HTTP Adaptive Streaming and Access Router Management: the Users' and Network's Perspectives

Sergio Livi, Lucile Sassatelli, Guillaume Urvoy-Keller
Université Côte d'Azur, CNRS, I3S - Sophia Antipolis, France
Emails: {first.last}@unice.fr

***Abstract*—The share of video streaming in the Internet traffic is expected to reach 80% by 2019. To deliver these video services in possibly strained settings and to meet with the users requirement of anywhere and multi-screen experience, HTTP Adaptive Streaming (HAS) is gaining momentum. However, TCP-transport of HAS flows is known to entail a number of issues both on Quality of Experience (QoE) and network metrics. Thanks to testbed experiments, we examine the video streaming performance in a home access scenario by exposing the interplay between the video rate adaptation logic (Buffer-Based -BBA, or Rate-Based -RBA) and the buffer sizes and policies (Droptail, Active Queue Managements - AQMs). A main finding is that AQMs move the bulk of TCP losses from the initial burst and ACK-trailing phases, problematic to HAS flows, to the more favorable ACK-clocking phase. We thereby show that implementing AQM CoDel at the access router is a simpler means, working for any family of client's adaptation, to improve QoE and network metrics than striving to smooth out the HAS bursts by pacing at the client or middleboxes, as previous approaches do.**

I. Introduction

According to [1], "Globally, IP video will represent 80 percent of all traffic by 2019, up from 67 percent in 2014". To deliver Over The Top (OTT) video services in possibly strained settings (variable and limited bandwidth due to difficult wireless channels or congested access networks) and to meet with the user's requirement of anywhere multi-device multi-screen experience, OTT and IPTV systems are increasingly adopting HTTP Adaptive Streaming (HAS) (e.g., BT's platform YouView [2]), now standardized as MPEG-DASH (ISO/IEC 23009-1:2014). Videos are chopped into segments of fixed duration. The goal is to choose the requested video rate for each segment so as to ensure high user's Quality of Experience (QoE), meant to represent users' perception of the video playback (metrics are given in Sec. III). Different video rate decision algorithms exist and consider different decision criteria. We can isolate two main families: rate-based (RBA) and buffer-based (BBA) algorithms, both of which try to maximize the video bit rate - by fitting it to the estimated network bandwidth for RBA, and to the buffer occupancy for BBAs (to avoid stalls as much as possible) [3].

Furthermore, the main advantages that drove OTT video to be carried through HTTP is that the transport protocol is TCP, thereby assuring middlebox traversal (HTTP on TCP port 80 is seldom filtered out) and Internet stability by the very aim of TCP. Owing to the peculiarities of HAS flows, their TCP transport entails however a number of issues highlighted in [4], [5], which have both showed that even a single HAS flow can cause a router's buffer overflow. The loss pattern being critical to HAS performance [4], the queue management policy at the bottleneck router is pivotal to QoE. In order to combat the full buffer problem, known as bufferbloat, Active Queue Management (AQM) policies have been designed to limit the inflation of queuing delays. However, the deployment of Random Early Detection (RED) and its adaptive version ARED [6] has been hindered by the need for parameter tuning. To remedy these problems, CoDel, has been designed to spare from any knob tweaking [7] and addresses the excessive available buffer size (e.g., todays access routers can hold tens to hundreds packets [5]).

The goal of this article is therefore to investigate, through testbed experiments, the impact on QoE and network metrics of the interplay between the type of HAS algorithm (RBA and BBA) and the buffer size and management policy. Our focus is specifically of a access router handling a restricted number of HAS flows. This scenario where the access router becomes the bottleneck of HAS flows is even more realistic considering the increase of video viewings per household and the content being stored close to the user in CDN or in-ISP caches. Our contributions are threefold:

• For a single client, the QoE metrics are not significantly impacted by the buffer sizes and management. The network metrics are however, and we unveil how AQMs help the HAS flows to obtain a better link utilization. By analyzing the loss location patterns, we identify that ARED and CoDel can achieve the same pacing result as the approaches proposed in the literature [5], [8] to mitigate the bad effect of the interplay between TCP and HAS on QoE and link utilization.

• We show and analyze why large buffers with Droptail can increase the re-buffering ratio of RBA and BBA, with static and variable channels. In particular, we identify two major issues they entail: to the TCP congestion window, and to the HTTP request structure.

