Delivering Stable High-Quality Video: An SDN Architecture with DASH Assisting Network Elements

Jan Willem Kleinrouweler[1], Sergio Cabrero[1], Pablo Cesar[1,2]

[1]CWI: Centrum Wiskunde & Informatica
Science Park 123
1089 XG Amsterdam,
The Netherlands

[2]Delft University of Technology
Mekelweg 4
2628 CD, Delft
The Netherlands

{j.w.m.kleinrouweler, cabrero, p.s.cesar}@cwi.nl

ABSTRACT

Dynamic adaptive streaming over HTTP (DASH) is a simple, but effective, technology for video streaming over the Internet. It provides adaptive streaming while being highly scalable at the side of the content providers. However, the mismatch between TCP and the adaptive bursty nature of DASH traffic results in underperformance of DASH streams in busy networks. This paper describes a networking architecture based on the Software Defined Networking (SDN) paradigm. Controllers in the network with a broad overview on the network activity provide two mechanisms for adaptation assistance: explicitly signaling target bitrates to DASH players and dynamic traffic control in the network. We evaluate how each of these mechanisms can contribute to the delivery of a stable and high quality stream. It shows that our architecture improves the quality of experience by doubling the video bitrate and reducing disturbing quality switches. As such, this paper contributes insights on how to implement DASH-aware networking that also enables internet service providers, network administrators, and end-users to configure their networks to their requirements.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; D.5.1 [**Multimedia Information System**]: Video

General Terms

Measurement, Performance, Design

Keywords

DASH, HTTP adaptive streaming, Video streaming, Wi-Fi, Performance, Architecture, Software Defined Networking

1. INTRODUCTION

Dynamic adaptive streaming over HTTP (DASH)[1] has become the most popular technology for streaming video over the Internet. The advantage of DASH is that it allows video players to adapt the video quality to the current network conditions, while the usage of HTTP enables content providers scalable distribution using content delivery networks (CDNs). However, several studies showed that DASH performs poorly in the presence of background traffic [14], and that DASH players suffer unfair sharing of the available bandwidth and instability when a bottleneck link is used by multiple DASH players [3, 2]. Besides a lower bitrate or resolution, changing the video quality too often negatively impacts the quality of experience [6, 11, 22].

Given the popularity of online streaming services, such as YouTube or Netflix, it is no longer an exception that multiple users stream videos in the same network at the same time. Although the capacity of computer networks keeps increasing, they might not be able to keep up with the growing demand for bandwidth that is caused by video streaming [5]. Bottleneck links are likely to be encountered, and can be found in different locations such as household-, campus- and corporate networks, or in Wi-Fi hotspots.

In such networks, DASH players have to compete for bandwidth with both each other, and with other traffic on the network link. Current networks offer little to no support for improving the streaming performance of DASH players. Moreover, the usage of TCP, in combination with the bursty traffic that DASH creates, makes that DASH players stream at suboptimal rates and suffer from unfairness and instability. State-of-the-art adaptation algorithms in DASH players improve the streaming performance. However, selecting a bitrate remains a difficult task for a player, because players have a limited view on other network activity and cannot make assumptions on the underlying network infrastructure.

This paper describes a DASH-aware networking architecture based on the novel software defined networking (SDN) paradigm. In SDN the data plane is separated from the control plane. A network controller, with a broad overview of the network activity and the active DASH players, assists DASH players in selecting optimal bitrates. Furthermore, it ensures that the players can reach sufficient download speeds via Quality of Service (QoS) mechanisms.

Our architecture keeps the DASH protocol stack simple and scalable, while it improves the quality of experience for the viewers. In a thorough evaluation of this architecture in a Wi-Fi hotspot, we show that the video bitrate can be doubled

[1]In general we are referring to the technique, but we are using the DASH standard [1] in our implementation and evaluations.

and the number of quality switches can be greatly reduced. Furthermore, given our architecture, internet service providers (ISPs), network administrators and potentially end-users can configure their network better. They can define how bandwidth should be shared between video- and other traffic, and how bandwidth should be shared among video players. This allows for network management with DASH-aware policies where the SDN-based network ensures successful execution of those policies. The dynamic architecture presented in this paper, in combination with the insights that we obtained in the evaluations, contribute to the understanding on how to realize DASH-aware networking and enables further research and development of service policies.

The remaining of this paper is as follows. Section 2 provides the related work. Section 3 outlines the architecture design. Section 4 describes the experimental setup. Section 5 provides a thorough evaluation of our architecture and shows the effectiveness of our solution in a Wi-Fi deployment. Section 6 concludes this paper.

2. RELATED WORK

When using DASH, a player is offered a list of different representations of the same video. The representations differ in bitrate and resolution, and a DASH player can select a representation that is optimal for the device and the current network conditions. In most DASH architectures, the mechanism for selecting the bitrate and resolution is built into the player. However, it has been shown that existing DASH-capable applications have difficulties selecting a bitrate and that they suffer instability and unfair network resource sharing when multiple DASH players share a bottleneck link [3, 2, 14]. Instability and unfairness are a result from the mismatch between the bursty ON/OFF client side behavior and the usage of the TCP transport protocol [8]. The double feedback loop, both the DASH player and TCP respond to changes in bandwidth, makes it difficult to get an accurate estimation of the available bandwidth.

As a consequence, the instability in video bitrate lowers the overall quality of experience and diminishes user engagement [6, 7, 22]. To optimize the viewers' quality of experience the video bitrate must be maximized, while keeping the number of switches in bitrate to a minimum [12]. Several solutions for improving the adaptation mechanisms in the players are proposed in the literature [16, 18, 20, 24, 15]. The downside of only improving the client-side is that it remains specific to the player, and that they cannot easily be configured by the user or a network administrator to achieve specific requirements.

As an alternative to changing the adaptation logic inside the player, network devices can be used to assist players in selecting a representation. Network devices have a broader view on the network and its capabilities. DASH players can obtain a better estimation of the available bandwidth when the network devices share their knowledge with the players. In [13] it is shown how this can be done implicitly by shaping the flows of the DASH streams. Others have routed DASH traffic through DASH-aware proxy servers that they have deployed in the network [4, 17]. In [21] a chain of proxy servers is used to address networking infrastructures with multiple bottlenecks. Georgopoulos et al. presented an OpenFlow-enabled system where an orchestrating module explicitly informs the DASH players which representation they should select [10]. In [9] SDN is leveraged to optimize DASH video streams by using in-network caches. The Server and Network Assisted DASH (SAND) proposal specifies the control messages between DANE and players, and between DANEs [23]. The architecture that we present in this paper fits that proposal as it implements a DANE, and could be extended to comply with the SAND message format once standardized.

