Achieving QoE Fairness in Video Streaming via Client-Network Interaction

Junyang Chen

Supervised by Prof. Rodrigo Fonseca

Department of Computer Science
Brown University

Spring 2016

Abstract

Adaptive video streams, when competing behind a single bottleneck link, experience unstable video quality, underutilized links, and unfair distribution of bandwidth. Different classes of solution have been proposed. On client side, bitrate adaptation algorithms have been designed to minimize the video bitrate changes as well as attain as high bitrate as the network resources allow. On the server side, efforts have been made to optimize the efficiency in Content Delivery Network (CDN) server caching and selection. However, those approaches do not account for the problem of unfairness because they are unaware of competing video streams. Network services have been proposed to bridge this gap, but the wide spread use of HTTPS renders those solutions ineffective, even impossible.

In this work, we present client-Driven Video Delivery (cDVD), an in-network service for video streaming over HTTPS that builds on software-defined principles. cDVD provides client-level APIs that enable the interaction between clients and network even in the presence of HTTPS, and uses the exchanged information to achieve stability, high bandwidth utilization, and fairness when multiple video clients share the same access link.

1 Introduction

Video streaming has become the dominant application on the Internet and its traffic is continuously increasing. A recent report from Cisco estimates that online videos will account for four-fifths of the global Internet traffic by 2019 [8]. With the increasing popularity of video streaming services, considerable amount of work has been devoted to optimize its delivery efficiency for the provider side [13, 19, 20, 26], as well as viewing experience for the client side [3, 6, 9, 12, 14, 15, 18, 30]. Specifically, adaptive video streaming techniques have been created and widely used to help users select the best video bitrate automatically based on current network resources.

The streaming standard, Dynamic Adaptive Streaming over HTTP (DASH), is defined by the MPEG Foundation [25, 29]. Studies show the advantages, as well as the pitfalls, of DASH [2, 3, 16, 21].

In particular, the presence of multiple DASH clients that compete for network resources under a single bottleneck link causes (i) instability in the selected video encoding, (ii) bandwidth under-utilization, and (iii) unfair distribution of bandwidth. Each of them has a corresponding negative impact on users' perceived viewing experience. Instability makes video quality vary over time, disturbing users' visual attention and thus affecting quality of experience (QoE) negatively [28]. Bandwidth underutilization prevents the client from selecting the highest possible video bitrate that it is able to play. Unfair distribution of bandwidth might cause one client to select a video bitrate that produces much better QoE than others. Such unfair distribution of bandwidth can result from discrepancies in start-time, operating system support, and player implementation [2, 21]. In total, the complex interaction among DASH clients, which are intended to improve video delivery, *can negatively impact the quality of experience*.

There have been many attempts to address the problems. Multiple adaptation algorithms have been developed to help clients select the best video bitrate based on estimated bandwidth and the size of accumulated buffer [9, 14, 18, 30], and there is a increasing tendency to do the adaptation based on buffer only [14, 30] because of proven improvements. On server side, centralized control frameworks with algorithms to optimize the efficiency in CDN caching and selection have been proposed, and shown to deliver better QoE to users [19, 20, 26]. However, the lack of considering multiple clients competing on a bottleneck link limits their capability to achieve the desired viewing experience for more than one video session. Network services with the global view of available resources as well as active video sessions have the best chance to achieve QoE fairness for multiple DASH clients [6, 12, 22]. Nevertheless, the required use of deep packet inspection (DPI) on video traffic makes these approaches infeasible as network traffic is increasingly encrypted [27].

To bridge the gap made by HTTPS, we have architected and implemneted an in-network service, called client-Driven Video Delivery (cDVD), that builds on software-defined networking (SDN) and participatory networking (PANE) principles [11]. cDVD provides a set of APIs for clients to interact with the network and uses the exchanged information to (i) allocate bandwidth based on an algorithm and metrics that are proven to be maximally fair [22], and (ii) generate feedback to clients to achieve stability and high bandwidth utilization in an *encrypted environment*.

This report is organized as follows. Background of how DASH works and its problems are presented in Section 2. We discuss about the level of fairness we can achieve based on the amount of information exchanged between clients and network in Section 3. In Section 4, we outline the architecture and our implementation of cDVD, followed by a set of experiments to evaluate the system in Section 5. Finally, we discuss about the new opportunities this work opens up and the potential future work before drawing our conclusion.

