2016 IEEE 41st Conference on Local Computer Networks

The Impact of Active Queue Management on DASH-based Content Delivery

Jonathan Kua, Grenville Armitage, Philip Branch
Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
{jtkua, garmitage, pbranch}@swin.edu.au

***Abstract*—With Netflix and YouTube accounting for more than 50% of North American, fixed network peak download traffic in 2015, video streaming is a significant source of Internet traffic. Dynamic Adaptive Streaming over HTTP (DASH) is a recent standard for live and on-demand video streaming services, where clients adapt their behaviour on-the-fly to match regularly updated estimates of network capacity. Consumer DASH streams are likely to be bottlenecked by last-mile ISP links, and impacted by emerging active queue management (AQM) schemes being deployed to counter bufferbloat. We experimentally characterise and evaluate the impact of bottlenecks utilising PIE, FQ-PIE, CoDel and FQ-CoDel AQM schemes on DASH streams. We show that PIE's higher burst tolerance provides better streaming quality for single DASH stream over moderate to high RTT paths and when coupled with a FlowQueue scheduler's flow isolation capabilities, FQ-PIE protects DASH streams in the presence of cross-traffic.**

***Index Terms*—DASH, TCP, AQM, PIE, FQ-PIE, CoDel, FQ-CoDel**

I. Introduction

A typical Internet-enabled home sees a range of applications competing for the last-mile link connecting the home to the outside world. Many applications rely on Transmission Control Protocol (TCP) to ensure reliable delivery with reasonable sharing of network capacity. However, the last-mile is also a bottleneck. In recent years vendors have sought to minimise packet losses by oversizing the packet buffering in their home gateway products, leading to a phenomenon known as *bufferbloat* [1]. Since loss-based TCP flows cyclically fill and drain buffers of bottleneck nodes, bufferbloat leads to significantly inflated round trip times (RTTs) for all traffic concurrently sharing a bottleneck point. To mitigate the effects of bufferbloat, new active queue management (AQM) schemes allow bottlenecks (such as home gateways) to support short bursts of traffic while keeping long-term queuing delays low. PIE (Proportional Integral controller Enhanced [2]), CoDel (Controlled Delay [3]) and FQ-CoDel (FlowQueue-CoDel [4]) are three specific schemes being developed in the IETF AQM working group [5]. PIE is already part of the DOCSIS 3.1 standard for cable modems [6], and FQ-CoDel is being heavily promoted for embedded Linux gateways.

Video streaming has become a significant source of Internet traffic, with companies such as Netflix and YouTube accounting for more than 50% of the peak download traffic on fixed networks in North America in 2015 [7]. Streaming services are predominantly layered on top of HTTP to improve traversal of firewalls, NATs and other middleboxes. Dynamic Adaptive Streaming over HTTP (DASH) is a recent ISO standard for content delivery [8], with Netflix and YouTube adopting DASH-like streaming strategies [9]. DASH offers content streaming from commodity web servers or caches, while adapting to a heterogeneous community of client devices and dynamically varying network conditions.

HTTP typically runs over TCP[1], so an end-user's experience of content streaming critically depends on how well DASH adaptive bitrate (ABR) algorithms respond to TCP's own utilisation of the network path between the server and client. This in turn depends on the choice of server-side TCP algorithm, the server-client RTT, and how competing network traffic flows are handled at bottlenecks along a network path. AQMs are emerging as important regulators of congestion and fairness at bottlenecks. Despite the significant work already done on client-side ABR, there has been only limited exploration of how DASH traffic interacts with modern AQMs.

In this paper we make the following key contributions:

- Evaluate the impact of PIE, FQ-CoDel and a novel "FQ-PIE" hybrid AQM scheme on DASH-based multimedia streaming traffic over a range of network RTTs
- Show that FQ-CoDel and FQ-PIE can protect DASH streams from collateral damage by cross-traffic, and combining FlowQueue scheduling with PIE's burst tolerance allows FQ-PIE to provide better DASH performance

The rest of this paper is structured as follows. Section II provides background on DASH, TCP, AQMs and their likely interactions. We describe our use case model and testbed setup in Section III. Section IV evaluates the impact of different AQMs on DASH streams across links with various path characteristics. Section V concludes and outlines future work.

II. Background and Related Work

In this section we summarise the technologies involved in DASH-based content streaming into home environments.

A. DASH

Modern Internet video streaming technologies have adopted DASH-like streaming strategies. In DASH systems, multi-

[1]With QUIC an emerging alternative, https://www.chromium.org/quic

DOI 10.1109/LCN.2016.24

media content is first encoded into multiple versions at different discrete bitrates (*representation rate*), with each version then fragmented into small multi-second video segments (*chunks*) [10]. Chunks encoded at different bitrates are aligned in the video time line so that the client can smoothly switch representation rates at chunk boundaries. Chunks are served using standard HTTP servers, while the underlying TCP layer regulates actual transmission of packets within each chunk.

The server provides a Media Presentation Description (MPD) file to describe the available chunks and associated encoding (representation) rates for each offered piece of content. DASH clients may implement a variety of *adaptive bitrate* (ABR) algorithms to retrieve chunks encoded at the highest *representation rate* (RR) believed supportable by recently observed network conditions.

To minimise the number of stalls in content playback (rebuffering events) and associated degradation in end user experience [11], a DASH client will typically pre-fill a local playout buffer with multiple chunks before initiating local playback. After the buffer pre-filling phase, the client enters a steady-state, periodic ON-OFF chunk retrieval phase, requesting video chunks every x seconds (x is typically the chunk size). Adaptive chunk retrieval helps maintain a backlog of content in the playout buffer as network conditions fluctuate.

