Towards Network-wide QoE Fairness using OpenFlow-assisted Adaptive Video Streaming

Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent,
Mu Mu, Nicholas Race
School of Computing and Communications, Infolab 21,
Lancaster University, Lancaster, LA1 4WA,
United Kingdom
{p.georgopoulos, y.elkhatib, m.broadbent, m.mu, n.race}@lancaster.ac.uk

ABSTRACT

Video streaming is an increasingly popular way to consume media content. Adaptive video streaming is an emerging delivery technology which aims to increase user QoE and maximise connection utilisation. Many implementations naively estimate bandwidth from a one-sided client perspective, without taking into account other devices in the network. This behaviour results in unfairness and could potentially lower QoE for all clients. We propose an OpenFlow-assisted QoE Fairness Framework that aims to fairly maximise the QoE of multiple competing clients in a shared network environment. By leveraging a Software Defined Networking technology, such as OpenFlow, we provide a control plane that orchestrates this functionality. The evaluation of our approach in a home networking scenario introduces user-level fairness and network stability, and illustrates the optimisation of QoE across multiple devices in a network.

1. INTRODUCTION

Recent years have seen the growing popularity of video streaming in best effort IP networks, thanks to the increasing computational and display capabilities of user devices. In 2011, Internet video traffic accounted for 51% of all consumer Internet traffic. By 2016, it is expected to be 55% [5]. High Definition content has also become the de facto quality level consumed by users: in 2011, high definition video traffic surpassed standard definition for the first time [5].

Although bandwidth provision in consumer networks has been greatly improved in the last few years, streaming high bitrate audio-visual content places ever increasing load on the underlying infrastructure. Without any adaptation in the user application and/or advanced traffic control, insufficient network resources can cause network congestion. This can lead to video quality degradations that manifest as video artifacts or frame freezing during playback. This is particularly critical in unmanaged best-effort networks where there are growing numbers of concurrent video streaming applications from a variety of different user devices.

One of the ultimate goals in future multimedia networks is to provide *a user-centric fair-share* of network resources, so that the *user Quality-of-Experience (QoE)* is *maximised for all users* in a network. There is a strong need to ensure QoE fairness across different devices in a network-wide manner, i.e. for devices that are sharing a bottleneck. Examples include residential, campus, and corporate networks, as well as publicly available hotspots.

Recent developments in scalable video streaming [22] and adaptive streaming [13] facilitate the dynamic adjustment of the streaming bitrate to minimise video pauses and buffering times, and ultimately improve the overall user experience. However, these types of video adaptation present three significant problems. Firstly, they are unstable and bursty in nature, especially when competing with other video clients and associated flows [1, 9]. Secondly, each video application is free to employ its own adaptation strategy, potentially leading to further network congestion. Thirdly, video adaptation occurs based upon each client's perspective. This can be problematic because the client has no knowledge of other clients on the same network and typically aims to selfishly maximise its own QoE. Such selfish behaviour could potentially lead to worsened QoE for multiple clients due to increased network congestion [1, 9, 12].

This paper introduces an OpenFlow-assisted QoE Fairness Framework (QFF) that aims to optimise the QoE for all video streaming devices in a network, whilst also taking into consideration various device and network requirements. QFF is designed to monitor video streams in a network and, in conjunction with a client control plane, dynamically adjust video flow characteristics to ensure network-wide QoE fairness. QFF accounts for both the user's, device-based, requirements and the current status of the network. The implementation and evaluation of the QFF is conducted using MPEG-DASH and OpenFlow, two promising standards for audio-visual content distribution and network management in future multimedia networks, respectively.

The remainder of the paper is organised as follows. Section 2 provides the background of this work, whilst the motivation and related work is presented in Section 3. Section 4 introduces QFF and our implementation. Evaluation is shown in Section 5 and finally, Section 6 discusses our future work and concludes the paper.