2 Background and Motivation

In this section, we will briefly introduce how DASH works, the problems it entails, and raise the inefficiencies in the existing solutions as our motivation for this work.

2.1 DASH

In DASH, each video has multiple encodings that coresspond to different bitrates, frame rates, and resolutions. Each encoding is segmented to video fragments that generally last 1 to 15 seconds. A DASH player issues intermittent HTTP requests to fetch the fragments, usually more than it needs,

to fill up the player buffer for smooth viewing experience. Each video is associated with a Media Presentation Description (MPD) file that contains the encoding information of video and audio streams, locations of content servers, and other information that is needed by the clients. Clients request the MPD file upon initiating a video session and select video and audio encodings to play based on estimated throughput and/or buffer size. The traffic is commonly encrypted (HTTPS) in commercial video streaming services.

2.2 Unstable Video Quality

Instability of video quality could be caused by varying network condition, variable bitrate (VBR) encoding, and on/off behavior of TCP connections. Videos are commonly encoded using VBR [17], which enables using low bitrate to encode simple scenes and high bitrate to encode complex scenes, resulting in reduced average (nominal) bitrate. Real world examples show that the highest bitrate of a fragment could be twice as large as the average bitrate [33], the bitrate used to label each encoding. Consequently, VBR videos that have significant variation in their fragment bitrates could cause rebuffering and bitrate down-switching. The on/off nature of TCP connection biases the bandwidth estimation when multiple clients share the same bottleneck link. Assume there are two clients and both issue one HTTP request a time for video stream. The durations of two clients' TCP connections could be overlapped or separate. The bandwidth estimation of the former case will be lower than that of the latter, leading to inaccurate estimations and instability [2].

2.3 Variation in Implementation and Setting

Given the DASH standard, unfairness still exits. The implementation details as well as the adaptation logic of clients are left to the discretion of the video providers. Consequently, some providers implement their clients more aggressive than the others [3] and the differences make achieving fairness difficult. For example, a client that uses multiple TCP connections simultaneously can easily obtain disproportionally more bandwidth [5]. Figure 1 shows the video bitrates selected by the clients that

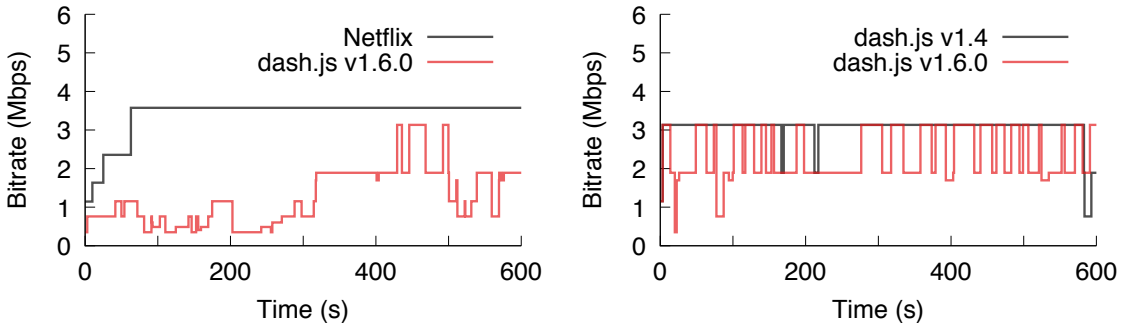


Figure 1: Serial vs parallel requests. Netflix and dash.js 1.4 use parallel TCP flows, and get more bandwidth than dash.js 1.6, which does not.

share the same bottleneck link and play for 10 minutes. It is apparent to see that Netflix and `dash.js 1.4` that request video fragments in parallel occupy more bandwidth than `dash.js 1.6` client that only requests one video fragment a time. Other factors, such as the the start-time, operating system

support, and content server resources are also relevant in causing the unfairness among competing clients [2, 21, 22].

2.4 Problems in Existing Solutions

Solutions specifically target multiple video clients sharing a bottleneck link have been proposed.

Client-side adaptation algorithm. FESTIVE is an adaptation algorithm that improves fairness, bandwidth utilization, and stability by averaging past throughput estimates for bandwidth estimation, considering current bitrate when switching, delaying switching, and randomizing scheduling of fragment download [15]. However, it does not consider that a video encoding generates different QoE on different screen resolution as suggested in [22], and the improvements it is able to achieve are limited without the global view of active video sessions and available resources.