The fundamental objective of an ABR is to select the RR of future chunks to maximise user-perceived Quality of Experience (QoE) [12] given estimated network conditions. Frequent or sudden changes in RR, mid-stream re-buffering events and a long start-up time all degrade QoE.

Although crucial to positive end-user experience in a heterogenous Internet ecosystem, the DASH specification does not recommend any particular ABR algorithm. Consequently, ABR designers have explored many strategies for good user experience under varying network conditions. Most existing ABRs are based on one of the following three strategies:

- *Throughput*: The *achieved rate* (AR)[2] estimated from recent chunk downloads drives selection of the next representation rate, as long as network conditions do not change. For example, when AR is high, the client retrieves future chunks encoded at a higher RR to improve end-user video quality. When AR is low, the client retrieves future chunks encoded at a lower RR to avoid playout-buffer under-run [13], [14], [15].
- *Buffer occupancy*: Select RR of next chunk based on application layer playout buffer occupancy [16], [17].
- *Control theory*: Complex, hybrid algorithms that use both throughput estimates and buffer occupancy as indicators of network conditions, and repeatedly (re)solve a control theory/stochastic optimal control equation for selecting the representation rate to satisfy user preferences [18], [19], [20], [21].

Regardless of ABR strategy, QoE ultimately depends on how DASH interacts with the underlying network.

[2]Length of a chunk divided by the time taken to receive it (essentially an estimate of per-chunk TCP throughput).

B. Impact of TCP under DASH

The timeliness (rate) of chunk delivery can vary widely depending on the speed of bottleneck links along the server-to-client path, the RTT of the path, the queue management at any bottlenecks and the nature of other short-term or long-term flows (cross-traffic) competing for path capacity. The use of TCP also means a DASH stream may cause collateral damage to other traffic sharing a network layer bottleneck [22].

The DASH server's TCP stack maintains a running estimate (the congestion window, `cwnd`) of how many bytes can be 'in flight' and unacknowledged at any given time. `cwnd` starts low[3], grows as packets are received and acknowledged by the client, and shrinks when packets are lost or the connection goes idle for too long [23]. A DASH client's chunk retrieval process means TCP sends repeated bursts of packets followed by idle periods. If `cwnd` is too low, or fails to grow quickly enough, the DASH client experiences low AR per chunk. If `cwnd` grows beyond the path's bandwidth-delay product (BDP) it starts filling bottleneck queues and inflicting additional queuing delays on all flows sharing the bottleneck.

Much work has been done to model and understand how application rate-limited traffic affects TCP behaviour. Recently, a revised proposal on congestion window validation (CWV) has been submitted to IETF (RFC7661 [24]). DASH traffic is a classic example of application rate-limited traffic with a bursty nature, creating an ON-OFF traffic pattern. Nazir et al. [25] evaluated the effects of TCP CWV and TCP pacing on DASH transport.

Huang et al. [26] dissected the interactions between TCP and adaptive video traffic by analysing traffic generated by major streaming services, and identified a vicious cycle they named a *"downward spiral"*. During multi-second OFF periods, TCP resets `cwnd` to its initial value due to inactivity longer than the current retransmit timeout (around 200ms in their experiments). Consequently `cwnd` ramps up in slow start mode to retrieve each new chunk. With no competing flows, the client still selects the highest sustainable RR. However, competing flows fill the bottleneck buffer during OFF periods, causing high packet loss and low video throughput (low AR) for subsequent chunks. As AR drops further, the client requests chunks of lower RR and shorter length. As chunks get shorter, TCP has less time to reach its fair share before it finishes downloading each chunk, leading to further underestimation of available capacity – a detrimental downward spiral.

C. Active Queue Management (AQM)

The proliferation of oversized first-in-first-out (FIFO) buffers in network devices (aka *bufferbloat* [1]) has recently reinvigorated IETF interest in new AQM schemes [5], revisiting an idea present since at least the late 1990s (e.g. RFC 2309 [27]). AQM schemes generally aim to provide congestion notifications[4] long before the queue fills, greatly reducing the

[3]The default was 2 packets, but some recent OSes now start TCP connections with a window of 10 packets.

[4]By packet drops or Explicit Congestion Notification (ECN) marking.

queuing delays introduced by classical FIFO (or *tail drop*) queue management.[5] Here we briefly summarise PIE, CoDel, FQ-CoDel and a novel FQ-PIE hybrid.[6]

1) PIE and CoDel: These two schemes operate on single queues, and differ from earlier AQM schemes (such as RED [27]) in that they explicitly aim to keep queuing delays low, rather than simply aiming to keep queue occupancy low.

With PIE an average dequeue rate is estimated based on the standing queue size. The latest PIE [2] has an initial burst allowance of 150ms which allows any number of packets that arrive within the first 150ms to be served successfully. After the initial burst, up to T_{target} (default 15ms) of queuing delay is allowed in the queue at any time. Dequeue rate is recalculated as every packet leaves the queue, and is used to calculate the current delay, which is then periodically used to calculate the probability that a packet is dropped (or marked) on arrival. This probability is adjusted based on the trend of the delay (up or down). When the probability exceeds 10%, packets are dropped even if Explicit Congestion Notification (ECN) marking is enabled.

CoDel uses the local minimum packet-sojourn time T_{target} (default 5ms) in the queue as a measure of the standing/persistent queue. As long as the minimum queue delay is less than an acceptable queue delay target, or the buffer contains fewer than one full-size packet worth of bytes, packets are not dropped (or ECN marked). When the minimum queue delay has exceeded the target for a time greater than a specific *interval* $T_{interval}$ (default starting at 100ms), CoDel enters a dropping mode where packets are dropped/marked according to a control law aimed at producing a linear change in TCP throughput. Once the minimum delay goes below the target, packets are no longer dropped or marked.