2. BACKGROUND

2.1 Adaptive Bitrate Video Streaming

Adaptive bitrate video streaming aims to support the increasing requirements of future multimedia networks. By dynamically adapting the bitrate of video during playback, it is possible to minimise video interruption and buffering times. By taking available bandwidth into consideration, a client can request the highest possible video quality. This should translate in a better experience for the user. Currently, there are a number of solutions that specifically support dynamic bitrate adaption, such as HDS by Adobe, Smooth Streaming by Microsoft and HLS by Apple. In this paper, we will be working with an implementation stan-

dardised by the Moving Pictures Experts Group under the name of MPEG-DASH or simply, DASH (Dynamic Adaptive Streaming over HTTP) [13]. This was chosen due to its vendor-agnostic design, that will see implementations on a wide range of devices and operating systems in the future.

Adaptive bitrate video streaming is almost exclusively delivered using HTTP. Different bitrate encodings of the same content are fragmented into fixed time *chunks*. These are then listed in a Media Presentation Description (MPD) or manifest. For any given time during the video's prescribed length, a number of different bitrate options are available. Swapping between the available bitrates is as simple as an application requesting a chunk with a different bitrate.

2.2 Software Defined Networking

Software Defined Networking (SDN) is a new, very promising, networking approach that facilitates the decoupling of the control plane in a network (i.e. the decision making entity) from the data plane (i.e. the underlying forwarding system). OpenFlow [17], currently the prominent SDN protocol, defines the communication between the Layer 2 networking devices (i.e. switches) and the controller of the network. OpenFlow allows experimenters, researchers, protocol developers or network administrators to exploit the true capabilities of a network in a quick, easily deployable and flexible manner. With the centralised network perspective that OpenFlow provides through its controller, an experimenter has an overarching view of the current status in the network. In addition, they have the ability to introduce, at run-time, new functionality without having to specifically modify any of the networking devices. OpenFlow's recent popularity is in part due to its open and vendor-agnostic nature. OpenFlow provides powerful and flexible tools to both network administrators and developers, and enables the implementation of a diverse range of functionality and network behaviour.

3. MOTIVATION AND RELATED WORK

Using DASH, a user device is offered a set of representations to fit their screen resolution. Each representation defines the bandwidth required to stream that bitrate of transcoded video content. This enables a client to optimise the video streaming based upon available bandwidth. A direct impact of video transcoding using limited bitrates is image compression loss. Research work in the field of video quality analysis shows that there is no linear correlation between the bitrate of a video stream and its perceptual quality [3]. In order to optimise the efficiency of network resource allocation whilst maintaining a satisfactory level of user experience, it is essential to quantify the non-linear mapping (in the form of utility functions) between bitrates and perceived video quality [3].

In an environment of heterogeneous user devices, such as a household or a campus network, efficient allocation of network resources is further complicated. The ultimate goal of resource sharing is not only optimising the QoE for a single user application, but rather achieving user-level fairness between relevant applications on multiple user devices. However, achieving such goal for live or on-demand HTTP streaming applications is non-trivial due to two shortcomings in current DASH implementations. Existing DASH-capable applications can suffer from instability and unfair network resource sharing. These are consequences of using HTTP as a transport layer which creates a mismatch between client behaviour and TCP, as we describe below.

Both TCP (the transport protocol employed by HTTP) and current DASH-capable implementations adapt their network usage based on feedback they get from it. TCP and DASH are, however, different in ways other than belonging to different networking layers. TCP aggressiveness is controlled by the sender, while it is the receiver who throttles the traffic in existing DASH clients. Another difference is their objectives: TCP aims to increase bandwidth utilisation whilst avoiding congestion through acting as a "good network citizen"; a DASH client is much more user-centric, aiming to ensure uninterrupted video playback by prefetching and buffering sufficient video chunks.

This mismatch allows a DASH client to continuously inflate its receiver window during ON periods. This inadvertently forces the sender to burst as much traffic as possible on to the network, until either enough video chunks are buffered at the client (which then switches to OFF mode), or until the sender incurs TCP packet loss. This behaviour causes extremely bursty traffic and TCP inefficiency, as connections are repeatedly restarted between ON/OFF periods, resulting in unstable video playback quality and unfair sharing of network resources [2, 12].