The drawback of DANE implementations that we mentioned is that they use intrusive means to detect DASH traffic and require the manifests to be inspected by the DASH Assisting Network Element (DANE). When encryption (i.e. HTTPS) is used for the manifest, it cannot be obtained by monitoring the network, or the chain of trust between the client and server must be broken. If the manifest can be obtained, the users' privacy is potentially invaded because the DANE knows what each user is watching. Our architecture is privacy aware, in the sense that the contents of the manifest and the stream does not have to be disclosed to the DANE, because it only needs the format (e.g. encoding scheme) to function properly. Our DASH player will communicate this information directly to the DANE.

Furthermore, the implementations mentioned above are evaluated in networks with only DASH traffic, and thus without background traffic. Therefore, they cannot guarantee that the adaptation assistance mechanisms are effective in network deployments with other traffic. Our architecture does take background traffic into account, and provides assistance to the DASH players at both the application level, as well as through traffic control in the network. Moreover, our evaluations are performed both with and without background traffic.

3. ARCHITECTURE DESIGN

In this section we propose a networking architecture with specific support for DASH streaming. The architecture employs the SDN paradigm: the network data plane is separated from the network control plane. Our solution uses programmable network hardware, i.e. configurable by OpenFlow, and a network controller. On top of it, we add the Service Manager, a component that is DASH aware and implements adaptation logic. It is our goal to keep the simplicity and the robustness of DASH technology, but to defeat the negative influence of TCP and bursty traffic on the streaming performance in busy networks. For that reason, we propose an hybrid adaptation technique, in which the DASH video players are assisted by the Service Manager. The architecture is illustrated in Figure 1.

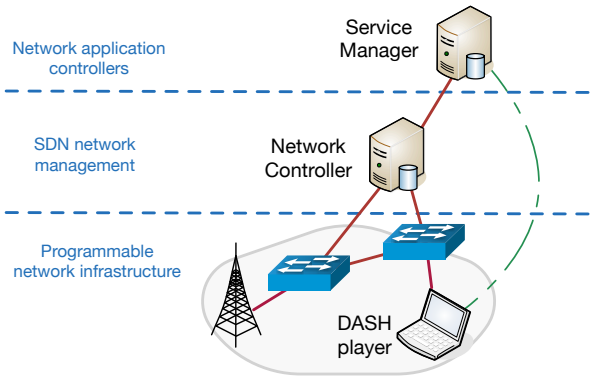

Figure 1: DASH aware SDN architecture that separates data plane (network infrastructure) from control plane (network management and application controllers). The mixed-dashed line indicates a communication channel between DASH player and Service Manager.

3.1 Programmable network hardware and Network Controller

Programable network hardware is at the basis of the architecture. In the SDN paradigm, network elements collect device status and statistics, and can execute packet forwarding following a set of rules more sophisticated than in traditional networks. A central entity, the *Network Controller*, is in charge of monitoring and configuring all network elements. Hence, the controller has a global view of the network and is capable to dynamically add and destroy forwarding rules in devices. The standard protocol that acts as interface between the Network Controller and the network hardware is OpenFlow [19]. This protocol can be used to assign traffic to specific QoS queues via the `enqueue` action in combination with a possibly detailed flow match. However, OpenFlow does not support quality of service configuration. For that reason, our network elements implement an additional QoS interface. This extra interface allows, for each queue, the specification of a minimum throughput guarantee, a maximum throughput cap, or a combination of the two. With the combination of both interfaces, the Network Controller has the ability to create queues specifying these QoS parameters.

In a future scenario, network controllers will manage the network of an organization, e.g. an Internet Service Provider, or part of it. DASH services can run over several of these networks. For that reason, we add a component, the *Service Manager*, that can interact with several network controllers and apply a specific service logic into different network domains.

3.2 Service Manager

The Service Manager oversees DASH traffic over different networks. By interacting with a Network Controller and the DASH players, it is aware of active DASH players in a network and of its available resources. Therefore, it is capable of allocating network resources and assisting players in the adaptation. For this paper, we have implemented network resource allocation and adaptation policies in the Service Manager. These policies are useful to illustrate its advantages, but they are not specific to the architecture. Instead, the Service Manager can be configured to implement the policies that better fit the specific DASH service, the device, or the networks where it is deployed.

Explicit adaptation assitance in the Service Manager follows the fairness criteria of equal bandwidth for every active DASH player and other flows in the network (i.e. background traffic). For simplicity, we specified a policy that treats each device equal, disregarding the type of device. However, it allows players to communicate a maximum bitrate that they are capable of displaying. Future policies should include device specific factors.

As a first approximation, the total bandwidth in each link is divided by the number of flows, including video streams and background traffic. If the bitrate required by a video stream is smaller than its fair share, the excess bandwidth is equally distributed among the remaining flows. We call the bandwidth assigned to each DASH player its target bitrate. The Service Manager uses the target bitrate to allocate network resources through the Network Controller. Then, the players are notified of this target bitrate, so they can use it as a reference of the maximum quality level they should request.

Dynamic quality of service is done by means of instantiating queues in network elements. We consider two queuing alternatives. The first is similar to the traditional differentiated services (DiffServ) approach. This technique creates one queue for DASH traffic and one queue for the rest, i.e. background traffic. Thus, all video streams traversing a network device share the same queue. The maximum throughput for traffic in the DASH queue is limited to the sum of the target bitrates. Similarly, the throughput of the background queue is capped to the remaining capacity of the network link. The second queuing technique goes one step further by creating a queue for each DASH player. Each queue provides a minimum rate guarantee that matches the target bitrate of the stream. We do not want to cap the throughput of each individual queue because it jeopardizes the continuity of the stream. Instead, the throughput of all DASH queues combined is limited to the sum of the target bitrates. This way, bandwidth that is not used by a DASH player, e.g. because some segments are smaller than the target bitrate as a result of VBR encoding, can be used by other players, and let them increase their buffers.

Adaptation and network resource allocation are executed every time a DASH player starts, stops or changes its status, or when there are changes in background traffic that would require the DASH players to change their bitrates. The network will be dynamically reconfigured to adapt to current conditions and, as a consequence, maximize the experience of the users by aiming for the maximum possible quality and avoiding interruptions.