• With Droptail and moderate to large buffers, the quality fairness degrades, specifically for BBAs which perform up to 150% worse than RBA in the 2-client constant and variable bandwidth cases, which are typical household streaming conditions. We explain why it is so, and show that the unfairness decreases dramatically with AQM.

II. Related works

Both Droptail (DT) and RED policies have shown incapable of preventing HAS flows from saturating large buffers [5], [4]. To the best of our knowledge, neither ARED nor CoDel have been evaluated for HAS performance. Only AQM policies involving mainly service differenciation have been considered so far for streaming flows, e.g., [9] and WiFi Multimedia (WMM, based on IEEE 802.11e). The impact of losses on TCP-carried HAS flows has been dissected in [4]. These flows have indeed very specific features: (i) the ON/OFF behavior generates regular bursty short-lived flows, corresponding to successive segment transfers, similar to small web objects, but (ii) the TCP connection is usually in persistent mode (no reset of the TCP congestion window at each segment download). We build on the analysis of [4] to investigate how the buffer size and management impacts the clients' QoE, depending on the HAS algorithm and client contention.

Some solutions (such as [5], [10], [11]) have been designed to alleviate these problems entailed by the peculiarities of HAS flows. For example, Sabre [5] modifies the client application so that it controls the rate at which the socket buffer is drained, in turn controlling the TCP advertised receive window. It is worth noting that, doing so, the downloading rate experienced by the client gets modified and the RBA must be turned into a simple two-level buffer-based. This reveals how the rate-decision algorithm, TCP and the queue management policy are entwined in the resulting QoE. Our goal is not to design a better HAS algorithm or AQM, but instead to consider representative classes of each and uncover the exact mechanisms involved in their interplay. We consider these classes being RBA of [12, Sec. 4.2], BBA of [3], DT, ARED and CoDel.

Complementarily to [3], [13], our work serves also as a detailed evaluation on a controlled testbed of the performances of BBA [3] in terms of QoE and fairness, on static and variable channels, for different buffer sizes and managements.

III. Experimental setup

Testbed: We have made our testbed publicly available at [14] for results reproduction. It is made of: (i) a Virtual Machine (VM) for each client with VLC in which the HLS implementation is used and the HAS algorithms of [12, Sec. 4.2] and [3] are coded; (ii) a VM for the HTTP server which is a simple node.js instance serving static files; (iii) two VMs emulate the network using the *tc-netem* Linux tool from the *traffic control* suite to shape the traffic according to the desired delay, bandwidth and buffer size. The servers use TCP Cubic. The *libcurl* library is used to allow the reuse of the TCP connection over the successive segment downloads. We make the bottleneck link bandwidth vary from 600 Kbps to 3Mbps, multiplied by the number of clients. Round Trip Delay (RTD) denotes the round trip physical latency. Round Trip Time (RTT) denote the total time between the sending of a packet and the reception of the TCP acknowledgment (ACK). RTT is the sum of RTD and queuing delay. RTDs and Buffer Sizes (BS, in packets) are configured considering a broadband copper access (DSL). We consider a total downlink bandwidth and RTD of 10 Mbps and 80 ms. The corresponding total Bandwidth Delay Product (BDP) is hence $BDP = 67$ packets. The home access router's buffer sizes are set to $\{6, 1, 0.5, 0.25\}BDP$. Note that the lower bandwidth per client mentioned above are adapted to the test video (and can account for cross-traffic). They would be scaled up and the results would apply with, e.g, the new identified trend of 4K videos, requiring at least 15 Mbps bandwidth. Let us specify that we consider not artificially too high RTDs (similar to, e.g., [4, Sec. 6] for DSL), which are typical of accessing a content in a close-by (CDN) cache, as it is prominently the case today. The variable bandwidth profile is: 4, 2.8, 1.5, 1, 0.6 and 4 Mbps, each for an equal share of the video duration. Also, the number of simultaneous video clients we consider is 1 to 3, based on 2.5 person per household [15].