2) FQ-CoDel: This scheme combines flow-classification, per-queue CoDel AQM, and scheduling. Flows are hashed into one of (usually) 1024 different queues, with CoDel separately managing the sojourn times of packets passing through each queue. The queues are serviced by a form of deficit round robin scheduling, where priority is given to queues with packets from new or "recently seen" flows.

An FQ-CoDel bottleneck achieves latency reductions (due to CoDel) and relatively even capacity sharing with minimal prior configuration (due to the fixed hashing function). It provides isolation for low-rate traffic (such as DNS, web, and video conferencing traffic) while keeping queue lengths short.

3) FQ-PIE: A novel contribution of our work is evaluating the recently released FreeBSD implementation of "FQ-PIE" [28]. Whilst the idea was alluded to PIE's original developers, [28] is the first well documented combination of the FlowQueue aspect of FQ-CoDel with PIE's individual queue management. FreeBSD's FQ-PIE hashes flows into one of N queues, applies PIE on a per-queue basis and (as with FQ-CoDel) services the queues with a form of deficit round robin scheduling, prioritising queues with packets from new or "recently seen" flows.

An FQ-PIE bottleneck achieves the latency reductions and burst tolerance of PIE, with relatively even capacity sharing and flow isolation similar to FQ-CoDel. We expect its burst tolerance may provide better support than FQ-CoDel for ON-OFF traffic like DASH.

[5]Large variations in queuing delays occur as TCP's cyclical capacity probing fills and drains bufferbloated FIFO queues.

[6]PIE and FQ-CoDel are emerging from IETF consideration as deployment candidates on ISP/consumer links. FQ-PIE blends their beneficial properties.

D. DASH and AQM

The impact of AQMs in residential settings has been studied in [29]. A DASH system is controlled by two control loops, i.e, the *inner* TCP congestion control loop that reacts to network conditions thus regulating its sending rate, and the *outer* client adaptive bitrate selection loop that reacts to TCP sending rate and tries to match the video bitrate to the average TCP rate. The likely deployment of modern AQMs at the last-mile bottleneck further complicates matters, particularly the interaction between bursty DASH traffic and the burst-tolerances of different AQM schemes. To the best of our knowledge, the intricate interactions between these three layers have not been studied and evaluated experimentally in detail.

In [30] the authors evaluated the performance of various traffic workloads, including HAS (DASH-like HTTP Adaptive Streaming), FTP, VoIP and web traffic, using ns-2 to model a variety of AQMs (FIFO, Adaptive RED, CoDel and PIE) in a downstream DOCSIS 3.0 environment. They found that when five HAS flows compete with a varied number of FTP flows, AQM significantly improves HAS performance compared to drop-tail based schemes on the video playback rate. The results further suggest that CoDel leads to higher HAS adaptation rates. They also observed that PIE maintains the target queue latency more reliably than CoDel, and conjectured that differences in the burst tolerances of PIE and CoDel contribute to observed differences in client adaptation rate. (Table 12 in [30] shows that PIE achieves a slightly higher throughput than CoDel when competing with up to 11 FTP flows.)

We would expect FQ-CoDel and FQ-PIE's flow separation to help prevent the previously noted "downward spiral" effect by isolating DASH traffic from competing TCP flows.

III. Evaluation Methodology

Here we describe our use case, experimental testbed and performance measures.

A. Consumer home network environments

Our use case involves home users accessing remote content across their ISP's (usually asymmetric) broadband last-mile link – commonly the bottleneck for traffic both towards the home (*downstream*) and away from the home (*upstream*). Content servers may be located domestically (with low intrinsic RTTs) or internationally (with larger intrinsic RTTs).

1) Sharing the last-mile bottleneck: A challenge for DASH-based streaming occurs when competing traffic overlaps in time, whether due to explicit action of the end-user(s) or automated background activities launched by in-home devices.

Whether brief or long-lived, the competition for last-mile resources degrades the quality of the DASH-based service.

Competing downstream traffic directly reduces bandwidth available for inbound DASH traffic. Upstream traffic reduces the bandwidth available to carry TCP Acknowledgement (ACK) packets from the DASH client to the server, which in turn reduces the achievable DASH download rate. Examples of competing in-home traffic sources include smartphones and tablets downloading 'app' updates or synchronising data out to 'the cloud', PCs pulling down OS updates, users sending video and photos via email, and so forth.

2) Representative network conditions: The typical home will have connections to both domestic (local/intra-ISP and off-net/inter-ISP) and international end points. We represent this with base RTTs ranging from 20ms to 240ms. For competing (cross-)traffic experiments, we fix the base RTT of DASH flow(s) to 40ms and vary the base RTT of the cross-traffic from 20 to 240ms. This represents DASH traffic being served from a domestic content distribution network (CDN) while competing traffic originates from diverse locations.

Broadband last-mile services vary significantly around the world, with the majority not yet using any form of AQM. We chose to explore two scenarios – a service offering 4Mbps down / 1Mbps up (4/1Mbps), and a service offering 12Mbps down / 1Mbps up (12/1Mbps). These represent mid-range broadband services that would benefit from AQM schemes at either end of the last-mile link. The 12/1Mbps case provides capacity well in excess of the highest RR in our chosen DASH dataset (Section III-B2) while 4/1Mbps provides capacity almost equal to the highest available RR, making the DASH client far more sensitive to changes in network conditions.