Another consequence of using HTTP as a transport protocol is a disruption in the feedback loop from the network. Current DASH implementations employ their own client-based bandwidth estimation tool which has been reported to yield inaccurate measurements [11], another reason for unstable and suboptimal selection of streaming bitrates [12].

Ultimately, instability diminishes user engagement [8, 18, 15], and inefficient network usage induces instability to other DASH users sharing the same network [1, 9]. Hence, there has been several efforts to readjust the imbalance between TCP and typical DASH behaviour. Most of these efforts have focused on altering the DASH client to improve its sensitivity to the state of the network. One solution, put forward in [12], is to have some cross-layer interaction between TCP and HTTP in order to provide the streaming application with better metrics and to allow TCP to reach steady-state. This would indeed improve TCP performance, but would not control the ON/OFF nature of DASH-style applications. Furthermore, it would not attain network-wide fairness across all devices. Tian and Liu [21] use throughput prediction algorithms to attenuate video rate fluctuations. Mansy et al. [16] have shown that DASH's bursty nature leads to excessive queuing in the network (a phenomenon commonly referred to as bufferbloat [10]) and proposed to adjust DASH's buffering behaviour to keep the size of the client's receiver window low. FESTIVE [14] attempts to improve fairness, stability and efficiency through using a DASH player with a stateful, delayed bitrate update mechanism. However, the outcome is client specific and cannot be easily adapted to achieve similar goals across multiple clients.

Efforts involving a proxy include the following. The QAVA architecture [4] was developed to be aware of the user's network usage quota and select video streaming quality in order to achieve maximum QoE whilst having control on network usage. QDASH [18] uses a non-aggressive probing technique to estimate available bandwidth and accordingly provide a gradual change in video quality.

However, renegotiation of network resource allocation is triggered every time a user application joins or leaves the

network. It may take from several seconds to minutes for the remaining user applications to renegotiate resources through competition, or they may not reach a stable state at all [1]. To this end, some proposals have been put forward for a network control plane. An in-network device has a good view of the network resource utilization per user application and can thus introduce agile resource renegotiation in order to attain relative fairness and maximum QoE across all clients. The in-network device is also in a better situation to select the best CDN to achieve high streaming bandwidth based on the geographical location of the network and the time of day. Efforts in this direction include the video control plane introduced in [15], which relies on the client periodically reporting playback session metrics (buffering state, bitrates, etc.) in addition to location, ISP and CDN information. Despite the significant reduction in rebuffering times introduced by the proposed system, the level of information required could be prohibitive in some scenarios.

4. QOE FAIRNESS FRAMEWORK

We introduce an OpenFlow-assisted QoE Fairness Framework (QFF) that fairly maximises users' QoE in multimedia networks. QFF employs OpenFlow that allows vendor-agnostic functionality to be implemented for network management and active resource allocation. With the help of OpenFlow, QFF monitors the status of all the DASH video applications in a network and dynamically allocates network resources to each device. This allocation ensures that the QoE of all video streams on even heterogeneous user devices is optimised to achieve the maximum user-level fairness.

QFF avoids user-agnostic network management, where bandwidth is blindly divided between active user sessions. Such a management approach leads to unfairness in terms of the experience users receive given different device requirements, as both our work (Sections 4.1 and 5) and related work show [1, 9, 12]. The underlying reason is at a single bitrate (delivered by equal bandwidth sharing), a device of higher resolution, such as an HD IPTV, would receive a significantly lower QoE compared to a device with less capabilities, such as a smartphone. Instead, the core of QFF is a bandwidth allocation algorithm, which is designed to seek the optimal representations for each video application based on bitrate to QoE utility functions.

