we found that the size of the queue(s) should be 1.2 times the (aggregated) target bitrate, ensuring smooth playback and healthy buffer levels. Furthermore, the safety margin provides robustness against variations in video bitrate (e.g. as a result of VBR encoding).

3.3. Assistance-enabled DASH Player

Existing DASH players need to be modified to support adaptation assistance from the Service Manager. The DASH player connects to the Service Manager via a websocket. Websockets offer a two-way communication channel and are a widely implemented web technology. Leveraging websockets makes our setup compatible with DASH players that work in the browser. Moreover, websockets do not create issues with firewalls and Network Address Translation (NAT).

DASH players open the websocket connection to the Service Manager during their initialization phase. Opening the connection indicates the Service Manager that DASH streaming is imminent. After receiving the manifest from the server, the DASH reports the bitrates from the manifest (for both audio and video) to the Service Manager. The player may leave out some of the (higher) bitrates in this report. For example, it can ignore the bitrates that exceed the capabilities of the device. If the player wants to change the set of bitrates during playback (e.g. the user resizes the player to fullscreen), it sends the new collection of bitrates to the Service Manager. The Service Manager redivides the bandwidth when it receives such an update. Players that pause or end a stream indicate this to the Service Manager using separate messages, or by closing the websocket.

The Service Manager applies the sharing policy after it receives a set of bitrates from a DASH player. The result is a target bitrate for each player. The target bitrate denotes the representation for which the Service Manager thinks it is the best representation given current network activities. The Service manager sends each DASH player its target bitrate in a control message. Besides target bitrate signaling, the Service Manager also instructs the Network Controller to configure traffic control, using the target bitrate as an indication for the size of the queues. The sequence of messages between DASH players and Service Manager, and between Service Manager and Network Controller is shown in Figure 2. It shows the full cycle for Player A (both start and stop) and the actions taken when Player B starts while Player A is still active.

In our configuration, the player follows the target bitrate from the Service Manager. However, as a back-up players perform bandwidth estimations. In case download

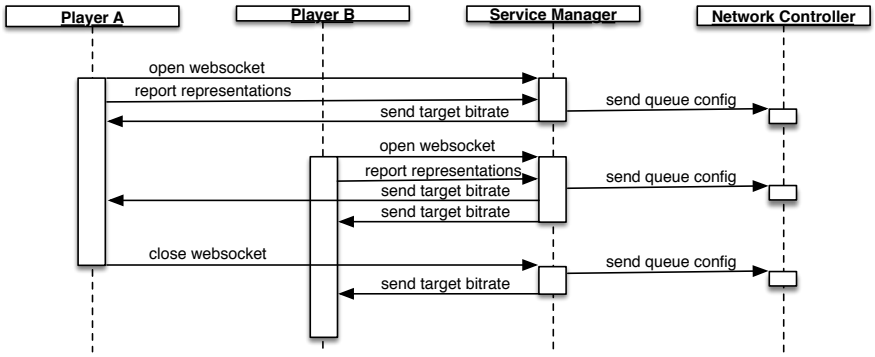


Fig. 2: Sequence of messages between DASH players, Service Manager and Network Controller

speeds are too low, it might jeopardize the stream continuity. Our DASH player then decides to ignore the target bitrate and rely on bandwidth estimations for adaptation. We also maintain a threshold on the buffer level (10 seconds in our implementation) to switch between bandwidth estimations and target bitrate. The player has the freedom to select lower bitrate segments during the initial buffering phase to start playing sooner. The advantage of this approach is that it allows players to make informed adaptation decisions while maintaining their flexibility and robustness to changing network conditions.

4. EXPERIMENTAL EVALUATION

We benchmark the performance of our system in our Wi-Fi testbed. This section describes the implementation of our system, the testbed, and the design of the experiments.

4.1. Wi-Fi Testbed

We implement our system using a Raspberry Pi 2B[2]. The Raspberry Pi acts as an OpenFlow-enabled switch, which we create using Open vSwitch (version 2.3.0)[3]. Open vSwitch is widely used software and offers a kernel based switch implementation. This switch allows for relatively high switching performance compared to non kernel-based implementations. We found that the switching performance while using Open vSwitch was not limited by the CPU power of the Raspberry Pi, only by the network interface. The CPU load remains under 10%, even when fully loading the network interfaces. We use Linux `tc` to implement our dynamic traffic control mechanism. Because OpenFlow does not provide queue configuration messages, we added an additional interface to the switch to configure `tc`.

The Raspberry Pi also acts as the Wi-Fi access point. A Wi-Fi network is an interesting environment for our experiments because users often encounter the Wi-Fi network as the bottleneck. Small variations in throughput add an extra layer of difficulty for the DASH players and our system. Nevertheless, we designed our system to be generic, and it also works on wired network connections.

We create the Wi-Fi access point using a USB Wi-Fi dongle (TP-Link TL-WDN4200) in combination with the `hostapd` software (version 2.5.0)[4]. The access point operates

[2]https://cdn-shop.adafruit.com/pdfs/raspberrypi2modelb.pdf (last accessed January 17, 2017)
[3]http://openvswitch.org (last accessed January 17, 2017)
[4]http://w1.fi/hostapd/ (last accessed January 17, 2017)

in the 5 GHz band on channel 44 (20 MHz wide channel). The transmission power of the Wi-Fi dongle is 20 dBm, in compliance with Dutch regulations.

The Network Controller is implemented using POX (version 0.2.0)[5]. POX is a Python-based SDN controller platform that enables quick prototyping of network controllers. The southbound interface talks OpenFlow 1.0 to the Open vSwitch software switch. In addition to OpenFlow, we implement an additional interface for configuring traffic control. The Network Controller offers an interface to the Service Manager on the northbound. The Service Manager uses this interface to apply the DASH specific bandwidth sharing policy. The Network Controller and Service Manager can run on different hosts. However, in our experiments both processes run on the Raspberry Pi.

The DASH player used in our experiments is browser-based and implemented using the DASH.js player (version 2.2.0). We extend it by adding the AssistanceController and AssistedAbrRule components to it. The AssistanceController implements the communication between the DASH player and the Service Manager. The AssistedAbrRule is the actual adaptation algorithm that follows the target bitrate from the Service Manager when it can. Splitting the implementation allows us to maintain a communication channel to the Service Manager while still relying on the built-in throughput-based and BOLA algorithm.

The DASH players run in the Chrome browser on current MacBook Pro and MacBook Air machines. The laptops connect to the Wi-Fi access point created by the Raspberry Pi. The Raspberry Pi stands in the middle of an office space with the laptops distributed around it.