Network service. AVIS is a gateway-level network resource management framework in cellular network [6]. It is able to differentiate DASH flows from regular video flows and other traffic and allocate proportionally fair share of bandwidth to DASH flows while maintaining high utilization. However, its ignorance of considering varying screen resolutions on mobile devices limits the degree of fairness it can gain and the ubiquitous use of HTTPS makes the DPI middleboxes it depends on impractical [27]. QFF is a QoE fairness framework using OpenFlow [12, 23]. Building on software-defined networking, QFF has a global view of active video sessions and network resources to allocate bandwidth in a QoE-fair manner using Structural Similarity (SSIM) index-based utility functions [31]. However, the use of OpenFlow and SSIM limits its application and generality since OpenFlow is not widely used in today's network and SSIM-based utility functions varies across videos. VHS, a network-layer QoS framework running on routers, is able to achieve QoE fairness in a home environment [22]. It uses a generic utility function by considering screen resolution and allocates bandwidth based on an algorithm that is proven to be maximally QoE fair. Nevertheless, its use of DPI limits its capability of monitoring encrypted video traffic.

3 Client-Network Interaction

The amount of improvement we can achieve depends on the level of interaction between clients and network. The types of interaction, along with their effect, are summarized as follows:

- *Level 0 - No Interaction*: Any kind of fairness is not guaranteed. Clients experience instability and bottleneck link is under-utilized. The best fairness clients can achieve is each TCP flow gets the same throughput [7], but under limited circumstances [24, 32]. Also, as discussed in Section 2.3, clients initiating multiple TCP flows unavoidably obtain larger portion of bandwidth.
- *Level 1 - Notifications from Clients*: Network is able to allocate bandwidth evenly to clients given session notifications from clients. However, in absence of utilities, QoE fairness is impossible to achieve.
- *Level 2 - Utilities from Clients*: With the knowledge of the utilities (e.g., screen resolution and priority), network is able to allocate bandwidth fairly such that the video sessions will have the closest (if not the same) quality of experience that network is able to achieve given the available bandwidth.

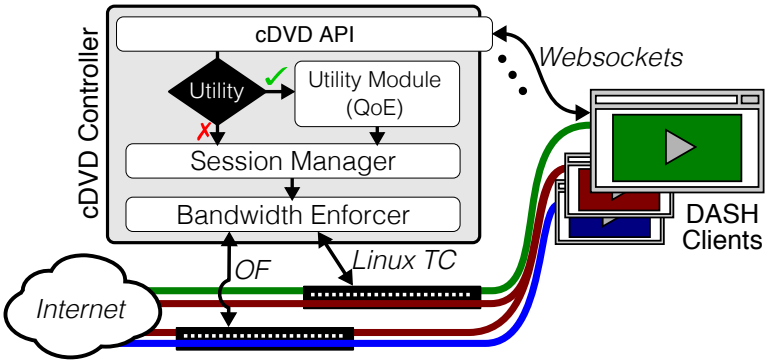


Figure 2: High-level cDVD architecture

- *Level 3 - Feedback to Clients*: Network can return feedback that will help improve QoE to clients. For example, network can inform clients their allocated bandwidth, so that the clients can select the best video encoding whose bitrate is less than the allocation. Another example would be that the network suggests the video encoding for the clients to play, which will be shown in our implementation.

4 Overview of cDVD

We present the architecture and a prototype implementation of cDVD, a client-Driven Video Delivery service in the network. By evaluating our implementation, we show that QoE fairness, stable viewing experience, and high utilization of bottleneck link become achievable via the interactions between clients and cDVD in an encrypted environment.

4.1 Architecture

Figure 2 shows the architecture of cDVD, where there are three video sessions that traverse two routers controlled by the cDVD controller. Each session can comprise multiple flows. Clients interact with the cDVD controller through a client interface, and the controller, in turn, applies bandwidth control to the routers via a QoS mechanism, such as OpenFlow or direct queue configuration (e.g., Linux Traffic Control). Finally, upon requested, cDVD can send feedback to clients to help with their bitrate adaptation. *Utility Module* contains the set of functions that utilizes the utilities sent from clients, including QoE metrics that maps a specific video encoding to a QoE score for a specific screen resolution and an allocation algorithm that can distribute the bandwidth to clients in a way that ensures QoE fairness across sessions using the QoE metrics. If utilities are sent from clients, this module will be activated and plugged to the *Session Manager*. *Session Manager* oversees the active video sessions and manages network resources for each session. It allocates bandwidth evenly, or QoE fairly if *Utility Module* is activated, to clients. It then passes the allocation to *Bandwidth Enforcer* to limit the bandwidth each session can use.