Videos: The video clip we use unless otherwise specified, is the 10 minute-long movie *Big Buck Bunny* (https://peach.blender.org/, same as in [16], [4]): eight bit rates available, from 350 Kbps to 2750 Kbps, 300 2-second segments with different sizes (corresponding to variable bit rates). As a baseline, we also use a constant bit rate 30-minute clip *Apple's BipBop* (https://developer.apple.com/streaming /examples/basic-stream.html): four bit rates available, from 232 kbit/s to 2 Mbit/s, 10 seconds fixed-size segments.

Algorithms: We consider the two most advanced flavors of BBAs presented in [3]. *BBA2* bases its rate decision on the playout buffer state and the size of upcoming segments (to account for variable bit rate), and ramps up the requested bit rate in the startup phase (when the buffer level is low) based on bandwidth estimation. *BBA3* reins in the bit rate switches to improve QoE, thanks to a lookahead parameter. We consider two RBA versions. *Classic1* is described in [12, Sec. 4.2] and [12, Sec. 2]. The bit rate matches an exponential weighted moving average of the downloading rate of each segment, to a 0.8 margin. *Classic2* proceeds the same way except the downloading rate of a segment is computed by dividing the segment size by the duration between the reception of the first and last bit (while *Classic1* considers the duration from the sending of HTTP request to last bit delivery).

Metrics: Despite the intrinsic subjectivity of such experience, the multimedia community has agreed on three most important QoE metrics for video, as defined in [17]: rebuffering ratio (representing the impact of playback stalls), video quality (a log function of the video bit rate) and quality instability. We consider the following metrics, the first three being usual QoE metrics, the last three being in-network metrics. They are referred by their acronyms in the next section.

- Re-buffering ratio (RBR): fraction of time spent in stall, waiting for new data to arrive and resume playing (includes the startup delay here).
- Average relative bit rate (ARB): bit rate averaged over all segments normalized by the per-client bandwidth.
- Instability (INS): number of bit rate switches normalized by the total number of segments [12].
- Quality unfairness (QU): absolute difference between the averaged quality level of each client, where the bit rates

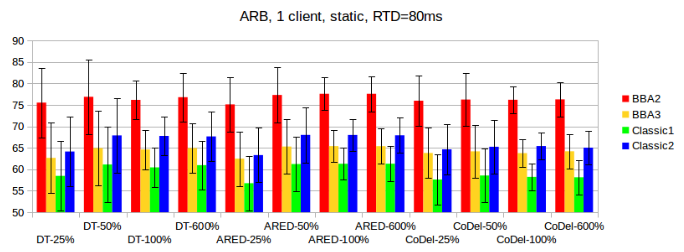

Fig. 1: Average relative bit rate, 1 client, static bandwidth.

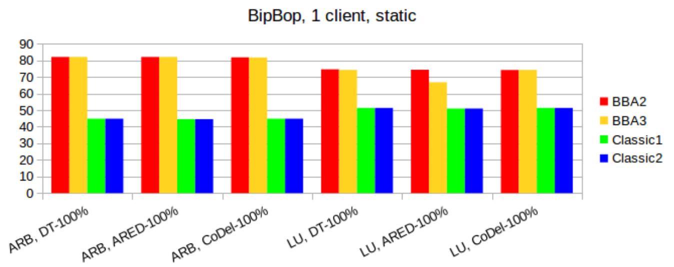

Fig. 2: Average relative bit rate, 1 client, static bandwidth, BipBop video.

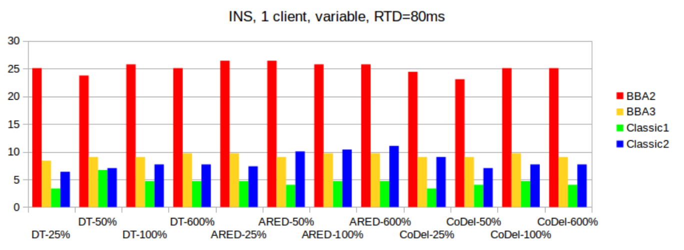

Fig. 4: Instability, 1 client, variable bandwidth.