4.2. Experimental Design

We evaluate the scenario of four DASH players sharing a Wi-Fi network connection. In this scenario, the Wi-Fi network forms the bottleneck link with a capacity of 25 Mbit/s. We replicate a setting of four family members sharing the household Wi-Fi where each member watches video on their own device. In the experiments with background traffic we add a fifth node to the network. This node generates background traffic in the form of an `iperf`[6] download.

The video nodes stream a ten-minute clip from the movie Sintel. The nodes start approximately one minute after each other. The video stream provides 12 different representations. The bitrates range from 300 Kbit/s at 240p to 10 Mbit/s at 1440p (2K)[7]. The DASH manifest implements the MPEG-DASH Live Profile [ISO/IEC 23009-1 2014] and is compatible with the guidelines from the DASH-IF.

In the evaluations we focus on the bitrate of the stream and the changes in bitrate that occur over time. These two factors play the biggest role in the user's QoE.The evaluation covers four experiments:

— **Impact of adaptation algorithms and target bitrate signaling (Experiment 1):** We evaluate the performance of the built-in adaptation algorithms (throughput-based adaptation and the BOLA algorithm) and our assisted algorithm. We assess the algorithms in an environment with and without background traffic.
— **Impact of dynamic traffic control (Experiment 2):** We focus on the dynamic traffic control mechanism. We implemented a total of four traffic control configurations. The first two options are using a service queue that puts the DASH players in a single queue, and using client queues where each DASH player is in its own queue.

[5]https://github.com/noxrepo/pox (last accessed January 17, 2017)
[6]https://iperf.fr (last accessed January 17, 2017)
[7]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

Both types of queues can work as rate limiting queue (i.e. capping the throughput) or as minimum rate queue (i.e. provide a throughput guarantee without a cap). In this experiment, we focus on how dynamic traffic control helps the built-in adaptation algorithms. We will not only investigate if adding traffic control counters streaming at low bitrate due to background traffic, but we will also look into the stability of the streams.

— **Combination of target bitrate signaling and dynamic traffic control (Experiment 3):** We combine target bitrate signaling with dynamic traffic control. In this part, we investigate if combining the two assistance mechanisms outperforms setups with only one of the two mechanisms enabled.

— **Differentiation of device types (Experiment 4):** In addition to the equal bandwidth sharing policy, we also implement a device-aware policy. In the last experiment, we demonstrate the effectiveness of our system and show that it can give assistance tailored to the device. For this analysis, we use the same devices, but we trick the system in thinking that they are different (i.e. the DASH players report a different device type to the Service Manager when reporting the representations in the manifest).

5. RESULTS

In this section, we evaluate the performance of our system and investigate how target bitrate signaling and dynamic traffic control contribute to the video quality of the stream. We first evaluate each mechanism individually, then combined. We end the evaluation with a demonstration of a sharing policy that assigns different bitrates to different devices.

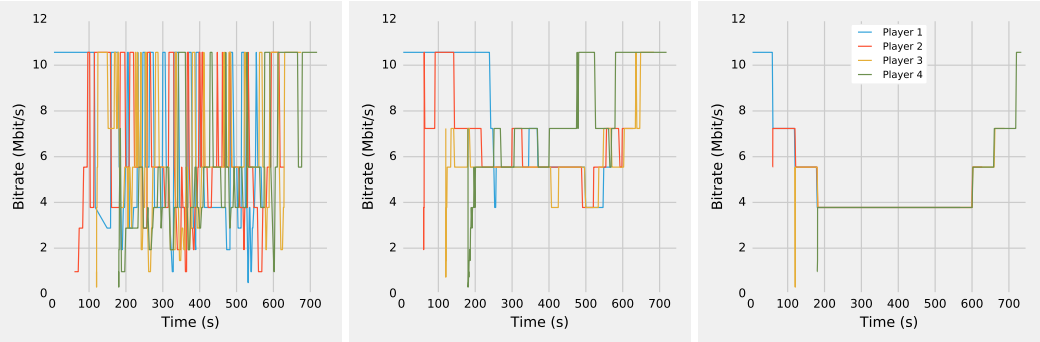
5.1. Impact of Adaptation Algorithms and Target Bitrate Signaling

Throughput-based algorithms have difficulties in selecting a stable bitrate when multiple DASH players share a bottleneck link [Akhshabi et al. 2011][Akhshabi et al. 2012]. We confirm this problem using the throughput-baed adaptation algorithm in the DASH.js player. Even though it is known to exhibit instability, we include this adaptation algorithm because it is the default algorithm in the DASH.js player. Figure 3a shows the frequent fluctuations in video bitrate when multiple players are active at the same time.

The DASH.js player also implements the BOLA algorithm [Spiteri et al. 2016]. BOLA uses the buffer level as an indication for adaptation. Figure 3b shows that BOLA significantly reduces the number of changes in video quality compared to the throughput-based algorithm. Nevertheless, it exhibits unfairness in the period from $t = 400$ until $t = 550$. One player increases its video bitrate, but the other players adapt to lower bitrates. Although BOLA proves to outperform the throughput-based algorithm in terms of number of switches, it is not immune to unfairness.

The run with our assisted adaptation algorithm shows to perform best regarding stability and unfairness. The DASH players strictly follow the target bitrate from the Service Manager. Figure 3c shows that assisted adaptation reduces the bitrate switches to the moments where DASH players start and stop. It also points out that the response to a starting player is much quicker and that active players release bandwidth to the new player. The video bitrate is lower compared to the BOLA algorithm (4.739 Kbit/s versus 6.307 Kbit/s). The lower video bitrate is an effect of the sharing policy that strictly assigns the same bitrate to each player. However, if we would assign a higher bitrate to one or two players, this effect would disappear.

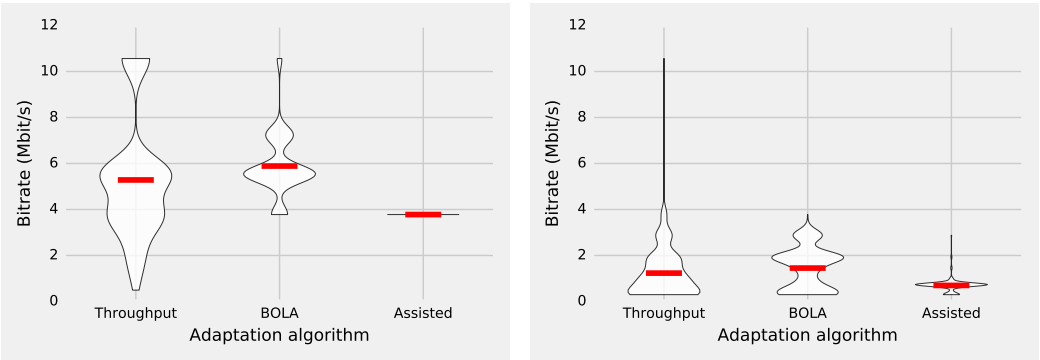
Adding adaptation assistance from the Service Manager improves the stability of the streams. However, without traffic control, the Service Manager cannot guarantee a sufficient throughput for the DASH players. Figure 4 shows the distribution of