4.2 Implementation

In our implementation, we implement and place the cDVD controller on a router (as in most homes), where there are one or more clients sharing the single bottleneck link. We assume that the router is the local DNS resolver as it is easy for clients to locate the controller, that there is only one simultaneous video session between each pair of client and server machines, that there is no other processes competing the local bandwidth on each client machine.

The cDVD controller is written in C++ with about 600 lines of code. We use TP-Link TL-WR1043ND v1.8 as our router, that runs OpenWrt, a GNU/Linux based firmware program for embedded devices [1]. We integrate into *Utility Module* QoE metrics that take into account the screen resolution and an allocation algorithm that is proven to be maximally fair [22]. To make it general, we uses Linux Traffic Control (`tc`) as our QoS mechanism. The controller has a websocket server that interacts with clients.

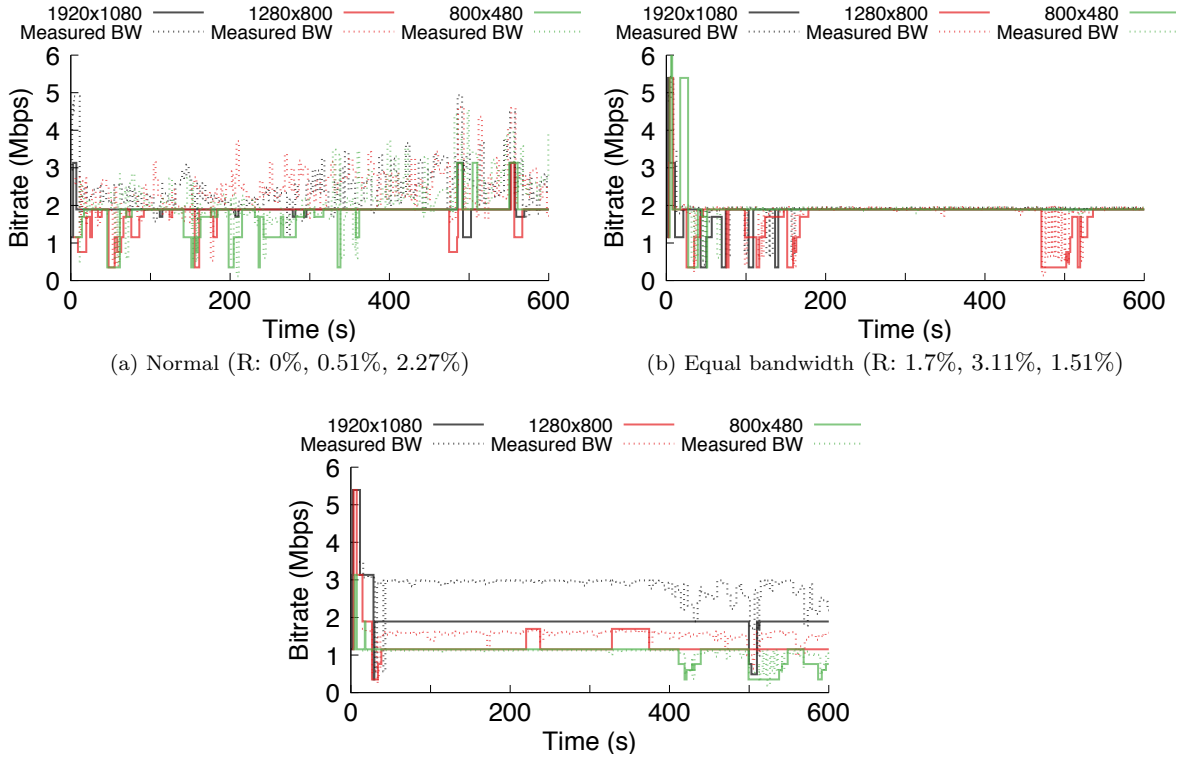
TP-Link TL-WR1043ND v1.8

We create our DASH client by modifying `dash.js 1.6`, the open-source reference DASH client from the DASH Industry Forum [10]. We instrument the client so that it can locate and interact with the cDVD controller using a websocket channel. We also change the adaptation logic so that the client is able to utilize the feedback from the controller to improve QoE.