B. Experimental Setup

Figure 1 shows our experimental TEACUP-based testbed [31]. The router uses either 64-bit openSUSE 12.3 Linux (kernel 3.17.4 running at 10kHz) or 64-bit FreeBSD 10.1-RELEASE to provide a configurable bottleneck between client(s) on 172.16.10.0/24 and server(s) on 172.16.11.0/24. Hosts run either 64-bit openSUSE 12.3 (kernel 3.17.4) or 64-bit FreeBSD 10.2-RELEASE-p7.

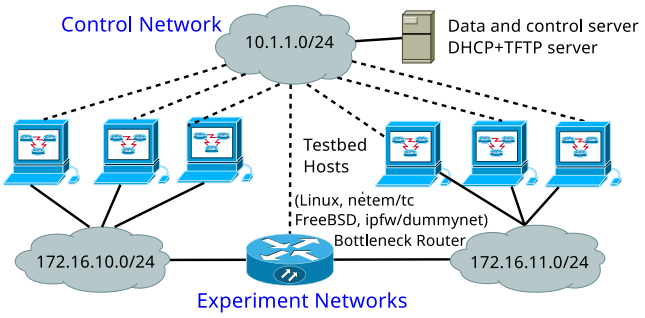


Fig. 1. Testbed overview

1) Emulating network conditions: Linux `netem` and `tc` modules provide FIFO, PIE, CoDel and FQ-CoDel queue management schemes and emulate Section III-A2's RTTs and bottleneck bandwidths.[7] For our FQ-PIE experiments, the bottleneck runs FreeBSD, with a patched `ipfw/dummynet` [28] providing rate limiting, base RTT and FQ-PIE AQM. In all cases, baseline RTT is emulated by independent delay of RTT/2 each way. Note: FQ-PIE is based on the latest PIE spec [2] while the current Linux PIE is based on an earlier draft [32], with some differences in traffic burst handling.[8]

Packets sit in a 1000-packet buffer while being delayed, then sit in a separate, configurable bottleneck buffer while being rate-shaped to the bottleneck bandwidth. The bottleneck buffer is 340 packets for FIFO experiments (to exceed each path's bandwidth delay product) and 1000 packets for PIE, FQ-PIE, CoDel and FQ-CoDel experiments.

2) Creating DASH flows and competing flows: Our DASH traffic flowed from a `lighttpd` version 1.4.35 web server[9] (with persistent HTTP connections enabled) to a BSD-licensed DASH client based on `dash.js` v2.0.0[10]. Our `dash.js` client uses a throughput-based ABR strategy, where the next RR is based on the average ARs of the previous three chunks. Long-lived competing flows were created using `iperf`.

We used FreeBSD and NewReno on the DASH server host (inspired by FreeBSD in Netflix's OpenConnect platform[11] and NewReno being similar to the default TCP in Apple's OSX and older Microsoft servers), and FreeBSD for our DASH clients We used Linux and TCP CUBIC to generate bulk transfer (`iperf`) flows, as many online services run on Linux and CUBIC is known for its effective bandwidth utilisation (which provides a good stress test for DASH's adaptive behaviour). All hosts disable ECN and enable both receive buffer auto-tuning and TCP window scaling.

TABLE I
REPRESENTATION RATES AVAILABLE IN 2-SEC DATASET

Resolution	Encoding Level	Representation Rates
320x240	1 - 3	46, 89, 131kbps
480x360	4 - 8	178, 222, 263, 334, 396kbps
854x480	9 - 10	533, 595kbps
1280x720	11 - 14	0.8, 1.0, 1.2, 1.5Mbps
1920x1080	15 - 20	2.1, 2.5, 3.1, 3.5, 3.8, 4.2Mbps

We used the ITEC-DASH BigBuckBunny dataset[12] [33], selecting a subset with content encoded in 2-second chunks of video (inspired by Netflix using 2-second chunks). Chunks were available at 20 different encoding RRs ranging from approximately 46kbps to 4.2Mbps as shown in Table I. We ran experiments for 400 seconds to ensure sufficient 2-second samples were collected.

[7] See Section IV-B of [31] for details on how TEACUP appropriately configures `netem` and `tc` for this purpose.

[8] Linux PIE has not changed between kernel 3.14 to the current 4.5, http://lxr.free-electrons.com/diff/net/sched/sch_pie.c?diffvar=v;diffval=3.14

[9] http://www.lighttpd.net/

[10] https://github.com/Dash-Industry-Forum/dash.js/wiki

[11] https://openconnect.netflix.com/software

[12] Full-length video sequences encoded at different bitrates, resolution and quality. Online at http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny

C. Performance indicators

Traffic was captured using `tcpdump` on all host and router interfaces. This data was then processed by Synthetic Packet Pairs (SPP) [34] to construct per-packet network-layer RTT measurements, RRs were extracted by parsing the client's HTTP GET requests, and per-chunk ARs were calculated using payload lengths extracted from HTTP response headers and the time taken to transfer packets making up each chunk.

We calculated a *Smoothed AR* (SA) – the average of the AR from the last three chunks – to mimic the signal used by `dash.js` to select the next chunk's RR. Users have been shown to be sensitive to frequent and significant representation rate switches [35], [36], [37], so we used the extracted RRs to derive the number of RR transitions and calculate the following instability index as defined in [15]:

$$\frac{\sum_{d=0}^{k-1} |b_{x,t-d} - b_{x,t-d-1}|.w(d)}{\sum_{d=1}^{k} b_{x,t-d}.w(d)}$$

Our instability index is obtained by calculating the weighted sum of the number of encoding level (Table I) transitions within the last k = 10 video chunks (where $b_{x,t}$ is the encoding level retrieved at time t), which is then normalised to a value between 0 and 1 by dividing it by the weighted sum of all encoding levels observed in the last 10 chunks (corresponding to 20 seconds of video). The weight function $w(d) = k - d$ is used to add a linear penalty to the more recent RR switches. We then slide the window in steps of one, re-calculating the instability index for each window.