(a) Throughput (b) BOLA (c) Assisted adaptation

Fig. 3: Video bitrates for four competing DASH players in a network without background traffic

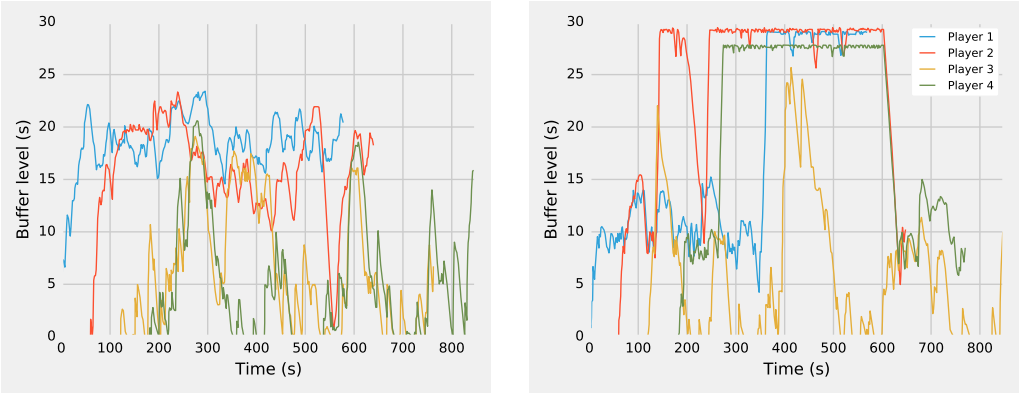
video bitrates. Figure 4a shows the experiments without background traffic. Figure 4b shows the experiments with background traffic(i.e. an `iperf` download), where the video bitrates drop significantly, also for the assisted adaptation algorithm. In the experiments with background traffic, the assisted algorithm did not adapt to the target bitrate. The bandwidth and buffer estimations detected that the throughput is too low. This triggered the fallback mode in the DASH players, resulting in a lower video bitrate. However, forcing the target bitrate would have led to more (severe) interruptions in the video stream.



(a) Only DASH players (b) DASH players and background traffic

Fig. 4: Distribution of video bitrates. Whiskers show the mean video bitrate.

Figure 5 displays the buffer fill levels for each player, comparing BOLA and the assisted adaptation algorithm. We estimate the buffer levels from the traces based on the interval between two segment requests (i.e. the buffer grows when the interval between two segments is less than the segment size; the buffer level shrinks when the interval between two segments is larger than the segment size). Figure 5 shows that BOLA players 3-4, and assisted player 3, have difficulties in maintaining a healthy buffer level. The buffers grow much slower and are empty several times.



(a) BOLA algorithm (b) Assisted adaptation algorithm

Fig. 5: Buffer levels of the DASH players, experiments with background traffic

The low buffer levels are the result of the competition for bandwidth between DASH players and the background TCP flow, where the background flow keeps to much of the available bandwidth. Figure 6 shows that the performance of the background flow is only marginally affected by the DASH traffic. Given the 25 Mbit/s capacity of the network, each TCP flow (DASH.js uses HTTP/1.1 and keeps the connection to the server open) should have a throughput of 5 Mbit/s. However, the background node uses a manifold of that and the DASH players stream below this rate. The adaptation algorithm does not have an mean bitrate. The DASH players stream on average at 1157 Kbit/s ($\sigma = 1341$ Kbit/s) for the throughput-based algorithm, 1434 Kbit/s ($\sigma = 955$ Kbit/s) for BOLA, and 1305 Kbit/s ($\sigma = 1433$ Kbit/s) for the assisted adaptation algorithm.

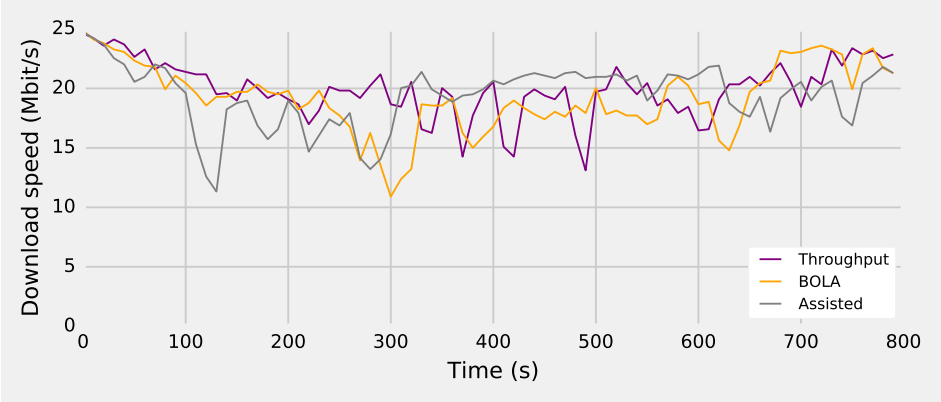


Fig. 6: Throughput for background traffic

From the results, we can conclude that the adaptation algorithm is essential for creating a stable streaming experience and to divide the bandwidth evenly among the players. However, adaptation algorithms react to the download speeds of the video

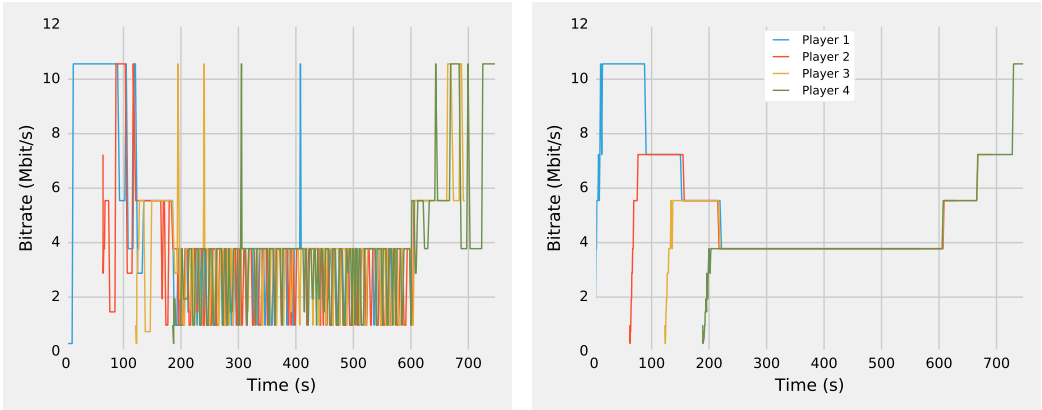
segments. They do this directly by using throughput measurements or indirect by observing the buffer level. If the download speeds are not sufficient, DASH players adapt to a lower video bitrate. When we add background traffic to the network, download speeds are no longer sufficient. Then, improving client-side adaptation algorithms or using target bitrate signaling is not enough. In the next part of this evaluation, we will show how traffic control can solve these issues and realize high-quality streaming.