When initiating a new video session, our DASH client tries connecting to well-known domain and port (`cdvd.local:9000`) via websocket. If the controller responds, the client operates in a cDVD mode, otherwise it operates as a normal DASH client. The `cdvd.local` is mapped to the controller's IP address in the router's DNS configuration. In cDVD mode, the client communicates with the controller via client APIs in `json` format. The details of the client APIs we support can be summarized as follows:

- $Notify(Controller, SessonId, Utility)$
- $NotifyFeedback(Controller, SessionId, Utility) \rightarrow Feedback$

Coupled with the description in Section 3, the *Controller* parameter specifies the location of the cDVD controller, which is `cdvd.local:9000` in our case. The $SessionId$ parameter is used to identify and differentiate video sessions. We use IP address here based on the assumption that each machine has only one video session. The *Utility* parameter contains the information needed to establish QoE fairness, such as screen resolution, bitrate list and resolution of each video bitrate in our case. cDVD allocates bandwidth to equalize QoE across sessions if *Utility* is given [22]. Otherwise, it allocates $1/N$ of total bandwidth to each session, where N is the number of current video sessions. Once allocation is computed, the controller generates a script that contains Linux `tc` commands to create one network queue per session, and to assign the TCP flow(s) corresponding with each session to the right queue. In response, cDVD controller replies $Feedback$ to clients. We use *suggested bitrates to play* here as our $Feedback$.



(a) Normal (R: 0%, 0.51%, 2.27%)

(b) Equal bandwidth (R: 1.7%, 3.11%, 1.51%)

(c) Equal QoE (R: 0.29%, 0.27%, 3.75%)

Figure 3: Bandwidth allocated fairer with more client information (with rebuffering ratios R).

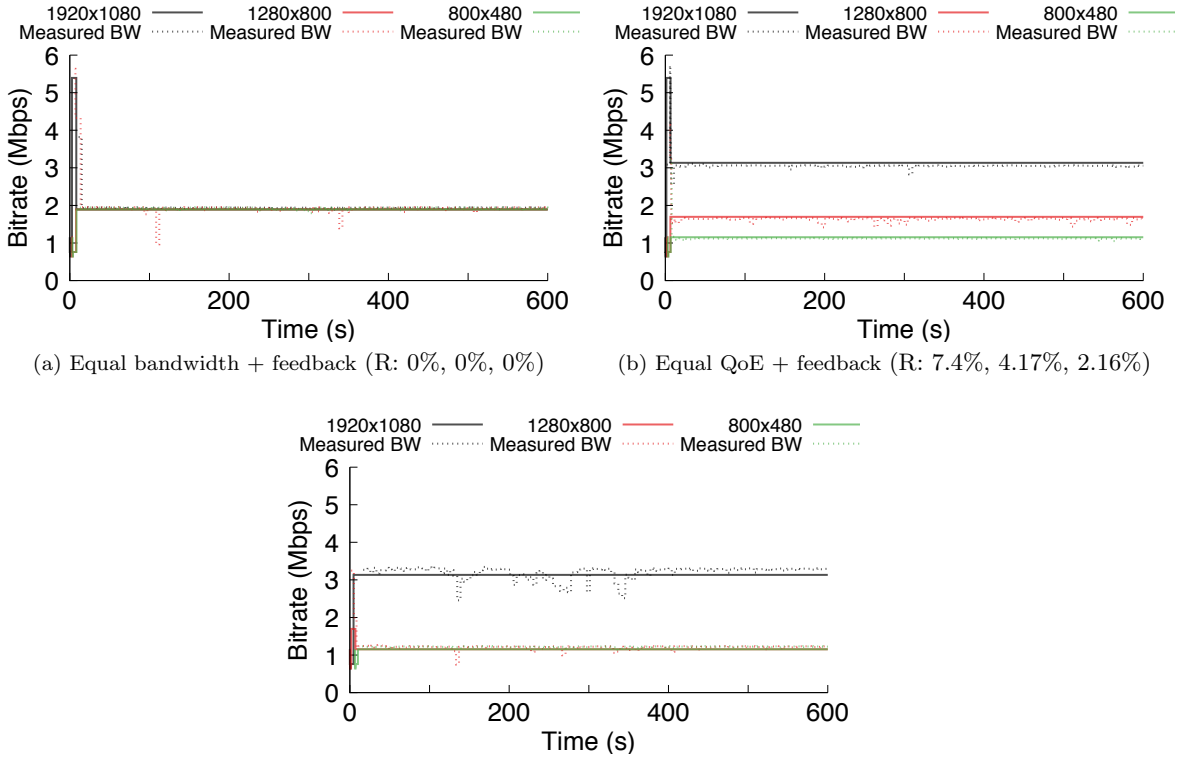
5 Evaluation

To evaluate cDVD, we set up 3 DASH clients with different screen resolutions to compete behind a 6Mbps bottleneck link, which is enforced using Linux `tc`[1]. The clients request the BBC DASH Testcard stream [4] from the wider Internet where upstream bottleneck exists as in streaming from commercial video services. The stream has 13 video encodings, 2 audio encodings, and a duration of up to 1 hour.