3.3 DASH player

DASH players need to be extended to benefit from the hybrid adaptation mechanism proposed in our architecture. The players share the bitrates available in the requested media and receive the recommendation of a target bitrate. When a DASH client downloads the manifest file, it extracts the bitrates for audio and video from it. From those bitrates, it makes a selection of possible bitrates and those are sent to the Service Manager. Note that during this process the privacy of the user, i.e. the media requested, can be preserved: only the list of bitrates is shared. At this point, the player may have excluded some bitrates, for example because they are too high and exceed the capabilities of the device. If the conditions of the player change, a new set of bitrates can be sent to the Service Manager. Then, the old set is overwritten and the Service Manager executes again the adaptation and the network resource allocation processes. This is a useful feature if the capabilities of the device change over time. For instance, higher bitrates can be included when switching to fullscreen mode, while they can be discarded to reduce energy consumption in a mobile device. The end or pause of a stream is indicated to the Service Manager by either signaling from the player, or by closing the connection to the Service Manager.

Explicit adaptation assistance is performed by pushing a target bitrate to the client, where the target bitrate is one of the bitrates that the client has reported to the manager. It is up to the client how to handle this information. In our implementation, we provide two different modes. In *automatic* mode, the player shares information about the stream with the Service Manager, but ignores the target bitrate and relies on the built-in adaptation algorithm. In *hybrid* mode, the DASH player requests the target bitrate suggested by the Service Manager, but only when it is confident that it does not compromise the continuity of the stream. During the initial buffering, the player relies on its built-in adaptation algorithm, with the restriction that it does not select bitrates higher than the target bitrate. This allows the player to fill the buffer more quickly and thus start streaming sooner. Once the buffer reaches a certain threshold (10 seconds in our implementation), the player requests the target bitrate. This mechanism prevents the DASH clients from overtaking each other by selecting too high bitrates that would have caused congestion. Furthermore, it allows the players to stabilize the streams and rely on the Service Manager to inform them about necessary adaptation actions (i.e. adapt the bitrate when a player

enters or leaves the network). In the case where the download speeds are not sufficient to keep up with the playback, and the buffer level is critically low (we set the lower threshold to 5 seconds), the player switches back to the built-in algorithm until the buffer contains at least 10 seconds of video. This fallback mechanism is useful when the player is under worse network conditions than those foreseen by the Network Controller and the Service Manager.

4. EXPERIMENTAL EVALUATION

To show the effectiveness of our architecture we evaluate it in our Wi-Fi based implementation. In this section we describe the testbed that we use in the evaluations, followed by the evaluation design.

4.1 Wi-Fi testbed

We implemented our architecture using an OpenFlow-enabled Raspberry Pi. The Raspberry Pi (model 2) is made into an OpenFlow switch using Open vSwitch (version 2.3.0). Open vSwitch is a kernel based switch implementation with high performance compared to the reference switch implementation[2]. Using Open vSwitch, the switching performance is not limited by the CPU. The CPU load stays below 10% even when fully loading the network interfaces. Quality of service is handled by the built-in Linux traffic control. The QoS configuration is applied by the QoS daemon that translates a queueing configuration into the corresponding `tc` commands.

The Raspberry Pi will also act as Wi-Fi access point. Although our architecture is generic and also applies to wired networks, Wi-Fi networks make an interesting environment for study as they are a commonly encountered bottleneck. Furthermore, the throughput variations that result from using the Wi-Fi medium add an extra level of difficulty for DASH players. The Wi-Fi network is created using a USB Wi-Fi dongle (TP-Link TL-WDN4200) in combination with a patched[3] version of the hostapd (version 2.5.0) software. The access point is configured to operate in the 5 GHz band on channel 44 (20 MHz wide channel). The transmission power of the Wi-Fi dongle is set to 20 dBm, in compliance with Dutch regulations.

The Network Controller is implemented using POX (version 0.2.0), a Python-based SDN controller platform. On the southbound it uses the OpenFlow 1.0 protocol to configure the switch and obtain traffic statistics. Because QoS configuration is separated from OpenFlow, the Network Controller configures the QoS queues via a JSON-RPC based interface provided by the QoS daemon. On the northbound, the Network Controller itself also offers a JSON-RPC based interface. The Service Manager uses this interface to dynamically apply the DASH specific policy. In our testbed the Network Controller and Service Manager are separate processes, but run on the same host that is connected to the Raspberry Pi via the local area network.

It is required that the Service Manager can be reached from the network created by the Raspberry Pi. To ease the implementation, the Network Controller comes with an implementation of a programmable DNS server. The DNS server provides on-the-fly adding, modifying and removing of domain records that can be used for our DASH management services. This gives the DASH players an easy mechanisms for locating the in-network management services. The DASH players will communicate with the Service Manager to report their available bitrates and receive adaptation instructions. This communication is done over a websocket that is provided by the Service Manager. Websockets provide a two-way communication channel that is also available to browser based DASH implementations. We are using a browser based player that is extended from the DASH.js (version 1.5.1) reference implementation. The DASH players run on recent MacBook Pro/Air machines in the Chrome browser.

The machines are connected to the Wi-Fi access point created by the Raspberry Pi. Unless otherwise stated, the machines are spread across a single office space, relatively close to the access point that is located in the center of the room.

4.2 Experiment design

We evaluate the scenario where four nodes share a Wi-Fi network for video streaming. The Wi-Fi network is the bottleneck link and not sufficient for the four nodes to stream at the highest video quality. This represents a scenario where four family members share the household Wi-Fi and each member wants to view their own content on their own device. A fifth node that does not engage in video streaming is sometimes be added to the network. This node generates TCP background traffic, and we will investigate how background traffic affects DASH streams in our architecture.

The video nodes start streaming a ten-minute high quality clip from the movie Sintel[4], approximately at the same time. The video is prepared in 14 representations with bitrates ranging from 300 Kbit/s to 18 Mbit/s, and resolutions ranging from 240p to 2160p (4K). The video is segmented with a segment size of two seconds and described using the MPEG-DASH live profile that is compatible with the guidelines from the DASH-IF industry forum. The fifth node generates background traffic in the form of a TCP download using iperf[5].

The evaluation is split up into four experiments. In the first experiment we evaluate the mechanism where the Service Manager explicitly informs the DASH players which bitrate to select. DASH players that are set to automatic mode will ignore the adaptation assistance from the player. When put into assisted mode, the DASH player will follow the Service Manager's advice. Both modes will be evaluated in an environment with and without background traffic.