5.2. Impact of Dynamic Traffic Control

In this part of the evaluation, we investigate the impact of our dynamic traffic control mechanism. The DASH players contact the Service Manager and thus benefit from traffic control, but they still rely on the built-in adaptation algorithms (default throughput-based algorithm and BOLA). First, we evaluate the impact of traffic control on the streaming bitrate. Second, we look into differences between using a service (i.e. DASH) queue and using client queues. Third, we investigate the impact of rate limiting queues and minimum rate queues. In any of the configurations, the Service Manager allocates an equal bitrate for each DASH player.

As shown in the previous section, the problem with DASH traffic in a network with significant background traffic is that DASH players cannot reach sufficient down speeds. The on/off traffic pattern in DASH traffic causes a reset of the TCP window. Segment download always start with the minimal TCP window. The TCP window size will only slowly increase because there is already a high load in the network. TCP will not have enough time to obtain a fair share since the video segments are short (e.g. between two and ten seconds). We can mitigate this issue and increase the download speeds for DASH, by creating different queues for different flows. Figure 7 shows the bitrates for the throughput-based algorithm and BOLA. The players with the throughput-based algorithm can stream at the preferred bitrate. However, they also show a considerable number of quality switches. BOLA shows that it can provide a stable stream following the queues set by the Service Manager.

Figure 7 shows the video bitrates in a network with background traffic and the most strict traffic control configuration. For these experiments, the Service Manager creates one queue per DASH player and one queue for background traffic. Furthermore, each

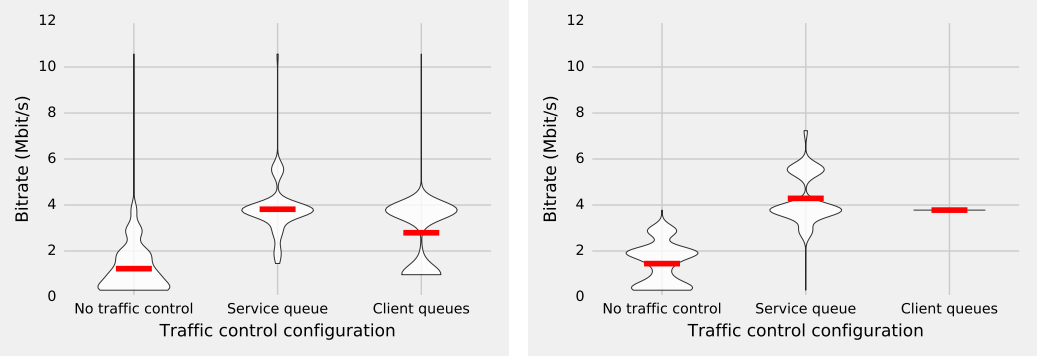


(a) Throughput-based (b) BOLA

Fig. 7: Video bitrates for four DASH players in a network with background traffic and client queues with a rate limit

queue has a rate limit and completely separates each flow from each other. In our SDN switch we use Linux `tc` with hierarchical token buckets (HTB). This allows us to create large queueing configurations. Other SDN switches may not offer this functionality and put a limitation on the number of queues.

We also evaluate a setup with a single queue for DASH traffic. Figure 8 shows the comparison of the video bitrates using no traffic control, a single DASH service queue, and client queues. For completeness, we also include the experiment where we disabled traffic control. The video bitrate is slightly higher when employing the service queue: 4768 kbit/s (σ = 2442 kbit/s) compared to 3578 kbit/s (σ = 2507 kbit/s) for the throughput-based algorithm, and 5191 kbit/s (σ = 2329 kbit/s) versus 4570 kbit/s (σ = 1837 kbit/s) for BOLA. Both adaptation algorithms increase the video bitrate when possible (i.e. when they either reach sufficient download speeds or have sufficient buffer). If we combine the four players into a single queue, then some players occasionally adapt to a higher bitrate (compared to the target bitrate). These temporary upgrades increase the mean bitrate. In contrast, the throughput-based algorithm shows a large number of requests for low bitrate video segments when using client queues. In this case, the players were not confident enough and did not fully utilize the whole queue. This has the effect of a lower mean bitrate for this experiment.



(a) Throughput-based (b) BOLA

Fig. 8: Distribution of video bitrates in a network with background traffic and traffic control. The whiskers denote the mean video bitrate.

For the BOLA algorithm, in stream variability is different when using a service queue compared to using client queues. Using the service queue, BOLA selects some of the segments in higher bitrates (relative to the target bitrate). However, other segments, also from other players, had to be requested in a lower bitrate as compensation. Regarding stream stability, using client queues increases performance. BOLA with client queues reduces the total number of switches by 52%, from 122 to 59 switches. BOLA combined with client queues eliminates most of the quality switches. This is also visible in Figure 7b.

Regarding background traffic we do not observe a difference in throughput for different queueing configuration. The queuing configuration does not affect the size of the background queue (i.e. the service queue has the same size as the combined client queues) Figure 9 illustrates how throughput of background traffic is adjusted and how network bandwidth is now available for DASH streaming. A starting player causes a decrease in throughput for the background flow, except when starting the fourth

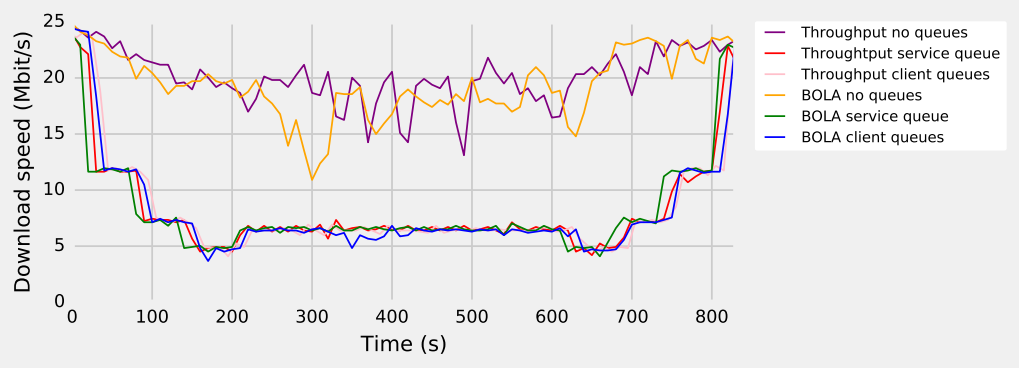


Fig. 9: Throughput for background traffic when using a service queue and client queues