Figure 1 presents a high-level view of the QFF. At its core there is an OpenFlow Module (OM), running on the OpenFlow controller of the network. This is responsible for orchestrating the main functionality of the QFF. Logically, the QFF is also composed of three additional parts :

1. **Input** : The Network Inspector and the MPD Parser provide the network and client's status as input to the core OM. The Network Inspector informs the OM of the number of devices in the network, the streaming bitrate each device is currently requesting and the available network capacity. The MPD Parser informs the OM as to the specifics of the clients' video requests, such as the duration and available encoding bitrates of a requested video file, and the number and size of its chunks. Both the Network Inspector and the MPD Parser use the OpenFlow protocol to capture this information from the OpenFlow switches in a network and pass them to the OM.
2. **Intelligence** : The Utility Functions and the Optimisation Function (described in Sections 4.1 and 4.2, respectively) interact with the OM to dynamically optimise QoE fairness at each point in time. In essence, each Utility Function provides a model that maps the bitrate of a particular video to the QoE delivered on that device. The Optimisation Function finds a set of bitrates for each streaming video in the network that results in equivalent QoE level for all devices according to the specific Utility Functions.
3. **Output** : The Flow Tables Manager and the DASH Plugin ensure that the decisions of the OM are being appropriately propagated to the network. In particular, the Flow Tables Manager adds the appropriate flows to the OpenFlow switches, so that each client receives the requested video stream. The DASH Plugin uses the OM's Intelligence to inform all DASH clients of the bitrate that they should now be requesting to achieve network-wide QoE fairness. QFF is not restricted to DASH and allows other adaptive video streaming technologies to be plugged into it.

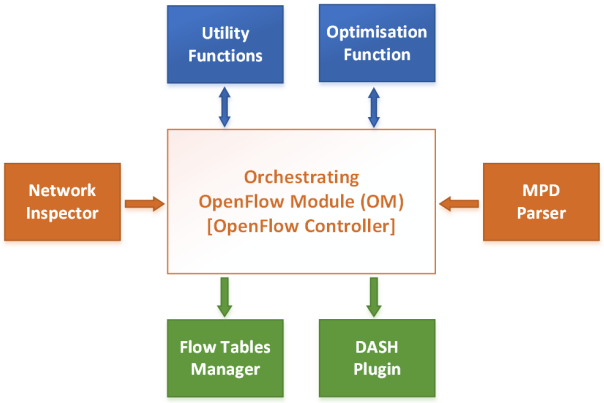


Figure 1: OpenFlow-assisted QoE Fairness Framework

4.1 Utility Functions

QFF's Utility Functions provide a model that maps the bitrate of a video at a particular resolution and the QoE perceived by the user. For wide adoption of QFF, a database consisting of a Utility Function per video at each resolution would need to be constructed.

To evaluate the feasibility of our design, we selected a reference source video file of an animated film called "Big Buck Bunny"[1]. This film is widely used by researchers in the area of adaptive content distribution. We acquired the uncompressed YUV video files in 360p, 720p and 1080p resolution and the FLAC audio file. Then, we used FFMPEG to encode the source files using the H.264/AAC audio-visual codec and MPEG4 encapsulation for direct HTML5 video playback using DASH-JS [7] (a DASH implementation). These resolutions are selected to represent three typical user devices; smartphone, tablet/PC and HD IPTV respectively. Although many modern devices are equipped with higher resolution screens and corresponding computing resources, the three selected resolutions encompass a significant number of use cases. We generated 22 test video sequences with various predefined bitrates for each resolution, with respect to practice, as shown in Table 1. In addition, a lossless ver-

[1]http://www.bigbuckbunny.org/

sion of the encoded video was also generated as a reference for each resolution.

In order to measure the quality of the encoded test sequences, we employed objective video quality assessment models. The Structural Similarity Index (SSIM) uses a functional model of the Human Visual System (HVS) by combining local luminance, local contrast and structure comparison [23]. The Video Quality Metric (VQM) is also a HVS-based objective model that measures the perceptual effects of video impairments [24]. The video quality of all our test sequences was measured by evaluating the perceptual loss of every test video sequence to the lossless encoded reference video using SSIM and VQM. Examining the assessment results from both metrics reveals an extremely high absolute Pearson correlation R of 0.9912 (R^2=0.9825). Therefore, it was decided to adopt SSIM as the quality metric to create our Utility Functions.