Although an instability index close to zero indicates a more stable system, the instability index *cannot be used alone* to determine a DASH flow's likely QoE. The instability index must be considered in the context of corresponding median RRs, with users potentially valuing one over the other. For example, a DASH flow experiencing high instability with higher median RR might provide better experience than a DASH flow with low instability at lower median RR. A DASH flow that self-limits itself to a lower range of RRs might present as "low instability" but in fact, is a "consistently bad (low RR)" experience for the user.

To ensure their relevance to long-term viewing, we calculate SA and RR statistics after the playout buffer is pre-filled.

IV. Results and Analysis

First we explore a single DASH flow being streamed into the home from different locations, then consider a DASH flow being streamed from a local CDN while competing with two upstream or two downstream bulk TCP transfers.

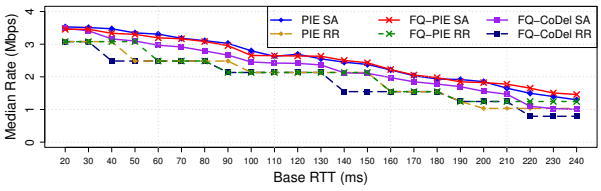
A. Impact of AQM and base RTT on a single DASH flow

Here we consider the results of a single DASH flow retrieving content from increasingly distant servers (RTTs from 20ms to 240ms), over either a 4/1Mbps or 12/1Mbps last-mile link and using FIFO (standard drop-tail queue management), PIE, CoDel, FQ-CoDel and FQ-PIE queue management schemes.

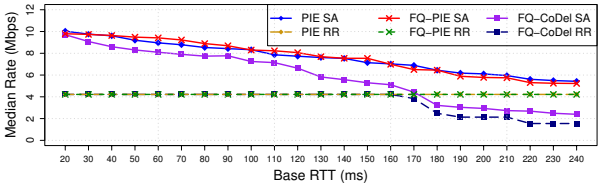
We do not present the CoDel results, as they are similar to FQ-CoDel's results in single-flow scenarios.[13] Our FIFO trials exhibited the usual, significant additional queuing delays consistent with previous observations around bufferbloat [1], so we will also not present them here.

1) DASH SA and RR: Figure 2 shows how Smoothed AR (SA) and median[14] RR of a single DASH flow varies with RTT to the content source, and how this variation is influenced by choice of AQM and available downstream last-mile capacity.

In all cases, SA drops off as RTT increases – due both to the time it takes `cwnd` to ramp up after packet loss on high RTT paths, and the delay-mitigation of all three AQMs essentially presenting TCP with a 'small buffer' bottleneck (leading to link under-utilisation). Both PIE and FQ-PIE achieve very similar SA and RR as base RTT increases. However, due to its tighter delay tolerance (and hence smaller effective queue), FQ-CoDel consistently achieves a lower SA than either PIE or FQ-PIE, which translates to lower RRs.



(a) 4/1Mbps: PIE, FQ-PIE and FQ-CoDel all drop off as RTT increases, with FQ-CoDel consistently achieving lower SA and RR figures.



(b) 12/1Mbps: PIE, FQ-PIE and FQ-CoDel achieve the highest RR up to RTT=170ms. FQ-CoDel's median SA then drops, dragging down the RRs.

Fig. 2. Median SA and RR for a single DASH flow vs base RTT

Figure 2a (the 4/1Mbps case) shows all three schemes achieving SAs below the peak RR offered by our content source, ensuring the median RRs track downwards as median SAs drop with increased RTT. Figure 2b (the 12/1Mbps case) shows all three AQMs achieving sufficient SA for the client to be regularly requesting the highest available RR, until RTT reaches 170ms. Once RTT exceeds 170ms, FQ-CoDel's SA drops below the content source's highest available RR, forcing the client to select lower RRs. {FQ-}PIE continues to achieve an SA above 4Mbps, so the client continues requesting the highest RR.

The different SAs can be attributed to different burst tolerances and queue delay targets. Being based on [2], FQ-PIE has T_{target} = 15ms, allows an initial burst of 150ms and includes drop probability autotuning and de-randomisation

[13]CoDel's authors also prefer deployment efforts focus on FQ-CoDel.

[14]The statistical spread of these measures is minimal, so we exclude this information due to space constraints.

(to minimise clustering of drops). Linux PIE (based on [32]) sets a default $T_{target} = 20$ms, allows a shorter initial burst of 100ms and lacks drop probability autotuning and de-randomisation. FQ-CoDel's burst tolerance starts at 100ms and it uses $T_{target} = 5$ms, meaning an even smaller effective queue than PIE or FQ-PIE and hence greater loss of throughput at higher base RTTs.

Note that the best SA in Figure 2a is lower than 4Mbps because our 4/1Mbps link rate limits at the IP layer while SA is based on the TCP layer 'goodput' (payload bytes successfully transferred after taking into account congestion control and retransmission of lost packets). Even at low RTTs, the AQMs trigger frequent packet losses to keep queuing delays low.

2) Induced queuing delays: Figure 3 shows the induced queuing delays due to each AQM for the trials shown in Figure 2. All three provide significant improvement over the hundreds of milliseconds of delay experienced using FIFO queue management. In general we see FQ-CoDel producing the lowest queuing delays and FQ-PIE the highest (even while FQ-PIE is generally the better option based on median SA). FQ-PIE's higher delay results are likely a consequence of [2]'s drop probability autotuning and de-randomisation – FQ-PIE drops fewer packets than Linux PIE over any given trial[15], leading to longer average queue length.