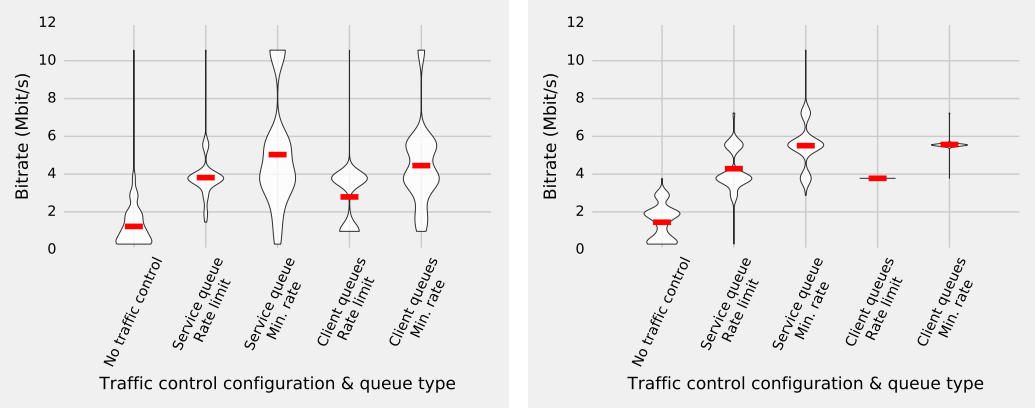
player at $t = 200$. The sudden increase in throughput is an effect of the policy that divides bandwidth among players. The sum of the target bitrates cannot exceed the reserved bandwidth for DASH. In our policy, this limitation is 20 Mbit/s. Three players at 6.653 Kbit/s use more bandwidth than four players at 4.535 Kbit/s (19.959 Kbit/s versus 18.140 Kbit/s). This behavior is inevitable and is the result of the limited number of available bitrates in the DASH manifest. In this experiment, adding the fourth player causes a decrease of only one quality level.

In this article, we also look into differences between rate limiting and minimum rate queues. We implemented both, queues that limit the throughput for certain packets and queues that guarantee a minimum throughput given their is enough traffic. Figure 10 compares video bitrates for these two types of queues. The results show that using minimum rate queues increases the mean video bitrate by at least 1 Mbit/s. This result is expected and comes from the greediness of the DASH players (i.e. they will increase the video bitrate when possible). In our system, we only provide a throughput guarantee for DASH traffic, but not for background traffic. Due to the implementation of hierarchical token buckets (HTB) in Linux `tc`, DASH players have an advantage over background traffic.

Users typically prefer an improvement in video bitrate. However, we can argue that the increase in bitrate when using minimum rate queues is unwanted behavior in this scenario. We assume that the Service Manager makes an optimal division of the available bandwidth, as that is its main purpose. The DASH players do not have to try to increase their video quality. However, we observe that for both algorithms the DASH players sometimes select higher quality segments. This causes a larger variability in video bitrate and more quality switches, eventually resulting in a lower QoE for the user.

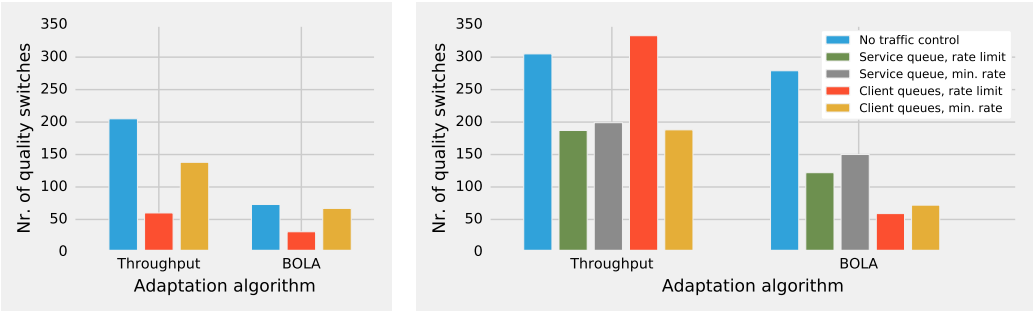
Figure 10 illustrates higher variability for minimum rate queues and shows that the distribution of bitrates in which DASH players request video segments is spread more over available bitrates. This effect is the strongest for the throughput-based algorithm, but it is also observed for BOLA.

In general, using minimum rate queues also increases the number of quality switches. Figure 11 displays the total number of quality switches (i.e. the sum of the number of switches from the four players) for the experiments without background traffic (a) and with background traffic (b). In the experiment without background traffic, we only evaluate client queues. A single DASH queue does not make sense because



(a) Default algorithm (b) BOLA algorithm

Fig. 10: A comparison of the distribution for video bitrate in a network with background traffic: rate limiting queue versus minimum rate queues



(a) No background traffic (b) Background traffic

Fig. 11: Comparison of the number of switches when using rate limiting queues and minimum rate queues

the size of the queue would then be equal to the capacity of the network. Figure 11a shows that applying queues with a rate limit reduces the number of quality switches. For the throughput-based algorithm, employing traffic control reduces the number of switches by 71%, from 205 to 60. For BOLA we observe a reduction of 58%, from 73 to 31 switches. However, minimum rate queues only reduce the number of switches by 32% for the throughput-based algorithm and by 8% for BOLA.

For the experiments with background traffic, we observe a similar effect. However, differences between queuing configurations are smaller. The exception is the run with the throughput-based algorithm in combination with rate limiting client queues. In this setting, we see that DASH players often switch to a low video bitrate. The throughput-based adaptation algorithm is too aggressive and reacts too fast to small changes in throughput. This causes over 330 quality fluctuations. Overall, BOLA in combination with client queues performs best regarding the number of switches.

A third consequence of using minimum rate queues is lower throughput for background traffic. The background queue does not have a minimum rate guarantee and

gives DASH players an advantage over background traffic. Figure 12 shows the decline in throughput when DASH players get active. If we compare download speeds of background traffic for both rate limiting queues and minimum rate queues, we can observe a decrease of around 80%. For example, the average download speeds drop from 6.506 Kbit/s (σ = 2.050 Kbit/s) to 1.250 Kbit/s (σ = 1.952 Kbit/s) given the BOLA algorithm with a single service queue (measured in the interval where all four DASH players are active, $t \in [200, 600]$).