5.1 Transition to QoE Fairness

A set of experiments is designed and and shown in Figure 3 to evaluate the improvements we can gain from clients sending information to the cDVD controller. The screen resolutions of the three clients are labelled at the top. The requested bitrates are plotted as solid lines along with their corresponding measured bandwidths plotted as dotted lines. The measured bandwidth records the throughput of

1. In our experiments, the traffic capacity of each network queue is increased by 6% to offset the under-allocation of Linux `tc`.



(a) Equal bandwidth + feedback (R: 0%, 0%, 0%)

(b) Equal QoE + feedback (R: 7.4%, 4.17%, 2.16%)

(c) Equal QoE + feedback + headroom (R: 0%, 2.15%, 0%)

Figure 4: Experimental feedback where cDVD *tells* clients which bitrate to select (with respective rebuffering ratios R).

the last downloaded video fragment. The clients adjust their video qualities using `dash.js 1.6`'s default bitrate adaptation logic.

Figure 3a records the scenario where there is no interaction between the clients and the controller. As we can see, the 3 clients are trying to converge their bitrates on the one that is lower but closest to the fair share of the bandwidth (2Mbps) with limited and different degree of success. Particularly noteworthy is the highly fluctuating measured bandwidth of the clients that is used to select bitrates. Without any interaction, competing clients get varying share of bandwidth with no fairness guaranteed.

In Figure 3b, cDVD is activated with the clients sending notifications (IP addresses) to the controller at the start of their video sessions using the APIs. Given only notifications, the controller is able to allocate 2Mbps to each client using Linux `tc`. This leads to more stable bandwidth measurements at client sides and the client are able to converge on the 'fair' bitrate more successfully. The instability that still exists in both Figure 3b and Figure 3c, we believe, is due to the upstream bottleneck and is beyond the control of our settings.

QoE fairness is achieved when the clients send both notifications and utilities to the controller in Figure 3c. As depicted in the measured bandwidths, three clients get different shares of bandwidth

with more bandwidth allocated to the client with higher screen resolution. The controller leverages an allocation algorithm that is maximally fair using the information of their screen resolution, available video bitrates and the resolutions of the bitrates. Because `dash.js 1.6`'s adaptation logic conservatively ignore bitrates that is larger than 90% of the measured bandwidth, selected bitrate could be substantially lower than measured bandwidth (e.g., black color lines in the Figure 3c), leading to an opportunity miss and bandwidth underutilization.

5.2 Feedback to Clients

We also explore the space where the cDVD controller sends feedback to the clients in response. In Figure 4, the controller sends back the suggested bitrates to play to clients, and the clients in response choose to only select the suggested bitrates. Bandwidth measurements are still recorded, but ignored in bitrate adaptation. A set of preliminary measures shows that QoE fairness, stability, and high utilization of bandwidth can be exactly achieved when the cDVD controller and clients interact bidirectionally.

Figure 4a and Figure 4b are the same as Figure 3b and Figure 3c respectively except that they only play bitrates suggested by the controller. As we can see, the measurements of bandwidth stabilize more when there is no switching of bitrates. The transition from Figure 3b to Figure 4a demonstrates the ideal effect of using feedback, where stability and efficiency are achieved without the expense of QoE (no increased rebuffering ratios). However, the increase of rebuffering ratios does occur in the transition from Figure 3c to Figure 4b, where bandwidth is allocated more aggressively. Specifically, the highest rebuffering ratio is 7.4% in Figure 4b, which exceeds the normal range in Figure 3 ($0\% - 3.75\%$). Quality of experience suffers if users spend long time waiting for rebuffering. To reduce the rebuffering ratios, 10% of headroom[2] is considered by cDVD controller when allocating bandwidth and generating suggestions as shown in Figure 4c, which is otherwise identical to Figure 4b. After adding the headroom, the rebuffering ratios decrease to the normal range and most streams do not experience any pauses, with more bandwidth still allocated to where there is greater utility (client with higher screen resolution).