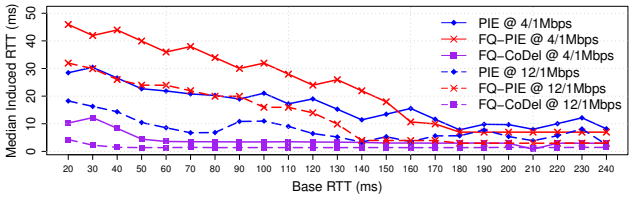


Fig. 3. Median *additional* RTT vs base RTT due to queuing delays: FQ-PIE, PIE then FQ-CoDel in descending order of induced queuing delays. Level of queuing delay also drop as SAs drop with increasing RTT.

3) RR and instability index: Figure 4 shows the instability index vs RTT (calculated using RRs collected after the local playback buffer was prefilled). Not surprisingly, PIE and FQ-PIE in the 12/1Mbps scenario score reveal low instability throughout the RTT range, due to achieving SA above the content's highest RR. All three AQMs become unstable at high RTTs in the 4/1Mbps scenario, as the achieved SAs cause the DASH client to frequently switch among various intermediate RRs. A similar problem afflicts FQ-CoDel in the 12/1Mbps scenario after RTT=170ms.

B. DASH and cross-traffic (bulk TCP transfers)

Here we evaluate a single downstream DASH flow when competing with cross-traffic in the form of two bulk downstream TCP connections or two bulk upstream TCP connections. As noted in Section III, we fixed the DASH flow's base RTT to 40ms while the simultaneous cross-traffic connections (using `iperf` and TCP CUBIC) have base RTTs ranging from

[15]Direct inspection confirmed FQ-CoDel had the highest drop rate, followed by PIE then FQ-PIE for all trials. Not included due to space constraints.

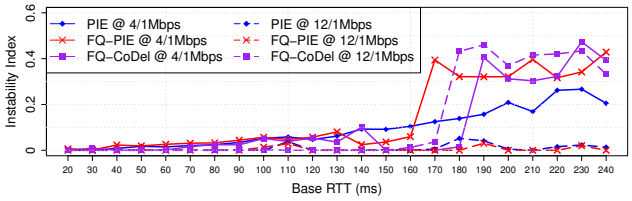


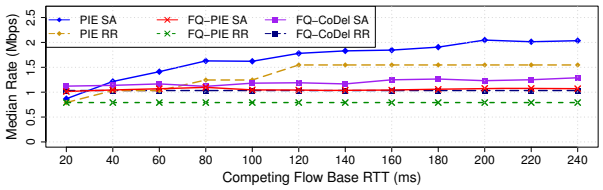
Fig. 4. Median instability index vs base RTT

20ms to 240ms to represent a diverse range of upload and download services.

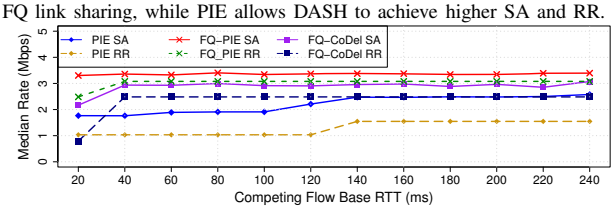
1) DASH SA and RR: Figure 5 (4/1Mbps case) and Figure 6 (12/1Mbps case) shows the impact of FQ-CoDel, PIE and FQ-PIE on DASH SA and RR as a function of the downstream or upstream cross-traffic's base RTT. In all trials, the bulk TCP flows either competed directly (when using single-queue PIE) or shared relatively equally (when using FQ-CoDel and FQ-PIE). There was some evidence of the bulk TCP flows suffering some drop-off in utilisation at higher RTTs due to all three AQMs essentially presenting TCP with a 'small buffer' bottleneck. But overall the cross-traffic did not suffer significant degradation.

Figure 5a shows that PIE achieves higher SA and RR than FQ-PIE and FQ-CoDel, peaking at ~2Mbps at 240ms RTT. This is due to the fact that FQ-PIE and FQ-CoDel's FlowQueue (FQ) scheduler enforces equal capacity sharing between all the flows presence, hence capping the maximum attainable bandwidth by DASH to a third of the bottleneck speed, ~1.3Mbps. When cross-traffic is generated upstream, as shown in Figure 5b, the FQ scheduler applies its equal capacity sharing in the upstream direction, impacting the upstream TCP ACKs from the DASH client. Due to limited upstream bandwidth (1Mbps), capacity sharing proves to be beneficial for DASH in the presence of upstream cross-traffic. TCP ACKs regulates the transmission speed and `cwnd` growth of the sender, so it is beneficial that both FQ-PIE and FQ-CoDel allow the ACKs to be forwarded in a timely manner. On the other hand, ACKs traversing a PIE queue must compete with the more aggressive bulk transfers, causing these ACKs to be delayed or dropped and hence reduce DASH's SA and RR.

Figure 6 shows the impact of FQ scheduler's capacity sharing across a 12/1Mbps bottleneck. FQ capped the maximum attainable bandwidth to ~4Mbps, whereas PIE allows the client to achieve a maximum rate of ~5.5Mbps as seen in the 240ms RTT scenario, allowing a higher RR selection. The same principle applies to the upstream case, where FQ-PIE and FQ-CoDel provides better SA than single-queue PIE. The higher burst tolerance of FQ-PIE, when coupled with FQ scheduler, enables DASH traffic to achieve a higher SA. As in the 4/1Mbps case, having an FQ scheduler allows the DASH flows ACK stream to attain a reasonable share of upstream capacity in the face of cross-traffic, and hence allows high SA. However, once the SA is regularly higher than the content's highest RR we see little impact on the selected RR. Even PIE achieves acceptable results for RTT > 60ms (when the



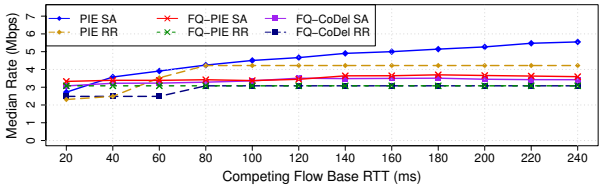
(a) DASH + 2 downstream: FQ-PIE and FQ-CoDel SA is < 1.5Mbps due to FQ link sharing, while PIE allows DASH to achieve higher SA and RR.