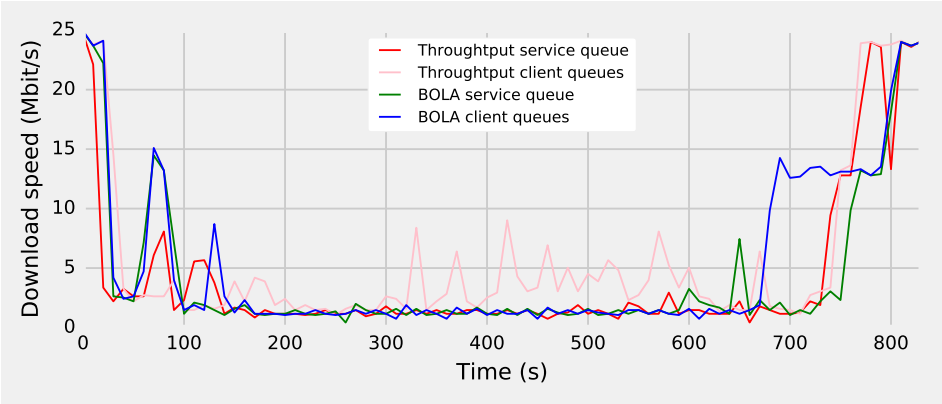
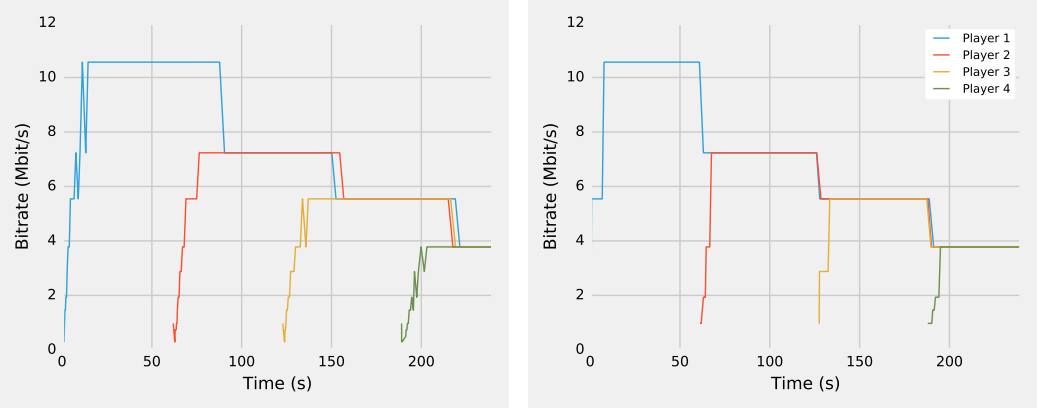


Fig. 12: Throughput for background traffic when using minimum rate queues

Based on the results in this part of the evaluation, we conclude that dynamic traffic control is essential for realizing high-quality DASH streaming. Given our goal is to stream close to the target bitrate from the Service Manager, using client queues with a rate limit provides the best performance. Nevertheless, using a single service queue is still useful, and it protects DASH players from background traffic taking over. We do not recommend to configure minimum rate queues with traditional adaptation algorithms. These algorithms are greedy and attempt to upgrade the video quality at the cost of instability and poor performance for background traffic.

5.3. Combination of Target Bitrate Signaling and Dynamic Traffic Control

In the first two sets of experiments, we showed that target bitrate signaling and dynamic traffic control both contribute to the streaming performance. However, we also pointed out that they have their shortcomings. In this part of the evaluation, we combine target bitrate signaling with dynamic traffic control. If we assume that the link that we manage with our system is the bottleneck, we can reduce the uncertainty for the player via target bitrate signaling and ensure sufficient throughput with our traffic control mechanism. By combining the two mechanisms, DASH players adapt quicker to the target bitrate and react faster to other DASH players starting and stopping. Figure 13 shows a comparison of the video bitrate for the BOLA algorithm and our assisted adaptation algorithm. In this experiment, we use client queues with a rate limit. Both BOLA and the assisted adaptation algorithm follow the target bitrate from the Service Manager. BOLA uses buffer estimations to find out the target bitrate where our assisted adaptation algorithm gets the bitrate sent directly. If we compare the time that a player needs before it streams at the target bitrate, then our assisted



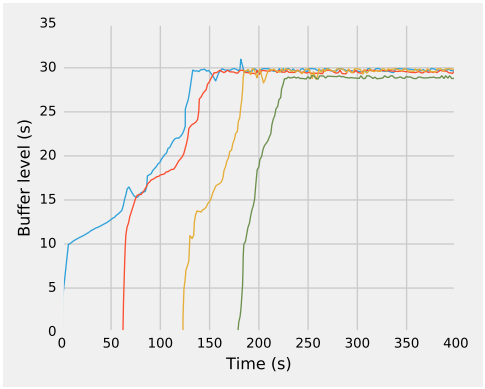
(a) BOLA algorithm (b) Assisted adaptation

Fig. 13: Video bitrates for DASH players in a network with background traffic and client rate limiting queues

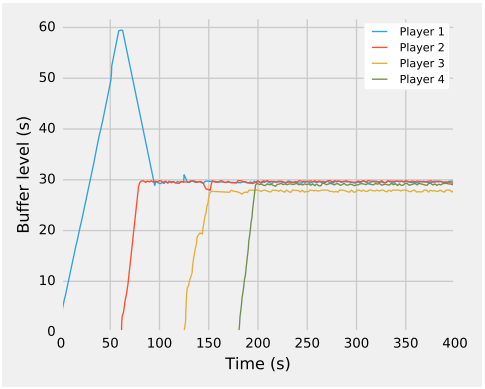
adaptation mechanism adapts 51% faster than BOLA. We observed an average reduction from 16.75 segments (33.5 seconds) for BOLA to 8.25 segments (16.5 seconds) for assisted adaptation.

Combining our assisted adaptation algorithm with other queuing configurations results in similar performance when looking at video bitrate and stability. However, Figure 14 shows that different queueing configurations produce different buffer levels. The buffers fill slower when using queues with a rate limitation. Rate limiting queues only offer little room for the DASH player to grow their buffer, especially because the throughput of traffic in the queue cannot exceed the target bitrate. Queues with a minimum rate guarantee provide a solution for this problem and allow DASH players to increase their download speeds temporarily. Minimum rate queues shorten the time that DASH players need to reach a full buffer of 30 seconds. A service queue with minimum rate reduces the buffering time from 91 to 24 seconds. For client queues, the reduction is from 167 to 21 seconds on average. The peak in buffer level for the first DASH player in Figures 14b and 14d is the result of a bug in the scheduler of the DASH.js player. We configured the players to maintain a 30 second buffer. However, the DASH.js player temporarily ignored this limit, allowing the buffer to grow up to 60 seconds. Nevertheless, this bug did not affect the streaming bitrate for the other players.