The scatter plot showing the mapping of bitrate to video quality using SSIM (Figure 2) presents our two fundamental principles. First, the relationship between bitrate and perceptual quality is not linear; as the bitrate increases, the gain in video quality is gradually saturated. Second, the equal division of network bandwidth for video steams of different resolutions (i.e. a vertical line representing a certain bitrate) results in unfair video quality levels as perceived by end-users. In order to generalise the bitrate to video quality mapping and derive a set of parameters for our QFF's Utility Functions, we conducted curve fitting using a number of general models. The general model Power2 has proven to be the most suitable function (Table 2, Figure 2) providing a high level of goodness of fit for all three selected resolutions whilst keeping a low computational complexity for the Optimisation Function.

Resolution	**Video Bitrate (kbps)**
1080p	100, 200, 600, 1000, 2000, 4000, 6000, 8000
720p	100, 200, 400, 600, 800, 1000, 1500, 2000
360p	100, 200, 400, 600, 800, 1000

Table 1: Set Bitrates for Three Video Resolutions

General Model For	**Power2** $f(x) = ax^b + c$			**Goodness of Fit**	
	a	b	c	Adjusted R^2	RMSE
1080p	-3.035	-0.5061	1.022	0.9959	0.006011
720p	-4.85	-0.647	1.011	0.9983	0.002923
360p	-17.53	-1.048	0.9912	0.9982	0.002097

Table 2: Model and Coefficients for Utility Function

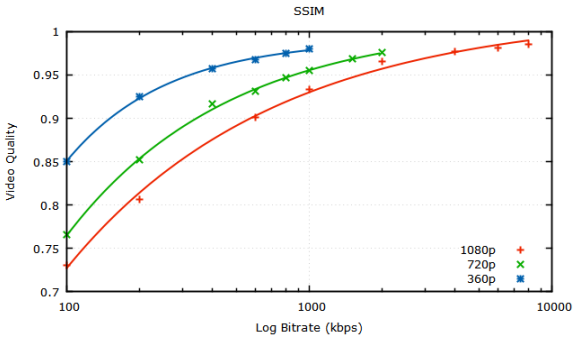


Figure 2: Scatter Plot and Derived Utility Function

4.2 Optimisation Function

The Optimisation Function uses the models that the Utility Functions provide in order to find the optimum set of bitrates that ensures QoE fairness across all DASH clients in the network. For each resolution i, there is a Video Quality (VQ) function $f_i(x_i)$, where x_i represents the bitrate the video is encoded in. To find the optimum set of bitrates for all network devices, we need to find the value $p = max[min_i(f_i(x_i))]$ such that $\sum x_i \leq B$, where B is the total amount of bandwidth available in the network. Since the $f_i(x)$ are strictly increasing functions, the above problem is equivalent to solving $\sum f_i^{-1}(p) = B$, where $f_i^{-1}()$ is the inverse of $f_i()$ satisfying $f_i^{-1}(f(x)) = x$. Essentially, this is finding the root of a univariate equation, which just requires a simple line search assuming that $f_i()$ are continuous.

However, $f_i()$ are not continuous: videos are made available only at any of a finite set of encoding bitrates (Table 1). Hence, finding the optimum p is not a straightforward process. We decided on a branch and bound algorithm [6] that uses heuristic evaluation that allows us to optimise over a discrete data set in linear time.

Our starting criterion is solving for p using the continuous functions of Table 2 that fit the VQ curves and the bandwidth constraint as stated above. This provides theoretically optimum bitrates x'_i, which have to be downgraded to the maximum discrete values x_i where $x_i \leq x'_i, \forall i$. QFF supports having pluggable algorithms for enforcing different optimisation policies. For our proof-of-concept design, we implemented two algorithms. Both algorithms calculate $\Delta x = \sum x'_i - x_i$ and try to maximize additional VQ that can be obtained from this surplus Δx. The first algorithm, *Promote*, tries to upgrade user a with the minimum VQ, i.e. $min(f_i(x_i)) = f_a(x_a)$, from bitrate x_a to x_{a+1}, if $x_{a+1} - x_a \leq \Delta x$. This is sequentially repeated until no longer possible. The second algorithm, *Boost*, has the same stopping heuristic but starts with upgrading user b with the least amount of needed bandwidth till its next bitrate, i.e. $min(x'_i - x_i) = x'_b - x_b$. Both algorithms have modest computational overhead, requiring $\leq$0.3s for optimising 100 Utility Functions with 10 different bitrates each.