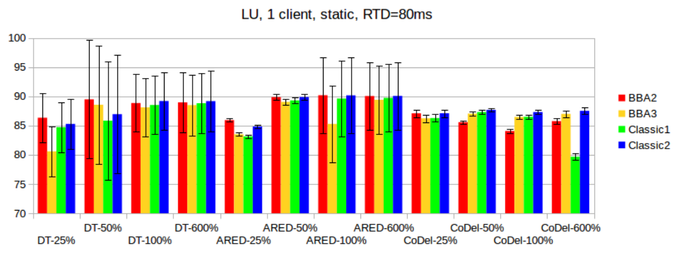

Fig. 5: Link utilization, 1 client, static bandwidth.

are numbered from 0 then mapped to a percentage scale to yield the quality levels. QU hence assesses the difference in perceived visual quality, independently from INS.

• Link utilization (LU): time average (polling intervals are 10 ms) of the actual rate on the bottleneck link normalized by the per-client bandwidth multiplied by the number of clients.

• RTT inflation (RTTi): time average of RTT seen by a TCP connection divided by RTD (in percent).

IV. Numerical results

Each experiment was run twice if one client was involved, four times otherwise. The 95% confidence intervals are shown. Apart from Fig. 2, all figures are obtained for Big Buck Bunny video.

A. General QoE features of the HAS algorithms

We first discuss the ARB and INS QoE metrics in the case of a single client, to highlight the general properties of the HAS algorithms considered. Both in the constant and variable bandwidth cases, there is no re-buffering events (RBR$=0$). Fig. 1 shows that ARB is higher with BBAs than with Classic1, but the difference with Classic2 is small, especially for BBA3. In contrast, it can be verified on Fig. 2 that BBAs consistently reach a higher ARB than Classics for a constant bit rate movie. This reflects the basic idea of BBAs: it acts as an integrator of the excess rate by accumulating on the periods where the requested bit rate is lower than the downloading rate, to allow to move up the requested rate from time to time, thereby increasing the time average ARB, something Classics are incapable of. This behavior is fully enabled with a constant bandwidth and constant bit rate video. However with a variable bit rate-encoded video, the periods where this accumulation can occur are in some extent balanced out with periods where the instantaneous rate is higher than the nominal rate of the segment, thereby reducing the margin between BBAs and Classics. The instantaneous segment rates for each nominal rate are represented in Fig. 3. As a direct consequence of the above observation, Fig. 4 (for variable bandwidth, qualitatively similar as static bandwidth) shows instability is much higher with BBA2 than with Classics. As BBA3 strives to limit the rate switches, its INS gets closer to that of Classics, but consequently its aggressiveness in ARB is lower. More in detail, we observe that in the constant bandwidth (single client case), Classic2 performs as well as BBA3 in terms of ARB, and better in INS.

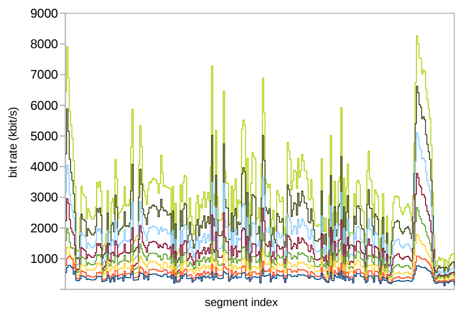

Fig. 3: Instantaneous rate of each nominal-rate representation for Big Buck Bunny video.

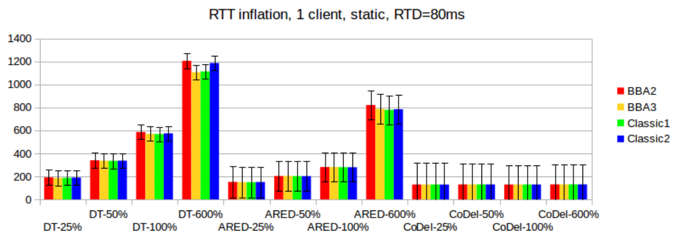

Fig. 6: RTT inflation (in %), 1 client, static bandwidth.

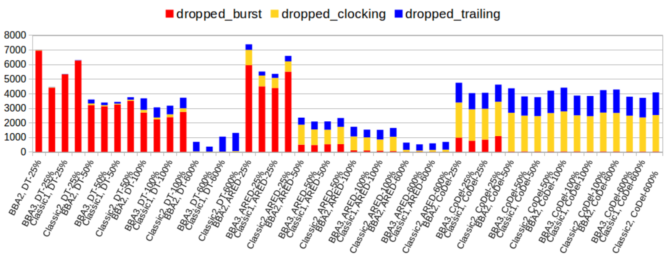

Fig. 7: Distribution of loss phases, 1 client, static bandwidth.