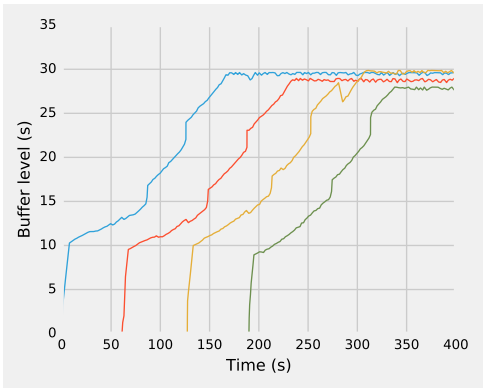
The problem with client-side adaptation algorithms, such as the throughput-based and BOLA algorithm in the DASH.js player, is that they are greedy and try to increase the video bitrate at all times. This behavior causes problems with the throughput for background traffic when using queues with a minimum rate. Target bitrate signaling not only prevents DASH players from selecting a too low bitrate, but it also restricts players from picking bitrates that are higher than the target bitrate. This approach has the advantage that it does less harm to background flow download speeds. Figure 15 shows that assisted adaptation has a smaller impact on the throughput for background traffic compared to client-side adaptation algorithms. Starting and stopping cause fluctuations in throughput, but minimum rate queues induce an overall increase in download speed.



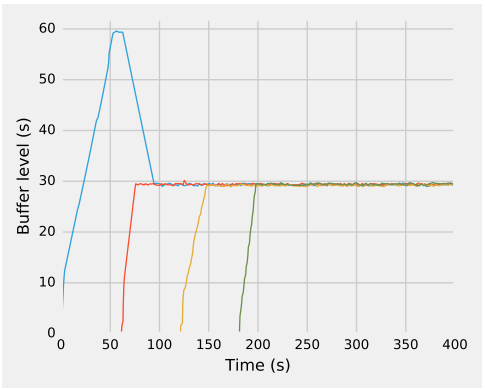
(a) Service queue, rate limit



(b) Service queue, min. rate



(c) Client queues, rate limit



(d) Client queues, min. rate

Fig. 14: Buffer levels when using assisted adaptation and different queuing configurations

Fig. 15: Throughput for background traffic when using our assisted adaptation algorithm with different queuing configurations

Based on these results, we conclude that combining target bitrate signaling with dynamic traffic control produces a high-quality video stream with excellent stability. Target bitrate signaling enables DASH players to adapt the video bitrate quickly when a new player starts, or an active player stops. It gives network administrators flexibility to configure different traffic control mechanisms.

5.4. Differentiation of Device Types

In the previous experiments, we used a simple sharing policy that equally divides 20 Mbit/s over the active DASH players. Until now, we made the assumption that each client is equal. However, device characteristics and user needs differ in practice. In the following experiments, DASH players communicate their type (e.g. TV set-top box, tablet computer) to the Service Manager. We configured two players to report themselves as a smartphone (small screen), one as a laptop, and one as a TV. The Service Manager assigns representations up to 360p for smartphones, up to 1080p for the laptop, and up to 1440p for the TV. The players start in the random order: smartphone, smartphone, laptop, TV.

Table I lists the mean bitrates for the four implementations that execute the device aware sharing policy. The first implementation combines the players into a single service queue and requires that the users manually configure a maximum bitrate in the players. In the second and third implementation, the maximum bitrate is enforced using per client queues with a maximum rate. The throughput-based algorithm and BOLA algorithm in the DASH.js player handle video bitrate adaptation in these implementations. The last implementation is the combination of the assisted adaptation algorithm and client queues with a rate limit.

Table I: Average video bitrates per device when using a device-aware policy

Adaptation algorithm	**Phone 1 Kbit/s (σ)**	**Phone 2 Kbit/s (σ)**	**Laptop Kbit/s (σ)**	**TV Kbit/s (σ)**
Throughput algorithm with fixed maximum	711 (89)	696 (121)	3.401 (995)	8.365 (3.089)
Throughput algorithm	717 (122)	712 (177)	2.953 (1.338)	10.101 (1.813)
BOLA algorithm	725 (51)	880 (140)	3.676 (579)	10.077 (1.916)
Assisted adaptation	726 (55)	728 (58)	3.707 (425)	10.391 (1.216)
Target bitrate	**732**	**732**.	**3.779**	**10.563**

The results show that all implementations result in higher bitrates for the laptop and TV. However, the mean bitrate for the TV is too low when using a service queue and manually configured players. Frequent quality fluctuations between the target bitrate and a lower bitrate cause the mean video bitrate to be lower. Using the throughput-based algorithm with client queues shows a similar effect. BOLA and assisted adaptation result in a mean bitrate that is close to the target bitrates.

Table II lists the number of quality switches for each player. The number of switches shows similar effects to results in the previous evaluation part. The BOLA algorithm produces less switches compared to the throughput-based implementations. The assisted adaptation algorithm makes players adapt quicker to the target bitrate and increases the stability. Results show that our system is effective in enforcing policies that go beyond equally sharing the network bandwidth. The advantage of assisted

adaptation is that sharing policies only have to be configured in a single place (i.e. the Service Manager). The Service Manager can potentially use a different policy at different moments based on which clients are active.

Table II: Number of quality switches per device when using a device-aware policy

Adaptation algorithm	**Phone 1 # switches**	**Phone 2 # switches**	**Laptop # switches**	**TV # switches**
Throughput algorithm with fixed maximum	4	7	12	46
Throughput algorithm	16	31	79	8
BOLA algorithm	5	11	7	12
Assisted adaptation	2	2	3	3

In our implementation, clients send a special message informing the Service Manager about the device form factor. This implementation does not require the Service Manager to know up front which devices are going to be active. However, it provides little means against cheating devices. Depending on the needs of the network administrator, we could potentially implement solutions like whitelisting, where only trusted devices can send their device type and untrusted devices can only stream up to a fixed limit.

6. CONCLUSION

Video streaming over the Internet is a popular application that accounts for a large share of todays Internet traffic. Many content providers use dynamic adaptive streaming over HTTP as technology for delivering video content. The simplicity of the protocol, the underlying use of HTTP for transport, and its scalability using content deliver networks make it an attractive technology for content providers. The downside of using HTTP in combination with the bursty on/off traffic patterns created by DASH, is that it underperforms when multiple DASH players share a network link, and in networks with heavy background traffic. Instead of fairly sharing network resources, downloading the DASH video segments becomes a competition for bandwidth. In this article we demonstrated that DASH players are the losing party.

The presented system is designed to reduce the competition for bandwidth. Doing this increases the video quality and reduces the instability of the stream. We leverage the novel networking paradigm of software defined networking (SDN) to reduce the effect of the protocols on streaming performance, while keeping DASH protocol stack benefits intact. We have shown to overcome the drawbacks of using DASH by adding a simple interaction mechanism between DASH players and an in-network component that we call the Service Manager.