5. EVALUATION

In order to evaluate QFF we considered a home networking scenario. In this scenario, users are connected to a home gateway and are accessing video content on three different DASH-enabled devices. These devices support different resolutions, namely, HD TV (1080p), Tablet (720p) and Smartphone (360p).

5.1 Testbed Setup

Figure 3 depicts our testbed setup that recreates the aforementioned home environment. We use off-the-shelf home networking equipment (TP-LINK WR1043ND) that supports Pantou [20], an OpenFlow implementation. We then attach three clients to it, each representing a different device type. The OpenFlow switch is then connected to the Internet so that each client may access DASH content hosted externally. An additional machine is used to provide an OpenFlow controller that runs our QFF. We also throttle the external link to 8Mbit/s (approximately the UK home average bandwidth [19]) to emulate a home networking scenario as accurately as possible.

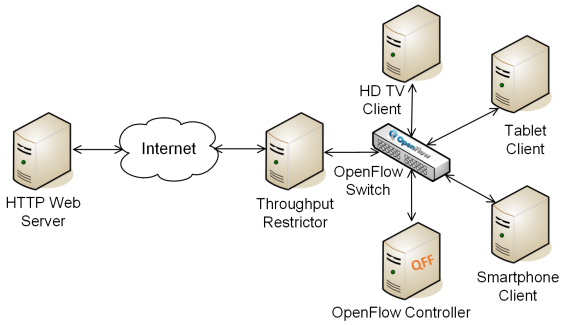


Figure 3 : Testbed Setup

5.2 Experiments

In order to effectively evaluate QFF, we run a number of experiments upon our testbed using three different approaches. Firstly, we use DASH-JS [13], an unmodified DASH client. Secondly, we use OpenFlow to manage the allocation of the available bandwidth equally between active users (*EqualBW*). Thirdly, we use QFF to run *Promote* (*Boost*'s evaluation is not shown for space, but its results are similar to that of *Promote*). For each experiment, we measure the video bitrates and associated QoE level of each user, as well as the overall network utilisation. The first experiment illustrates what user applications experience under the inherent DASH behaviour. The second experiment is a control to demonstrate what is gained by equally allocating network resources via an OpenFlow-based control plane. The third experiment shows the contribution of QFF in improving user-level QoE fairness with the use of OpenFlow. In each run, the HD TV client (1080p) starts playing at time $0s$. The smartphone (360p) starts its DASH session at $30s$, followed by the tablet device (720p) at $60s$. Playback is continued for another minute beyond that point in time.

The results of our experiments are now presented. Figure 4 portrays the bitrate of the downloaded video chunks over the duration of each experiment run. Figure 4a shows one out of 10 runs we carried out, during which there were 23 changes in user bitrate. The number of DASH bitrate changes across our experiments varies from 18 to 31, with an average of 23. For *EqualBW* and *Promote*, all runs result in the same changes, i.e. 2, depicted in Figures 4b and 4c.