QoE fairness, stability, high utilization of bandwidth, and low rebuffering ratios are all realized in Figure 4c, proving the significance of interaction between clients and network.

6 Discussion and Future Work

Incentives: The first question about any scheme that requires multiple parties to participate is whether incentives align. Clients should be no worse off by participating, and only serve better to users. By sharing information with the controller, the clients can have stable and sufficient bandwidth allocated. In practical, a cDVD controller can redirect non-participating video sessions' traffic to a default network queue, to which all other non-video traffic would also be redirected, and give no guarantees there. Clients should not get any unfair advantages by lying about their utilities. We view it as a potential future improvement to cDVD to handle theses scenarios.

Identify video session by ports: Our setting assumes that each machine only has video traffic, while realistically other traffic such as file transfer could also happen in the background. Therefore, not only IP address of machine but also ports each video session uses are needed so that we can control

2. By default, the required bandwidth for each bitrate is the same as the bitrate. After adding headroom, the required bandwidth for each bitrate is increased by x, where x is the headroom.

video traffic at the granularity of ports to avoid other competing local traffic. However, identifying video sessions' used ports is found difficult as information of transport layer is invisible to application layer. We leave it as a future work that DASH clients will be able to keep track of the ports they use and share this information with the controller.

Upstream bottlenecks and VBR: 10% of headroom used in our experiments is able to resolve the increased rebuffering ratios in Figure 4b, but it is not a general solution. For one, the Testcard media used in our experiments is mostly encoded in Constant Bitrate (CBR), where every video fragment has the same size, while most of the online videos are encoded in VBR for compaction. For another, upstream bottlenecks could be more likely to happen when streaming from commercial video providers, as there are more variances in terms of number of users competing server-side resources and geolocations of content severs (varying round-trip time). Our solution can only control the bandwidth of links down-stream of the control point and provide an upper bound on the bandwidth to a session. We view bitrate adaptation as a complement, and necessary fallback.

Different interaction: Clients are free to choose their own degree of interaction. For example, a client can only choose to share session notification while other might share their utilities. Those who share session notifications should get worse QoS in terms of QoE than those who share utilities but must be better off than those who share nothing. cDVD should be able to handle this case, as different parties have different interests and policies for their services. Conversely, clients can do bitrate adaptation using their proprietary algorithms without utilizing the feedback from the controller.

Bandwidth as feedback: Instead of having cDVD controller tell clients which bitrates to select, the controller can alternatively inform the clients what their allocated bandwidth is to help them adapt. For example, clients who use buffer-based only adaptation can use the allocated bandwidth to converge on the appropriate bitrates immediately before the required buffer is built up for adaptation [14]. How to utilize the knowledge of allocated bandwidth to improve existing adaptation algorithms remains an open question and should be further investigated in the future.

API expressiveness and scale: We define some client APIs in Section 4.2 for our exploration and experiments, but standardization is required. Other kinds of interaction arise, for example, when clients start reporting consistent upstream bottlenecks. The controller could then lend the unused bandwidth to other video sessions temporarily until the bottlenecks are resolved, improving bandwidth utilization even more. Other question is where are other possible control points that can utilize cDVD.

7 Conclusion

Unfairness, instability, and bandwidth underutilization occur when multiple video sessions compete behind a bottleneck link. Existing solutions on the client side lacks the global view to establish fairness, while solutions with a global view are rendered ineffective due to the ubiquitousness of HTTPS. We present cDVD, an in-network service, to bridge the gap. Building on SDN and PANE principles, cDVD is able to utilize the interaction between clients and network to achieve QoE fairness, stable video quality and high utilization of bandwidth while maintaining low rebuffering ratios in an encrypted environment. We propose a set of client APIs to allow clients and cDVD controller interact at different granularities, and use modified `dash.js 1.6` as our DASH client to conduct preliminary measurements, and get direct and significant improvements.

References

[1] OpenWrt. `http://openwrt.org`.

[2] Saamer Akhshabi, Lakshmi Anantakrishnan, Constantine Dovrolis, and Ali Begen. What happens when http adaptive streaming players compete for bandwidth? In *ACM NOSSDAV*, 2012.

[3] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proc. Multimedia Systems (MMSys)*, 2011.

[4] BBC. Testcard Stream. `http://rdmedia.bbc.co.uk`, 2015.