In the second experiment we inspect the impact of the quality of service mechanisms. We implemented the two queueing strategies that are mentioned in Section 3. The goal is not only to investigate if DASH streaming improves when adding QoS, but also to identify if a queueing configuration with separate queues for different players has advantages over a single video queue. The DASH players are set to automatic mode, where they do inform the Service Manager about their streams, but rely on the built-in adaptation mechanism for selecting the video representations.

The two mechanisms of explicit adaptation assistance and dynamic quality of service are both enabled in the third experiment in the evaluation. In this part we will investigate if combining the two mechanisms outperforms a setup where only one of the two mechanisms is enabled.

In the last experiment we focus on the Wi-Fi specific issue where one node is further away from the access point than the other nodes. This node will receive less throughput when the network becomes loaded. We evaluate if our architecture reduces the inequality between different nodes and how the streaming performance of the node that is further away can be improved.

In the evaluations we focus on the bitrate of the stream and the changes in bitrate that occur over time. These two factors have been identified to play the biggest role in the quality of

[2]http://archive.openflow.org/wp/2008/05/reference-implementation-v081/

[3]https://github.com/hschaa/hostapd/commit/c89daaeca4ee90c8bc158e37acb1b679c823d7ab

[4]https://durian.blender.org

[5]https://iperf.fr

experience [12]. We look at the quality level instead of the absolute bitrate of the segments. The quality levels of the segments correspond to the 14 representations in which the video is prepared. The video is prepared such that every next representation gives a similar improvement in quality. However, the step in bitrate between two different representations increases with the quality level. Comparing absolute bitrates will lead to a bias towards the higher bitrates. For the stability of the streams we will look at the number of switches, the time between two switches, and the size of the switches.

5. RESULTS

In this section we show how the adaptation assistance mechanisms of our architecture add to the quality and stability of DASH streams. We will first evaluate each of the two mechanisms separately, and then combined. We end the evaluation with a case specific to Wi-Fi networks, where one node has less network performance than the other nodes because it is located further away from the access point.

5.1 Only explicit adaptation assistance

For DASH players it is known that they have difficulties in selecting a stable bitrate when multiple DASH players share a bottleneck link. We demonstrate this effect in Figure 2, where the DASH players select representations ranging from the lower qualities up to the highest. Given the throughput of 42 Mbit/s of the Wi-Fi network, the players could have stabilized at the representation of 7.2 Mbit/s (level 10) and 10.6 Mbit/s (level 11). However, the players adopted a selfish optimization strategy, trying to increase the video quality to the highest quality level. This selfish optimization came at the cost of having to switch to the lower bitrates afterwards.

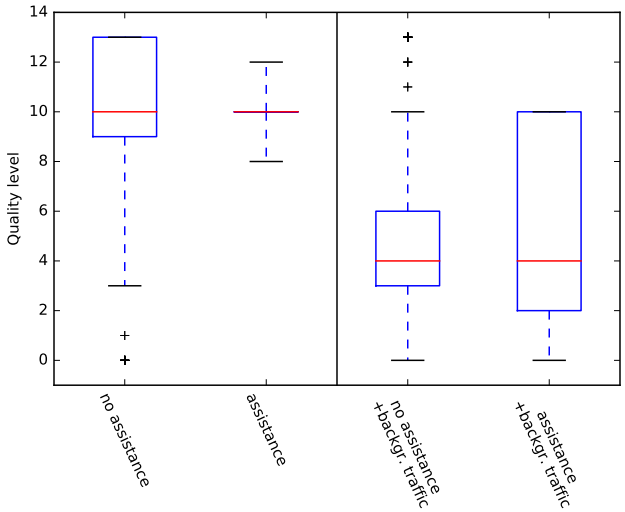

Figure 2: Video qualities for the scenarios without and with assistance (left: without background traffic, right: with background traffic)

Adding explicit adaptation assistance makes all players stream stable at the target representation that they received from the Service Manager. The total number of quality switches went down from 291 to three switches while the mean quality level remained the same ($\mu = 10.32$, $\sigma = 2.49$ without assistance, versus $\mu = 9.99$, $\sigma = 018$ with assistance).

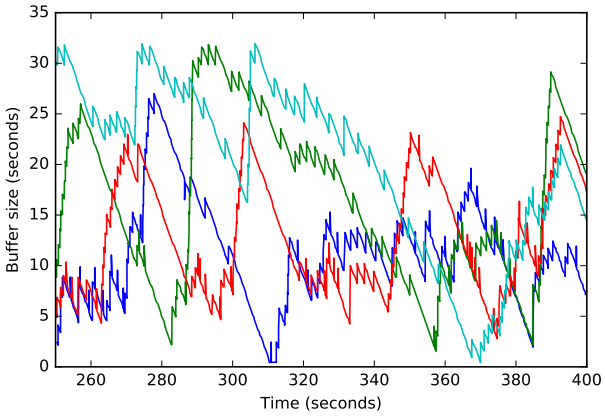

Figure 3: Buffer fill levels for the four DASH players in the assisted scenario with background traffic

Although providing the DASH players assistance from the Service Manager seems to improve the stability of DASH, there is no guarantee for the players that they can actually reach the target bitrate. Figure 2 on the right shows the impact of background traffic in the network on DASH performance. In both cases, without and with assistance, the video quality is significantly lower compared to the experiments without background traffic. Adding adaptation assistance increases the mean quality level by one ($\mu = 5.15$, $\sigma = 3.58$ with assistance, compared to $\mu = 4.30$, $\sigma = 2.87$ without assistance), however it also shows a large variation in video quality.

Even though DASH players are explicitly informed about the fair quality level (quality level 10 at 7234 kbit/s), they were not able to reach sufficient download speed to actually stream at this rate. Figure 3 shows the buffer levels for each player in the assisted scenario with background traffic. The buffer levels in the DASH players regularly drop below the threshold of five seconds, for which the players decide not to follow the assistance from the manager and switch to the built-in adaptation mechanism. As a result, result that also for the assisted scenario the video quality is lower and more unstable. Rebuffering events were very unlikely due to the fact that the DASH.js aggressively switched back to the lowest available bitrate (300 Kbit/s), and was still able to continue the stream without interruptions.