We note in Figure 4a, where DASH-JS is allowed to resort to its own adaptation, an unmistakable instability in the bitrate of the video stream even before the smartphone starts streaming. Another interesting observation is that the HD TV user sacrifices the most in competition over network resources, whilst the other two devices (of lesser capability) seem to be content bitrate-wise. The HD TV user never reaches $8000kbps$ which the network can sustain before $30s$. We attribute this to poor bandwidth estimation, something that has already been highlighted (see Section 3). When bandwidth fairness is enforced, as depicted in Figure 4b, the instability caused by the inaccurate estimation of available bandwidth is immediately rectified, allowing the clients to continue smooth playback for longer periods. However, strict bandwidth fairness is oblivious to the needs of the user applications. This results, between 0 and $30s$, in allocating half of the available bandwidth between the two active users. In other words, the HD TV runs at $4000kbps$ and the smartphone at $1000kbps$. The smartphone user in this instance is achieving maximum QoE as he is receiving the maximum bitrate possible for the resolution of his device. However, the HD TV is not achieving maximum possible QoE. This is resolved by our QFF (refer to Figure 4c). *Promote* uses the SSIM Utility Functions to factor in the device-specific relationship between bitrate and QoE, and adapts the bandwidth allocation policy to achieve maximum QoE for all users. *Promote* also achieves improved stability, as *EqualBW* does.

The bitrate figures however do not indicate what is perceived by the user consuming the video, and hence they do not tell the whole story. To study the range of QoE levels delivered by each approach, we present Figure 5. This shows the mean value as well as the range of QoE levels delivered per user type and experiment, averaged across our experiments. Conforming to our previous findings, we observe a distinct instability in the QoE delivered by DASH clients. This is especially true as the user's network requirements increase: signified by diminished mean and inflated variance in QoE for the HD TV and the tablet. Stability is improved using the network plane for equal bandwidth sharing. It is improved even further with our QFF, which produces in-

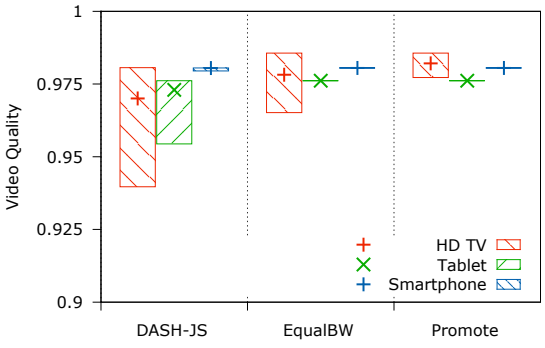
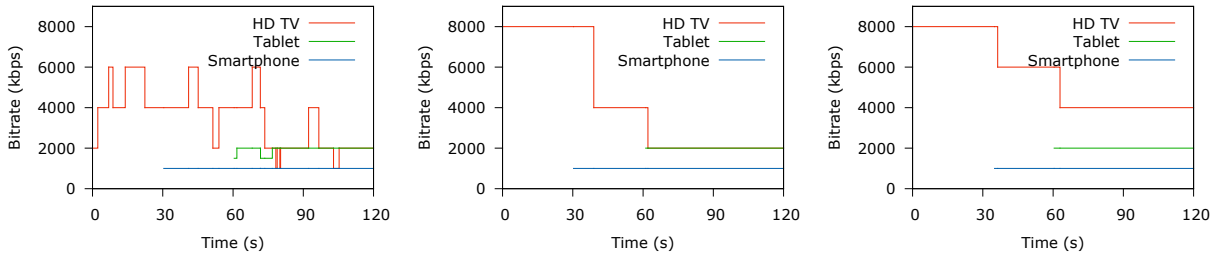


Figure 5 : Variance in QoE



(a) DASH-JS (b) Bandwidth Fairness (c) Promote Algorithm

Figure 4 : Bitrate Stability

creased mean QoE and reduced QoE variance, particularly for the HD TV user.

6. FUTURE WORK AND CONCLUSION

The Utility Function presented in this paper (Section 4.1) is dependent upon the characteristics of the test source video, as it was used to evaluate the feasibility of our design. In order to increase the applicability of QFF for Internet streaming, we would need to create either a Utility Function that is more generic and covers video content of all natures, or generate Utility Functions that incorporate video content metrics, such as motion and complexity. We plan to do this as future work, in addition to evaluating QFF in a more widely scoped scenario with enterprise-grade OpenFlow switches. Furthermore, subjective user studies will also be conducted as a feedback loop for our future development.