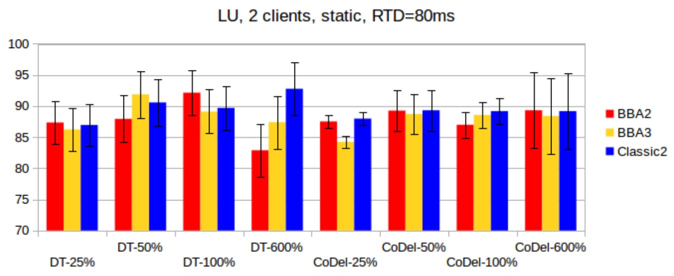

Fig. 8: Link utilization, 2 clients, static bandwidth.

B. *Single client: QoE and network metrics*

Fig. 5 and 6 show the network metrics link utilization (LU) and RTT inflation (RTTi), respectively. First it can be observed that the LU (by a single flow here) remains between 85% and 90% for all buffer sizes, queue managements and almost all HAS algorithms. This is to be related to the results presented in [4, Fig. 1], where utilization is shown to decrease with RTD and drop below 80% for RTD$>$ 140ms. Esteban et al. [4] dissect the download of a segment by breaking it down into three phases: initial burst, ACK-clocking and ACK-trailing. If the segment is small enough so that the transfer can be completed in one or only few RTTs, then the losses in the startup phase may prevent *cwnd* (TCP congestion window) from reaching the optimal value, entailing a low utilization of the bottleneck link. The losses in the ACK-trailing phase are very detrimental as the fast-recovery mechanism cannot be triggered by lack of packets to transmit, the last *cwnd* being consequently spuriously shrinked for the next segment. The extra RTTs needed for the transfer to complete are also detrimental to the accuracy of the rate estimation (see Sec. III, Classic1). Hence, despite the sizable increase in RTT depicted in Fig. 6 (with DT and ARED, due to big buffers), utilization remains high in our case of RTD$=$ 80ms. Correlating with [4], we conclude it is the increase in RTD and not in RTT which is responsible for a low link utilization by HAS flows, despite the Cubic-transport (meant to handle high RTTs). Indeed, losses may occur in the burst phase if *cwnd*$>$BS, a necessary condition to maintain high utilization is hence BS$>$BDP$=$ $bwRTD$, which is the case in most samples considered to average LU (a sample correspond to a per-client bandwidth): we get lower LU for BS$=$ 25%BDP (total BDP, see Sec. III) for which case BS$=$ 17 packets, while $RTD.bw = 20$ packets for maximum per-client bandwidth of bw $=$ 3Mbps. The distribution of losses between the three phases is shown in Fig. 7. The higher number of losses in the ACK-trailing phase with CoDel explains the utilization slightly lower than with DT and ARED, though this is mitigated by the low maintained RTT of CoDel, compared with DT and ARED. In particular, it is important to observe that CoDel is able to move the losses from the detrimental burst and ACK-trailing phases to the ACK-clocking phase, where they are best corrected (with fast-retransmit, and fast-recovery is possible) [4]. This important phenomenon has further implications on QoE, as we shall see for 2 and 3 clients. CoDel is indeed designed to let pass the initial flow bursts (considered as “good queue” in [7]), while cutting back belatedly on queue and hence on *cwnd* involved in next bursts. Let us finally mention that CoDel yields more losses than DT and ARED (which unsurprisingly has similar ability as CoDel to move loss onto the ACK-clocking phase) in order to better rein in the RTT inflation. It is also very interesting to observe that, even with no multiplexing (one HAS alone is considered here), AQM and CoDel in particular achieve the pacing mechanisms which used to require client [5] or middlebox [11] modifications.

Furthermore, it is worth noting that, for this 1-client case, the observed differences in network metrics do not necessarily reflect on QoE. For instance, utilization of BBA2 and BBA3 is pretty much the same for CoDel 25%, while the obtained ARBs are significantly different. As shown in the literature, this highlights that the relation between network metrics and QoE is not straightforward. Here, we observe that ARB is more related to the losses than to utilization: the ARB patterns distinguishing the different HAS algorithms (Fig. 1) are also found in the total loss levels (Fig. 7). Indeed, unlike Classics, BBAs do not base the requested rate on the experienced downloading rate: requesting rates higher than the bandwidth yields more losses but maintains utilization to a level similar to that of Classics.