[5] Bob Briscoe. Flow rate fairness: Dismantling a religion. *ACM SIGCOMM Computer Comm. Review*, 37(2):63–74, April 2007.

[6] Jiasi Chen, Rajesh Mahindra, Mohammad Amir Khojastepour, Sampath Rangarajan, and Mung Chiang. A scheduling framework for adaptive video delivery over cellular networks. In *The 19th Annual International Conference on Mobile Computing and Networking, MobiCom'13, Miami, FL, USA, September 30 - October 04, 2013*, MobiCom '13, pages 389–400, 2013.

[7] Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 17(1):1–14, June 1989.

[8] Cisco. Cisco VNI: Forecast and methodology, 2014–2019, May 2015.

[9] Nicola Cranley, Philip Perry, and Liam Murphy. User Perception of Adapting Video Quality. *International Journal of Human-Computer Studies*, 64(8), 2006.

[10] dash.js player. `http://dashif.org/reference/players/javascript/`.

[11] Andrew D. Ferguson, Arjun Guha, Chen Liang, Rodrigo Fonseca, and Shriram Krishnamurthi. Participatory networking: An api for application control of sdns. In *Proc. ACM SIGCOMM*, 2013.

[12] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. In *ACM FHcMN Workshop*, 2013.

[13] Monia Ghobadi, Yuchung Cheng, Ankur Jain, and Matt Mathis. Trickle: Rate Limiting YouTube Video Streaming. In *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, 2012.

[14] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc ACM SIGCOMM*, 2014.

[15] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *ACM CoNEXT*, 2012.

[16] Robert Kuschnig, Ingo Kofler, and Hermann Hellwagner. Evaluation of http-based request-response streams for internet video streaming. In *Proc. ACM Multimedia Systems (MMSys)*, 2011.

[17] D.J. LeGall, A. Wells, and K.M. Uz. Variable bit rate encoding, July 27 1999. US Patent 5,929,916.

[18] Zhi Li, Xiaoqing Zhu, J. Gahm, Rong Pan, Hao Hu, A.C. Begen, and D. Oran. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE Journal on Selected Areas in Communications (JSAC)*, 32(4):719–733, April 2014.

[19] Hongqiang Harry Liu, Ye Wang, Yang Richard Yang, Hao Wang, and Chen Tian. Optimizing cost and performance for content multihoming. In *ACM SIGCOMM*, 2012.

[20] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *ACM SIGCOMM*, 2012.

[21] Ahmed Mansy, Mostafa Ammar, Jaideep Chandrashekar, and Anmol Sheth. Characterizing client behavior of commercial mobile video streaming services. In *Proc. ACM MoVid*, 2014.

[22] Ahmed Mansy, Marwan Fayed, and Mostafa H. Ammar. Network-layer fairness for adaptive video streams. In *Proc. IFIP Networking*, 2015.

[23] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[24] Sándor Molnár, Balázs Sonkoly, and Tuan Anh Trinh. A comprehensive tcp fairness analysis in high speed networks. *Comput. Commun.*, 32(13-14):1460–1484, August 2009.

[25] MPEG. DASH. `http://dashif.org/mpeg-dash`.

[26] Matthew K. Mukerjee, David Naylor, Junchen Jiang, Dongsu Han, Srinivasan Seshan, and Hui Zhang. Practical, real-time centralized control for cdn-based live video delivery. In *Proceedings of ACM SIGCOMM*, 2015.

[27] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberger, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. The cost of the "s" in https. In *Proc. ACM CoNext*, 2014.

[28] Demóstenes Z. Rodríguez, Zhou Wang, Renata L. Rosa, and Graça Bressan. The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over http. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):1–15, 2014.

[29] Iraj Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE Multimedia*, 18(4):62–67, 2011.

[30] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. BOLA: near-optimal bitrate adaptation for online videos. *CoRR*, abs/1601.06748, 2016.

[31] Zhou Wang, Ligang Lu, and A. C. Bovik. Video quality assessment using structural distortion measurement. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages III–65–III–68 vol.3, 2002.

[32] Qian Wu, Mingwei Gong, and Carey Williamson. Tcp fairness issues in ieee 802.11 wireless lans. *Comput. Commun.*, 31(10):2150–2161, June 2008.

[33] Y. Zhou, Y. Duan, J. Sun, and Z. Guo. Towards simple and smooth rate adaption for vbr video in dash. In *Visual Communications and Image Processing Conference, 2014 IEEE*, pages 9–12, Dec 2014.