By looking at the download speeds of the background traffic node in Figure 4, we can see that the throughput of the background traffic is only marginally impacted by the active DASH players, given that the background node reaches about 42 Mbit/s of effective throughput when there are no DASH players active. Nevertheless, considering that each player and the background node all use one TCP connection (the DASH.js reference player uses HTTP/1.1 and keeps the TCP connection to the server open), a fair allocation of the available bandwidth would have been 9 Mbit/s per TCP connection. The background node is using a manyfold of that, while the DASH players stream at an average bitrate of 1871 kbit/s ($\sigma = 2703$ kbit/s) without assistance from the Service Manager, and at 2764 kbit/s ($\sigma = 2812$ kbit/s) with assistance.

From this we can conclude that the fair sharing mechanisms in TCP are not adequate for DASH traffic. The bursty ON/OFF download behavior of DASH players sets them back when they have to compete with other continuous TCP flows. Explicitly informing DASH players about optimal quality levels is thus only effective when there is no background traffic. However, this is

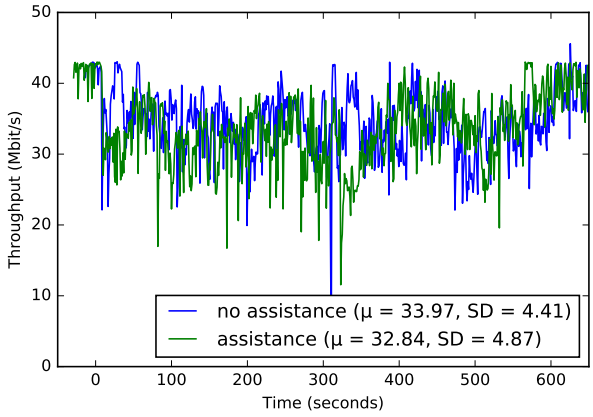

Figure 4: Throughput of the background traffic. DASH streaming starts at t = 0.

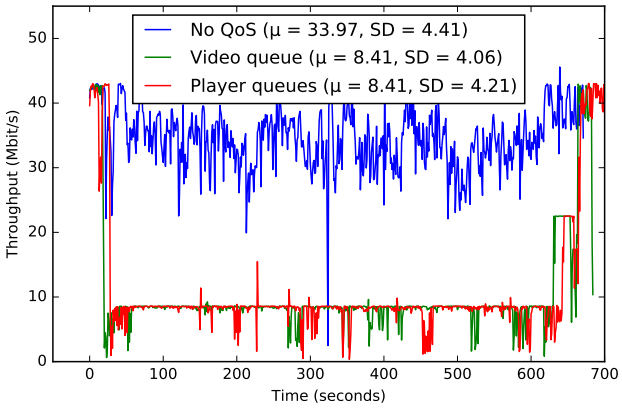

Figure 5: Throughput of background traffic is brought back to 8.4 Mbit/s when DASH players are active

an unrealistic assumption. Thus, explicit adaptation assistance on its own is not a sufficient mechanism for improving the quality of DASH streams.

5.2 Only dynamic QoS support

The second mechanism that is offered by our architecture is to provide dynamic quality of service support in the network. The Network Controller can create, modify, and destroy queues on the fly. In this evaluation we study two different queueing implementations. The first implementation classifies traffic into two groups: DASH traffic and non-DASH (background) traffic. For each group it will create a separate queue, modifying the size the of DASH queue to the current demand.

The second QoS implementation uses one queue per active player. The throughput of individual queues is not capped. Instead, the traffic control mechanism guarantees a minimum transmission rate, if there is data to be transmitted. This means that the minimum transmission rate only applies to the bottleneck link that is controlled by the DANE. In case of other bottlenecks, The combined throughput for the video queues is limited to the demand of the DASH streams. We call this queueing implementation *client queues*.

The problem with DASH streams in combination with background traffic, is that the background traffic takes up a too large portion of the available bandwidth. Limiting the throughput for background traffic leaves the DASH players the room they need to stream at the desired bitrate. The difference in throughput for the background traffic is shown in Figure 5. In both queueing configurations the throughput is brought back to 8.4 Mbit/s when the players are started, and restored to 42 Mbit/s after the streams are finished.

The difference in streaming bitrate for limiting, and not limiting the background traffic, is shown in Figure 6 for the different queueing implementations. Without QoS support and in the presence of background traffic, the DASH players perform poorly. Adding QoS support has the effect that the players are able to reach the target quality level 10 at 7234 kbit/s. The difference between the two queueing implementations is minimal in terms of mean quality level. Without background traffic, the mean quality in one level higher when using client queues: 9.10 ($\sigma = 3.08$) with one video queue, versus 10.43 ($\sigma = 1.45$) with separated client queues. With background traffic, the two queueing implementation perform similarly: 9.31 ($\sigma = 2.95$) with one video queue, versus 9.74 ($\sigma = 2.64$) with separated client queues.

Figure 6 does indicate a difference in the range of representations that are selected. To evaluate the stability of the stream we look

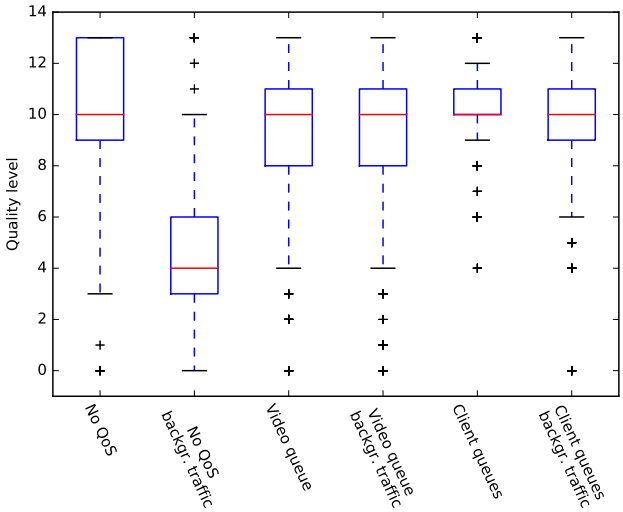

Figure 6: Comparison of the quality levels for different QoS configurations, with and without background traffic

at the number of switches, the time between switches, and the number of quality levels jumped. Figure 7 shows the number of switches for the four players combined. As expected, adding a single queue does not lower the number of quality switches. The traffic control mechanism still provides the players a single band which can be freely divided among the players. The video queue keeps the background traffic from interfering, but the players still have to compete with each other for bandwidth.