In our system, we offer DASH players two mechanisms for improving their streaming performance: target bitrate signaling and dynamic traffic control. Target bitrate signaling involves sending the DASH players a bitrate for which the Service Manager thinks it is a good bitrate, considering the current network activity. Given that the Service Manager has a better overview of the network activity compared to a DASH player, it is better capable of fairly dividing the available bandwidth among clients. The second mechanism, dynamic traffic control, ensures that DASH players are able

to reach sufficient download speeds by setting up traffic queues in the network matching the DASH demand.

We demonstrate that our system can provide a highly stable and high quality video streaming experience through evaluations in our Wi-Fi testbed using real DASH players that are extended to support adaptation assistance. We showed that both adaptation assistance mechanisms increase streaming performance. Target bitrate signaling can be used to increase stability when multiple DASH players share a connection. Dynamic traffic control shows to increase the video bitrate and stream stability in networks with heavy background traffic. From results obtained in this study, we can conclude, that separating DASH players in different rate limited queues gives the best performance when only dynamic traffic control is enabled. Using a single service queue or minimum rate queues shows to increase the variability of video quality in the stream and increases the number of quality switches. Furthermore, we showed that the greedy nature of current DASH adaptation algorithms gives problems when using minimum rate queues. Only by combining the two assistance mechanisms, we can deliver an optimal video streaming experience, while also delivering acceptable download speeds for background traffic. Using target bitrate signaling gives more freedom in using different types of traffic control, potentially increasing the number of network switches that could used.

By implementing a policy that assigns different bitrates to different devices – for instance because they have different capabilities – we demonstrate that our system is an effective tool for configuring how the network bandwidth should be shared among devices. Configuring bandwidth sharing rules in the Service Manager has the advantage of a central location where bandwidth sharing is configured, compared to configuring each individual DASH player.

In future work we will focus on the improvement and evaluation of scalability of our system. We built our system using scalable technologies (SDN and DASH). Both technologies are known to be deployed in large networks serving a large number of users. For our system, two issues have to be addressed in order to support large networks: computing resources for the Network Controller and Service Manager, and the traffic control configuration. For every DASH player which receives adaptation support, a connection to the Service Manager is maintained. The Service Manager needs to be able to handle a large number of connections. Additionally, for each update (i.e. starting or stopping players) the bandwidth sharing policy is applied. This should be fast (e.g. new players could quicker start streaming) and the Service Manager host machine should have sufficient computing power for the complexity of the sharing policy. The Service Manager was located at the Wi-Fi access point in the experiments in this article. It needs to be offloaded to another server or the cloud to support larger numbers of DASH players. Our design supports flexible locating of the Service Manager (and Network Controller). However, the effect of latency between DASH players and Service Manager should be evaluated when moving the Service Manager further away from the players. With regards to traffic control, we show that there are no differences between service and client queues when target bitrate signaling is used. However, maintaining a large number of client queues may not be feasible. A single service queue with a large number of DASH players may be less effective. In future experiments, we will evaluate the use of multiple service queues to address this issue.

Furthermore, we will look into generalization of our system, to create a networking platform where applications work together with the network through a Service Manager, with the goal to tailor network performance to the demands, and improve the user experience. This will require the development of bandwidth sharing policies, that know how to divide DASH traffic, taking the characteristics of the user and its devices into account, as well as the characteristics of other non-DASH services.

Acknowledgements

We would like to thank Britta Meixner for comments that greatly improved the readability of the manuscript.

REFERENCES

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. DOI:http://dx.doi.org/10.1145/2229087.2229092

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. DOI:http://dx.doi.org/10.1145/1943552.1943574

Niels Bouten, Jeroen Famaey, Steven Latré, Rafael Huysegems, B D Vleeschauwer, W V Leekwijck, and F D Turck. 2012. 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, 336–342.

Nicola Cranley, Philip Perry, and Liam Murphy. 2006. User perception of adapting video quality. *International Journal of Human-Computer Studies* 64, 8 (2006), 637–647. DOI:http://dx.doi.org/10.1016/j.ijhcs.2005.12.002

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. DOI:http://dx.doi.org/10.1145/2428556.2428577

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*. ACM, New York, NY, USA, 21–26. DOI:http://dx.doi.org/10.1145/2229087.2229094

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. DOI:http://dx.doi.org/10.1145/2491172.2491181

Roelof Hamberg and Huib de Ridder. 1999. Time-varying Image Quality: Modeling the Relation between Instantaneous and Overall Quality. *SMPTE Journal* 108, 11 (1999), 802–811. DOI:http://dx.doi.org/10.5594/J04337

Tobias Hoßfeld, 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.

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. DOI:http://dx.doi.org/10.1145/2155555.2155557

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. DOI:http://dx.doi.org/10.1145/2398776.2398800

ISO/IEC 23009-1. 2014. Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats. (2014).

Dmitri Jarnikov and Tanır Özçelebi. 2011. Client intelligence for adaptive streaming solutions. *Signal Processing: Image Communication* 26, 7 (2011), 378–389.

Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. 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*. ACM Request Permissions, New York, New York, USA, 97–108. DOI:http://dx.doi.org/10.1145/2413176.2413189

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. DOI:http://dx.doi.org/10.1145/2910017.2910599

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.

Stefan Lederer. 2015. Why YouTube & Netflix use MPEG-DASH in HTML5. (Februari 2015). https://bitmovin.com/status-mpeg-dash-today-youtube-netflix-use-html5-beyond/

Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. 2011. Rate adaptation for adaptive HTTP streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 169–174.

Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (March 2008), 69–74. DOI:http://dx.doi.org/10.1145/1355734.1355746

Konstantin Miller, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. 2012. Adaptation algorithm for adaptive streaming over HTTP. In *19th International Packet Video Workshop (PV), 2012*. IEEE, 173–178.

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. DOI:http://dx.doi.org/10.1145/2818361

David C Robinson, Yves Jutras, and Viorel Craciun. 2012. Subjective Video Quality Assessment of HTTP Adaptive Streaming Technologies. *Bell Labs Technical Journal* 16, 4 (2012), 5–23. DOI:http://dx.doi.org/10.1002/bltj.20531

Sandvine, Inc. 2016. Global internet phenomena report. (2016).

I Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *Industry and Standards* (2011).

Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2016. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *CoRR* abs/1601.06748 (2016). http://arxiv.org/abs/1601.06748

E Thomas, M O van Deventer, T Stockhammer, A C Begen, and J Famaey. 2015. Enhancing MPEG dash performance via server and network assistance. In *The Best of IET and IBC*. Institution of Engineering and Technology, 48–53. DOI:http://dx.doi.org/10.1049/ibc.2015.0014