C. *Two clients: re-bufferings and large buffers*

We now analyze the competition between 2 clients, first for static per-client bandwidth, as showed in Fig. 8, 9 and 11. Only DT and CoDel are considered for two clients. We notice that the differences between the different algorithms are more pronounced than with a single client. The case DT 600% is interesting as it corresponds to the case of over-dimensioned buffer, and yield specific problems. In this case, compared to BBA3, BBA2 has a lower utilization because, as it can be seen in Fig. 10, it has a higher number and fraction of losses in the ACK-trailing phase. BBA2 however gets a higher ARB because it keeps being more aggressive than BBA3. In this same case, Classic2 exhibits both a higher utilization and ARB, despite an even larger number and fraction of losses in the ACK-trailing phase. This comes however at the cost of a high re-buffering ratio, shown in Fig. 11 (note that startup buffering is included, and the log scale).

Remark: The buffer size from which Classic2 is likely to produce re-bufferings can be estimated as follows. A segment download is not possible if the duration from the sending of the HTTP Get to the receiving of the last data packet is higher than

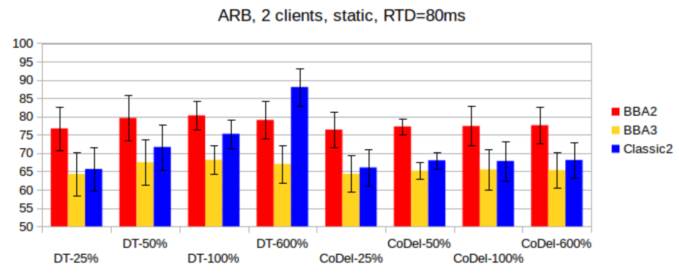

Fig. 9: Average relative bit rate, 2 clients, static bandwidth.

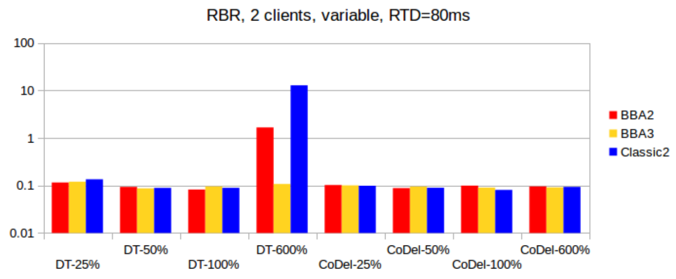

Fig. 11: Re-buffering ratio, 2 clients, variable bandwidth.

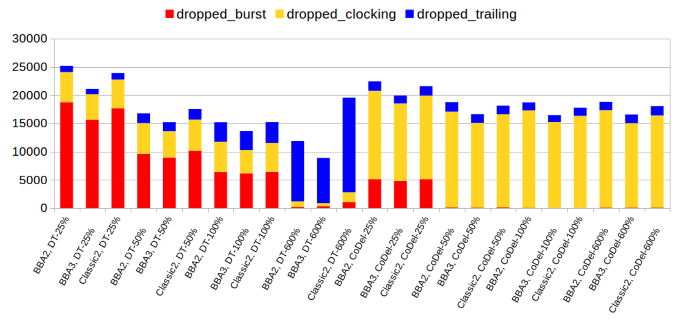

Fig. 10: Distribution of loss phases, 2 clients, static bandwidth.