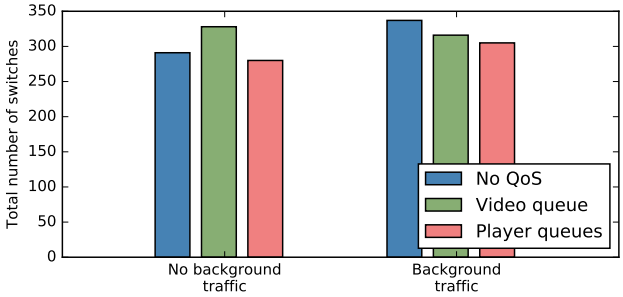

Figure 7: Number of switches in video quality given different QoS configurations, with and without background traffic

Enabling separate queues for different players also does not eliminate switches from occurring, nor does it help the players to stabilize at a certain rate. Figure 8 shows the distribution of the time between two switches, weighted by the number of segments in each interval without a switch. From this we can obtain that in none of the scenarios the DASH players were able to stabilize at a single quality level for a longer time, and that the quality switches are thus distributed over the length of the video. Separated video queues do not give longer intervals without a quality switch.

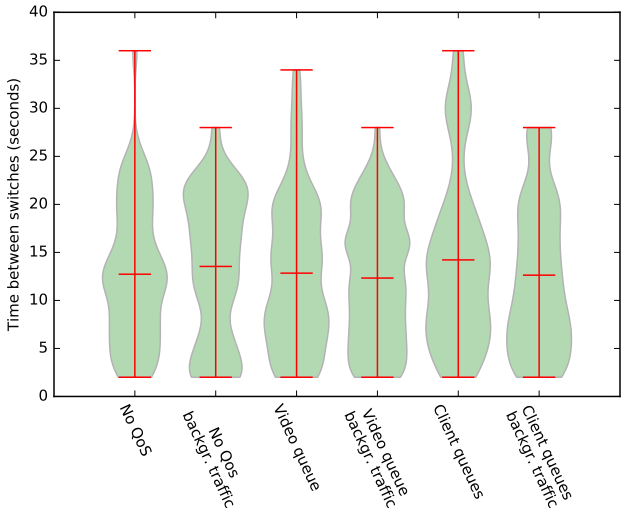

Figure 8: Probability density of the time between two switches for different QoS configuration, with and without background traffic

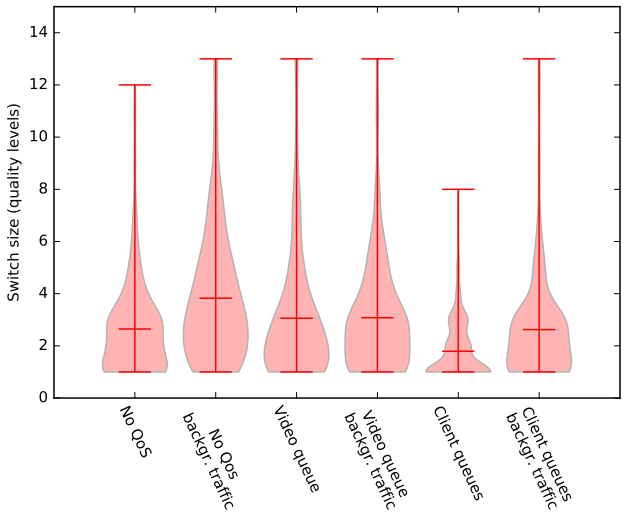

Figure 9: Probability density of switch sizes for the different QoS configurations, with and without background traffic

With regards to the size of the switches, using client queues reduces the size of quality switches, as depicted in Figure 9. A smaller switch will be less noticeable by the users and thus improves stability. Guaranteeing a minimum throughput for each player thus have a positive effect on the quality of experience for the stream. Nevertheless, this effect diminishes in the scenario with background traffic.

We ascribe this to the fact that we are using a Wi-Fi network in our evaluations. In most wired networks down- and upstream traffic are separated and do not affect each other. However, in a Wi-Fi network the same transmission channel is used for both down- and upstream. When the network becomes more loaded it takes longer for the clients to successfully make a request for a next video segment. New video segments are only requested when the download of the previous segment is completed, reusing the TCP connection to the server. Latency between sending out the segment request and the actual start of the download of the segment negatively affects the effective download speed.

In the cases with background traffic enabled, the load on the Wi-Fi network reached its maximum. In these scenarios, it takes more time for the clients to access the medium and request segments. Both the single video queue and the client queues work only one way, and the node must still compete for the medium. Therefore, they will not improve the upstream traffic. The fluctuations and delay of the segment requests, caused by the heavy load on the Wi-Fi network, cannot sufficiently be removed by adding QoS queues. Therefore, the DASH player will react to these fluctuations causing more instability.

5.3 Combining explicit adaptation assistance with dynamic QoS support

When explicit adaptation assistance is combined with quality of service support in the network, the user can benefit from the advantages from both the mechanisms. The explicit adaptation assistance eliminates the uncertainty and the unfamiliarity of the players with the network. The dynamic QoS support in the network is matched to the target bitrates that are sent to the players. Therefore, the DASH players are not sensitive to the background traffic and show good performance.

In this section we only focus on the scenario with background traffic. Figure 10 shows that enabling quality of service in the network significantly increases the quality level, and when combined with application level assistance in the player, the stream show almost no variation.

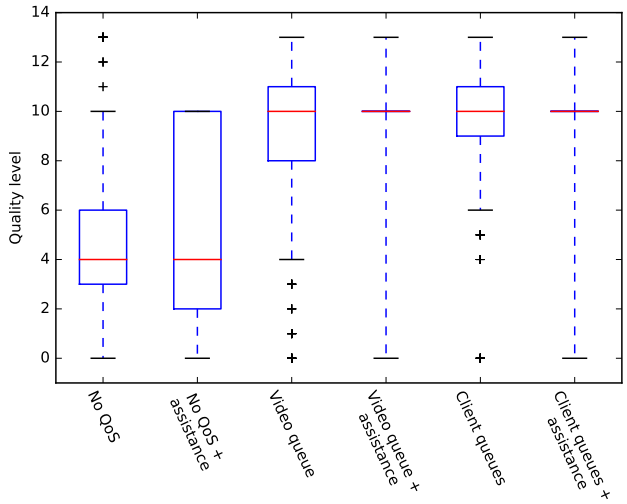

Figure 10: Comparison of only using QoS support with combining QoS support with explicit adaptation assistance in scenarios with background traffic