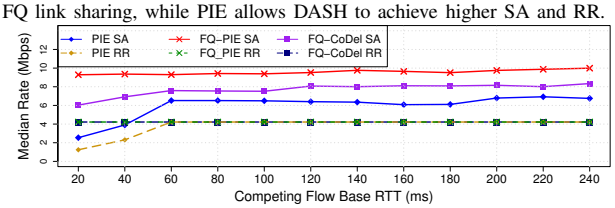
(b) DASH + 2 upstream: Cross-traffic impacts DASH ACK flow, and hence SA and RR when using PIE. FQ-PIE and FQ-CoDel achieve better results as FQ scheduler protects DASH ACK streams.

Fig. 5. Median SA and RR vs RTT in the presence of cross-traffic, 4/1Mbps

competing `iperf` bulk transfer flows begin to cede some capacity).



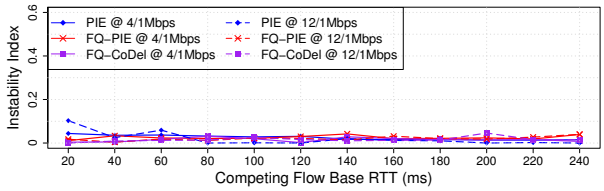
(a) DASH + 2 downstream: FQ-PIE and FQ-CoDel SA is < 4Mbps due to FQ link sharing, while PIE allows DASH to achieve higher SA and RR.



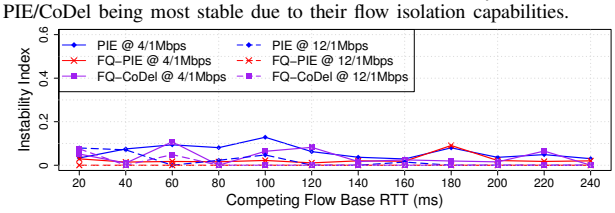
(b) DASH + 2 upstream: Although FQ scheduler protects DASH ACK streams, all three AQMs show similar median RRs as their SAs are well above the content's highest RR.

Fig. 6. Median SA and RR vs RTT in the presence of cross-traffic, 12/1Mbps

2) Instability index: Figure 7 shows the impact of AQMs on DASH instability index when faced with bulk transfer cross-traffic. In the competing downstream case, all three AQMs across 4/1Mbps and 12/1Mbps show very low instability index. As the cross-traffic's base RTT increases (and performance decreases), the DASH flow is able to consistently obtain the capacity it requires. The instability index is also relatively low for the upstream scenarios, but does increases somewhat when using PIE in the 4/1Mbps upstream scenario due to the DASH flow's ACK streams having to compete with the cross-traffic. FQ-PIE and FQ-CoDel shows low instability index due to their flow isolation capabilities, protecting the ACK stream and ensuring the DASH flows achieve more consistent (and reasonably high) SA and RR.



(a) DASH + 2 downstream: All three AQMs are relatively stable, with FQ PIE/CoDel being most stable due to their flow isolation capabilities.



(b) DASH + 2 upstream: High instability index for DASH across plain PIE due to its ACKs competition against bulk transfer traffic.

Fig. 7. Median instability index vs RTT in the presence of cross-traffic

V. Conclusion and Future Work

The likely deployment of modern AQM schemes in home gateways and ISP borders will complicate the relationship between DASH-based stream services and other user-generated cross-traffic. Using a controlled experimental testbed, we have evaluated the potential impact of PIE, FQ-CoDel and a new hybrid FQ-PIE AQMs on DASH-based content delivery in the presence of cross-traffic. Part of the challenge is the ON-OFF nature of DASH traffic, and the other is the effective small bottleneck buffer exhibited by modern AQM schemes. We have shown that PIE's greater burst tolerance and higher delay targets will lead to better performance for single DASH streams than FQ-CoDel. Moreover, we also show that FQ-CoDel's inter-flow capacity sharing leads to more predictable behaviour in the face of cross-traffic (and avoidance of the "downward spiral" effect when competing with cross-traffic). The most useful insight is that we see best overall results when the bottleneck link implements a hybrid of FQ-CoDel's FlowQueue scheduling algorithm and PIE's individual queue management. We demonstrate this using a recently released FreeBSD implementation of "FQ-PIE".

Future work includes investigation of how buffer occupancy or control-theory ABR schemes interact with FQ-CoDel and FQ-PIE, and whether any benefits may accrue from running DASH over delay-based TCP or hybrid delay/loss-based TCPs.

References

[1] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue*, vol. 9, no. 11, pp. 40:40–40:54, Nov. 2011. [Online]. Available: http://doi.acm.org/10.1145/2063166.2071893

[2] R. Pan, P. Natarajan, F. Baker, G. White, B. VerSteeg, M. Prabhu, C. Piglione, and V. Subramanian, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem," IETF Draft, draft-ietf-aqm-pie-06, April 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-pie-06

[3] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," IETF Draft, draft-ietf-aqm-codel-03, March 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-codel-03

[4] T. Høeiland-Jørgensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, "FlowQueue-Codel," IETF Draft, draft-ietf-aqm-fq-codel-06, March 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-06

[5] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567 (Best Current Practice), IETF, Jul. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7567.txt

[6] G. White and R. Pan, "A PIE-Based AQM for DOCSIS Cable Modems," IETF Draft, https://tools.ietf.org/html/draft-ietf-aqm-docsis-pie-02, February 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-aqm-docsis-pie-02

[7] Sandvine, "Sandvine Global Internet Phenomena Report," https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-latin-america-and-north-america.pdf, 2015 (accessed May 2016).