In this paper we proposed a new framework to achieve user-level fairness in an adaptive video streaming environment. QFF focuses on the requirements of future multimedia networks and optimises the QoE of multiple competing and heterogenous clients. Our approach leverages the advantages of OpenFlow to overcome the one-sided client perspective of video applications, and offer network-wide QoE fairness. The results of our experiments demonstrate that QFF provides network stability and optimises video streaming QoE across heterogeneous devices in a network.

7. REFERENCES

[1] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. Begen. What Happens When HTTP Adaptive Streaming Players Compete for Bandwidth? NOSSDAV, 2012.

[2] S. Akhshabi, A. Begen, and C. Dovrolis. An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP. In *Proceedings of the 2nd annual ACM Conference on Multimedia Systems*, MMSys '11, pages 157–168, New York, USA, 2011.

[3] G. Cermak, M. Pinson, and S. Wolf. The Relationship Among Video Quality, Screen Resolution, and Bit Rate. *IEEE Transactions on Broadcasting*, 57:258 – 262, 2011.

[4] J. Chen, A. Ghosh, J. Magutt, and M. Chiang. QAVA: Quota Aware Video Adaptation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 121–132, New York, NY, USA, 2012. ACM.

[5] CISCO. The Zettabyte Era. Technical report, 2012.

[6] R. J. Dakin. A Tree-search Algorithm for Mixed Integer Programming Problems. *The Computer Journal*, 8(3):250–255, 1965.

[7] DASH-JS: A JavaScript-based DASH library for Google Chrome. `http://www-itec.uni-klu.ac.at/dash/?page_id=746`.

[8] 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. *SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.

[9] J. Esteban, S. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac. Interactions Between HTTP Adaptive Streaming and TCP. NOSSDAV 2012.

[10] J. Gettys and K. Nichols. Bufferbloat: Dark Buffers in the Internet. *Queue*, 9(11):40–54, November 2011.

[11] O. Goga and R. Teixeira. Speed Measurements of Residential Internet Access. In *Passive and Active Measurement*, pages 168–178. Springer, 2012.

[12] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, Rimid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proceedings of the 2012 ACM conference on Internet Measurement*, pages 225–238. ACM, 2012.

[13] ISO-IEC 23009-1:2012 Information technology. Dynamic Adaptive Streaming over HTTP (DASH), 2012.

[14] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 97–108, New York, NY, USA, 2012. ACM.

[15] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A Case for a Coordinated Internet Video Control Plane. In *Proceedings of the SIGCOMM 2012 Conference on Applications, Technologies, Architectures and Protocols for Computer Communication*. ACM, 2012.

[16] A. Mansy, B. Ver Steeg, and M. Ammar. SABRE: A Client based Technique for Mitigating the Buffer Bloat Effect of Adaptive Video Flows. In *Proceedings of the 3rd annual ACM Conference on Multimedia Systems*, MMSys '12, New York, NY, USA, 2012. ACM.

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

[18] R. Mok, X. Luo, E. Chan, and R. Chang. QDASH: a QoE-aware DASH system. In *Proceedings of the 3rd annual ACM Conference on Multimedia Systems*, MMSys '12, pages 11–22, New York, USA, 2012.

[19] Ofcom. Overview of UK Broadband Speeds. `http://stakeholders.ofcom.org.uk/market-data-research/other/telecoms-research/broadband-speeds/bb-speeds-nov-11`.

[20] Pantou:. OpenFlow 1.0 for OpenWRT. `http://www.openflow.org/wk/index.php/Pantou_:_OpenFlow_1.0_for_OpenWRT`.

[21] G. Tian and Y. Liu. Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 109–120, New York, NY, USA, 2012. ACM.

[22] I. T. Union. H.264: Advanced Video Coding for Generic Audiovisual Services (Part 10), 2003.

[23] Z. Wang, L. Lu, and A. C. Bovik. Video Quality Assessment based on Structural Distortion Measurement. *Signal Processing : Image Communications*, 19(2):121–132, 2004.

[24] S. Wolf and M. H. Pinson. Spatial-temporal Distortion Metric for in-service Quality Monitoring of any Digital Video System. In *Proceedings of SPIE*, volume 3845, page 266, 1999.