A similar effect is also visible when comparing the number of quality switches, as displayed in Figure 11. It shows that our

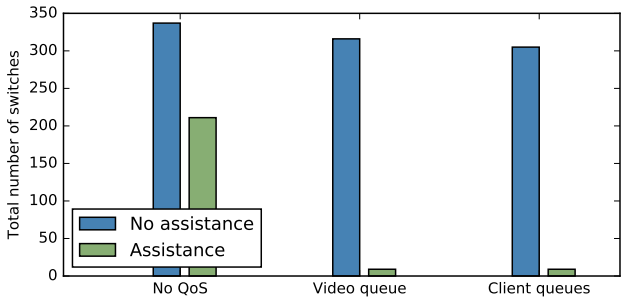

Figure 11: Comparison of the number of quality switches with and without adaptation assistance for the different QoS configurations

architecture considerably reduces quality switches and provides the viewers a stable streaming experience. The only time that players make quality switches and are not streaming at the target bitrate, is at the beginning of the stream. Once a player has loaded the manifest it will share the stream's detail with the manager, and it will start downloading segments based on the built-in adaptation mechanism. Only when the player has received a target bitrate from the Service Manager, and it reaches sufficient download speed and buffer level, the player is confident enough to follow the adaptation assistance.

Alternatively, players could have immediately followed the Service Manager's target bitrate. However, we have chosen not to implement this for two reasons. The first reason is that it provides players more freedom in starting the stream. Selecting lower bitrates in the beginning increases the buffer fill levels and allows to start streaming sooner. Secondly, the manager can only provide guarantees on the local network, but cannot ensure sufficient capacity at the server or on other network links. Starting at the target bitrate from the beginning could have caused stalls in the video, which should be avoided at all costs.

The effect of this decision for the DASH.js based players is shown in Figure 12. It shows that the players download between 16 and 24 segments (i.e. between 32 and 48 seconds of video) in a bitrate that is not the target bitrate. However, once the players hit the target bitrate, they stabilized and all further segments are requested for the target bitrate.

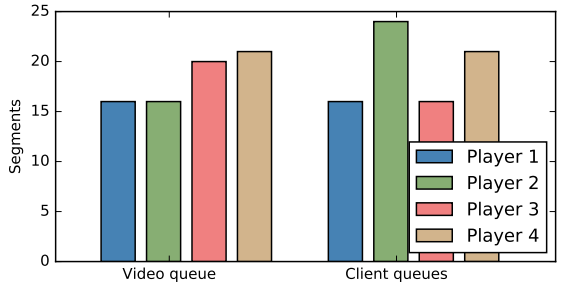

Figure 12: Number of segments downloaded before the streams stabilize

5.4 Node at network edge results

In the previous experiments all nodes where relatively close to the access point. For none of the nodes, the distance to the access point was such to be a cause for degraded network performance. However, in many Wi-Fi deployments it is common that there is a node that is further away from the access point. For such a node it would not be possible to use the full capacity of the Wi-Fi network. Video streaming, even with the high quality DASH video that we have prepared, does not require the full capacity of the network. We consider the scenario where a node that is further away from the access point can stream at high bitrates, but only when it is the only active node in the network. However, when there is other traffic in the network, the network performance of this node will decrease and the DASH player will have difficulties to maintain the high quality stream.

In this experiment the node is located at such a distance from the access point that it is able to stream at the highest quality levels, but it does not have any remaining capacity left. The streaming performance for this node is shown in Figure 13 on the left. It displays that the node alone selects the high quality segments. When the other DASH players become active (Figure 13 center), the quality of the DASH stream of the node further away is significantly lower. Adding QoS support in the network cannot restore the streaming performance of this player. The video quality is further degraded in the scenarios with background traffic. In these scenarios the video quality is degraded to the lowest video levels. The DASH players that are close to the access point perform similar to the previous evaluations, but are not shown for brevity.

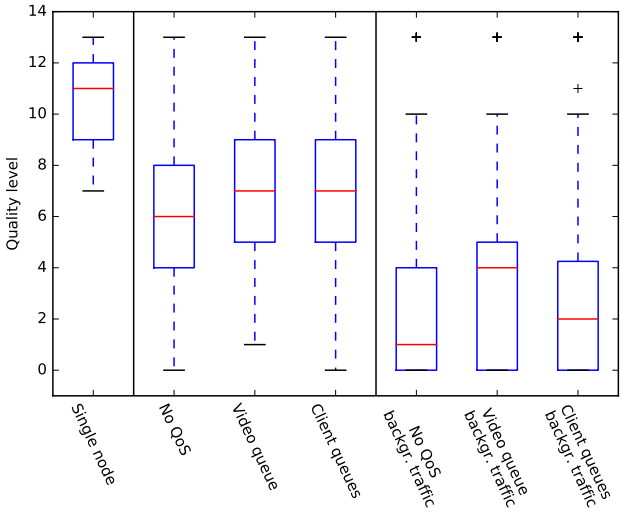

Figure 13: Video quality levels for the node further away from the Wi-Fi access point given QoS support in the network (left: one node, middle: all video clients, right: video clients and background traffic)

This shows again that in a Wi-Fi network the upstream traffic from time-sensitive applications – such as DASH streaming – should have enough room to get through without much delay. Part of this can be accomplished via explicit adaptation assistance, because it works in two ways. On the one hand it tells the players that they don't have to select the lower bitrates and that fluctuations in bandwidth are only temporary. On the other hand, it prevents the players from selfishly increasing the video capacity and putting a too large demand on the network.

Figure 14 on the left shows that without background traffic, the node further away from the access point streams at the same quality level as the nodes that are close to the access point. With background traffic the network is loaded too much to reach the same quality. However, compared to the setting without assistance, the quality level is almost doubled and the number of switches in quality is reduced, as shown in Table 1.