[8] ISO/IEC, "ISO/IEC 2309-1:2012 Information Technology: Dynamic Adaptive Streaming over HTTP (DASH) Part 1: Media presentation description and segment formats," http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57623, 2012.

[9] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling Netflix: Understanding and improving multi-CDN movie delivery," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 1620–1628.

[10] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943572

[11] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the Impact of Video Quality on User Engagement," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 362–373. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018478

[12] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *Communications Surveys Tutorials, IEEE*, vol. 17, no. 1, pp. 469–492, Firstquarter 2015.

[13] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *CoRR*, vol. abs/1305.0510, 2013. [Online]. Available: http://arxiv.org/abs/1305.0510

[14] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 169–174. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943575

[15] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014. [Online]. Available: http://dx.doi.org/10.1109/TNET.2013.2291681

[16] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 187–198. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626296

[17] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: near-optimal bitrate adaptation for online videos," *CoRR*, vol. abs/1601.06748, 2016. [Online]. Available: http://arxiv.org/abs/1601.06748

[18] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 325–338. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787486

[19] X. Yin, V. Sekar, and B. Sinopoli, "Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIII. New York, NY, USA: ACM, 2014, pp. 9:1–9:7. [Online]. Available: http://doi.acm.org/10.1145/2670518.2673877

[20] C. Zhou, X. Zhang, L. Huo, and Z. Guo, "A control-theoretic approach to rate adaptation for dynamic HTTP streaming," in *Visual Communications and Image Processing (VCIP), 2012 IEEE*, Nov 2012, pp. 1–6.

[21] C. Zhou, C.-W. Lin, X. Zhang, and Z. Guo, "Buffer-based smooth rate adaptation for dynamic HTTP streaming," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, Oct 2013, pp. 1–9.

[22] 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 4th ACM Multimedia Systems Conference*, ser. MMSys '13. New York, NY, USA: ACM, 2013, pp. 214–225. [Online]. Available: http://doi.acm.org/10.1145/2483977.2484004

[23] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681, IETF, Sep. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5681.txt

[24] G. Fairhurst, A. Sathiaseelan, and R. Secchi, "Updating TCP to Support Rate-Limited Traffic," RFC 7661 (Experimental), IETF, Oct. 2015. [Online]. Available: https://tools.ietf.org/rfc/rfc7661.txt

[25] S. Nazir, Z. Hossain, R. Secchi, M. Broadbent, A. Petlund, and G. Fairhurst, "Performance Evaluation of Congestion Window Validation for DASH Transport," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, ser. NOSSDAV '14. New York, NY, USA: ACM, 2014, pp. 67:67–67:72. [Online]. Available: http://doi.acm.org/10.1145/2578260.2578275

[26] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/2398776.2398800

[27] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309 (Informational), IETF, Apr. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2309.txt

[28] R. Al-Saadi and G. Armitage, "Dummynet AQM v0.2 – CoDel, FQ-CoDel, PIE and FQ-PIE for FreeBSD's ipfw/dummynet framework," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 160418A, 18 April 2016. [Online]. Available: http://caia.swin.edu.au/reports/160418A/CAIA-TR-160418A.pdf

[29] T. Høiland-Jørgensen, P. Hurtig, and A. Brunstrom, "The good, the bad and the wifi: Modern {AQMs} in a residential setting," *Computer Networks*, vol. 89, pp. 90 – 106, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615002479

[30] G. Hong, J. Martin, and J. M. Westall, "On fairness and application performance of active queue management in broadband cable networks," *Computer Networks*, vol. 91, pp. 390 – 406, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615002789

[31] S. Zander and G. Armitage, "TEACUP v1.0 - A System for Automated TCP Testbed Experiments," http://caia.swin.edu.au/reports/150529A/CAIA-TR-150529A.pdf, Melbourne, Australia, 29 May 2015.

[32] R. Pan, P. Natarajan, C. Piglione, M. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem," IETF Draft, draft-pan-aqm-pie-00, June 2013. [Online]. Available: https://tools.ietf.org/html/draft-pan-aqm-pie-00

[33] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12. New York, NY, USA: ACM, 2012, pp. 89–94. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155570

[34] S. Zander and G. Armitage, "Minimally-Intrusive Frequent Round Trip Time Measurements Using Synthetic Packet Pairs," in *The 38th IEEE Conference on Local Computer Networks (LCN 2013)*. IEEE, October 2013.

[35] R. K. Mok, E. W. Chan, X. Luo, and R. K. Chang, "Inferring the QoE of HTTP Video Streaming from User-viewing Activities," in *Proceedings of the First ACM SIGCOMM Workshop on Measurements Up the Stack*, ser. W-MUST '11. New York, NY, USA: ACM, 2011, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/2018602.2018611

[36] N. Cranley, P. Perry, and L. Murphy, "User Perception of Adapting Video Quality," *Int. J. Hum.-Comput. Stud.*, vol. 64, no. 8, pp. 637–647, Aug. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.ijhcs.2005.12.002

[37] C. Mueller, S. Lederer, R. Grandl, and C. Timmerer, "Oscillation compensating Dynamic Adaptive Streaming over HTTP," in *2015 IEEE International Conference on Multimedia and Expo (ICME)*, June 2015, pp. 1–6.