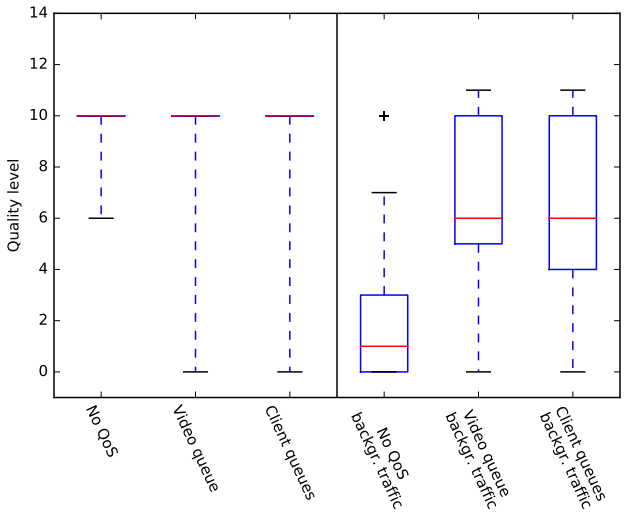

Figure 14: Quality levels for the node further away from the access point with explicit adaptation assistance enabled (left: without background traffic, right: with background traffic)

Table 1: Mean quality and number of quality switches for the node further away from the access point, with backgr. traffic

	Quality level (σ)	Nr. switches
No assistance		
No QoS	2.54 (3.18)	89
Video queue	3.66 (3.29)	85
Client queues	2.95 (3.45)	82
Assistance		
No QoS	2.26 (3.02)	76
Video queue	6.51 (3.09)	53
Client queues	5.98 (3.05)	43

The quality of the stream of the node further away could potentially be improved by lowering the throughput of the background traffic via the QoS mechanism. However, it is up to the users or the network administrator to decide whether further limiting the background traffic weighs up to the increase in video quality for a single user.

6. CONCLUSION

Dynamic adaptive streaming over HTTP is a popular technology for streaming video over the Internet. The reason for its popularity is the simplicity of the protocol and of the underlying transport mechanism using HTTP. DASH allows to do adaptive streaming and provides content providers scalable distribution in which off the shelf HTTP servers can be used in combination with content delivery networks. The downside of using HTTP as transport mechanism is that it creates a mismatch between the type of traffic and the used protocols. TCP shows to underperform when it is combined with the bursty traffic that DASH creates.

Novel networking paradigms, such as software defined networking, can be leveraged to reduce the effect of the protocols on network performance. An important aspect of the architecture that we presented in this paper, is that the competition for bandwidth, for which it is shown that DASH players lose too much bandwidth, is eliminated. The benefits of the DASH protocol stack remain intact, while the drawbacks are overcome by adding a simple interaction mechanism between the DASH players and the in-network Service Manager. This keeps our architecture simple, but it is powerful.

Our SDN-based architecture offers DASH players two mechanisms for improving the streaming performance: explicit adaptation assistance by informing the players about optimal bitrates, and via dynamic quality of service support. This approach has the advantage over fully client-side adaptation because the Network Controller and Service Manager have more knowledge about the network, the current network activity, and other DASH players. That makes that the Network Controller and Service Manager can make network-wide optimization decisions. Furthermore, our architecture is based on DASH and SDN, techniques that have shown to be scalable. We expect our architecture to scale as well, because in bigger network infrastructures its components can be distributed over multiple network devices and controllers.

In perfect conditions, without other bottlenecks between the server and the DASH players, our architecture provides highly stable and high quality video. If there are other bottlenecks, the players will decide not to follow the Service Manager's target bitrate. Therefore, our architecture will increase the streaming quality when it can, and DASH players will perform the same as the state-of-the-art players in less ideal conditions. A limitation of our architecture is that it requires DASH players to co-operate with the Service Manager. Otherwise, the players cannot benefit from assistance and network resource provisioning, and they are potentially assigned to the background queue.

By means of a thorough evaluation in our Wi-Fi testbed, we demonstrate the impact of these two mechanisms. We show that although both mechanisms improve the streaming performance, neither of the two are sufficient to provide a stable streaming experience when used on their own. Only when combining these two mechanisms we can provide users stable streaming at high video quality. We summarize the insights that we obtained in this study in the following paragraphs.

Explicit adaptation assistance enables stable streaming and fair sharing of network resources between players, but only in networks that are solely used for DASH streaming. However, it does not prevent DASH streams to underperform in busy networks (i.e. with background traffic), as a result of the mismatch between TCP and the bursty nature of DASH traffic. The underperformance of the DASH streams can be restored by adding quality of service support in the network. Dynamic quality of service support increases the video quality up to the desired level, but the DASH players will still react to slight variations in throughput, causing instability.

Providing DASH players separate queues with a minimum rate guarantee does not reduce the number of quality switches, but the size of the switches is smaller compared to a single video queue that is shared among the players. The only way to eliminate the quality switches is to combine explicit adaptation assistance with QoS support. Using this setting, DASH players stream stable at the desired quality level in scenarios without and with background traffic. However, in the setting where both mechanisms are active, it is not required any more to use separated queues.

In a Wi-Fi environment, nodes that are further away from the access point are hindered in steaming. Our architecture can restore streaming performance of these nodes in a network without background traffic, and significantly improve the video quality in environments with background traffic.

In future research we will work on extending our architecture to Wi-Fi specific issues. We will investigate which mechanisms are available, for example to improve the video quality of the node that is further away from the access point, or to handle interference from

other Wi-Fi networks. Moreover, we will investigate the impact on DASH players that are not modified to interact with the service manager. Finally, we will focus on the development of bandwidth sharing policies, where we will not only focus on the performance of DASH streams, but also on the effects of a certain policy on background traffic.

7. REFERENCES

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

[2] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis. What happens when HTTP adaptive streaming players compete for bandwidth? In *NOSSDAV '12: Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, pages 9–14, New York, New York, USA, June 2012. ACM Request Permissions.

[3] S. Akhshabi, A. C. Begen, and C. Dovrolis. An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, pages 157–168, New York, NY, USA, 2011. ACM.

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

[5] Cisco. Cisco visual networking index: Forecast and methodology, 2014–2019. *CISCO White paper*, 2015.

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

[7] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. *Commun. ACM*, 56(3):91–99, Mar. 2013.

[8] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac. Interactions Between HTTP Adaptive Streaming and TCP. In *Proceedings of the 22Nd International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 21–26, New York, NY, USA, 2012. ACM.

[9] P. Georgopoulos, M. Broadbent, A. Farshad, B. Plattner, and N. Race. Using Software Defined Networking to enhance the delivery of Video-on-Demand. *Computer Communications*, 69:79–87, 2015.

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

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

[12] T. Hoßfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia. Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies. *Computer Networks*, 81:320–332, 2015.

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

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

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

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

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

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

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

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

[21] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.*, 12(2):28:1–28:24, Oct. 2015.

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

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

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