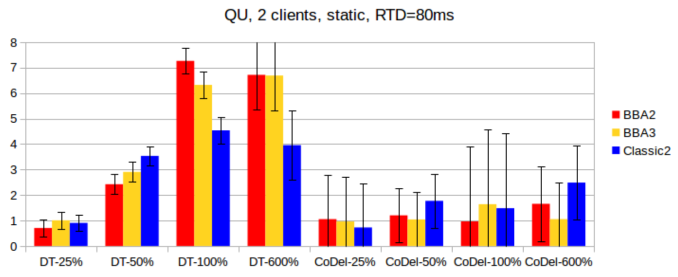

Fig. 12: Quality unfairness, 2 clients, static bandwidth.

the segment duration (S seconds), i.e., $segment\_size/bw > S - RTT$. Considering the maximum $RTT = bwRTD + BS$, $BS = (1+\alpha)bwRTD$ and that the maximum requested rate by Classic2 is $0.8bw$ (see Sec. III), we get that a re-buffering occurs when $\alpha > 0.2S/RTD - 1$. For $bw = 2$Mbps, $S = 2$s and $RTD = 80$ms, we get $\alpha > 4$. Let us mention that no re-buffering occurred in the single-client case because the buffer was likely big enough to support the overload incurred by Classic2. However in the case of two clients, the per-client bandwidth keeps the same as in the single case (see Sec. III) but the buffer size does not increase. This produces more losses, as verified by Y-axis scale in Fig. 10 compared with Fig. 7. $\diamond$

The above remark underlines how serious an issue big buffers handled with DT can be for HAS algorithms. Indeed, for rate-based decision algorithms such as Classic2, this can readily incur re-bufferings owing to the underestimation of the segment download duration. For Classic1, big buffers lead conversely to an underestimation of the available bandwidth, hence not making the most of the available resources owing to protocol design (persistent need of 1 RTT for the HTTP request). Buffer-based algorithms only partially dodge the problem as they still work with HTTP request at each segment. This can be observed in Fig. 11 where, for big buffer and DT, re-bufferings appear for BBA2 too.

D. *Several clients: unfairness*

Fig. 12 and 13 show that quality unfairness (QU) is high for high buffer size for DT. First, for 2 clients and static bandwidth, BBAs have a significantly worse QU than Classic2 for large buffers. Still for DT, increasing the number of clients to 3 worsens QU and blurs the difference between the algorithms. With variable bandwidth and 2 clients, unfairness gets much worse with BBA2 from moderate buffer size (50%BDP), aas well as with BBA3. This setting is however very important as it corresponds to a typical home setting where 2 devices are simultaneously streaming. Why BBAs are more unfair than Classic2 for large buffer managed with DT (and low multiplexing) can be explained as follows. As demonstrated in [8], the players requesting higher bit rates observe a higher bandwidth. Unlike the RBAs, the BBAs react to a bandwidth change only through the buffer state, therefore later than a RBA. The losses in the ACK-trailing phase yield an unfair bandwidth share. Hence, if a client experiences a loss in the ACK-trailing phase, the other gets a higher bandwidth share for several RTT, therefore filling up its playout buffer and keeping requesting high bit rates. This is illustrated in Fig. 15. We note in passing that BBA3 has not necessarily a better fairness than BBA2 (e.g., static bandwidth), which means that a lower instability does not imply a better fairness.

Second, these figures consistently show a significantly better fairness with CoDel, for all HAS algorithms and buffer sizes. The reason for having CoDel allowing for a better fairness than DT lies in the fundamental difference of their respective loss patterns (see Fig. 10): there are almost no more initial burst losses and the vast majority of the losses occur in the ACK-clocking phase. This phase allows to perform a fast-recovery along with the fast-retransmit upon loss, and hence the TCP rate is maintained and can ramp up again, thereby ensuring that the sender does not lose ground against a contender. The opposite happens when losses occur in the ACK-trailing phase, and this explains why QU worsens when buffer size increases with DT. The loss patterns therefore impact the quality unfairness through the bandwidth share they yield: a fairer bandwidth share is favored by losses in the ACK-clocking phase, and hence by AQM.

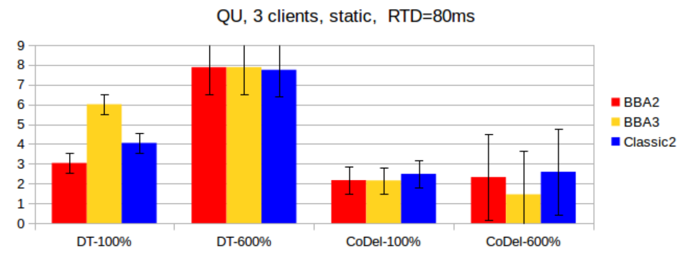

Fig. 13: Quality unfairness, 3 clients, static bandwidth.

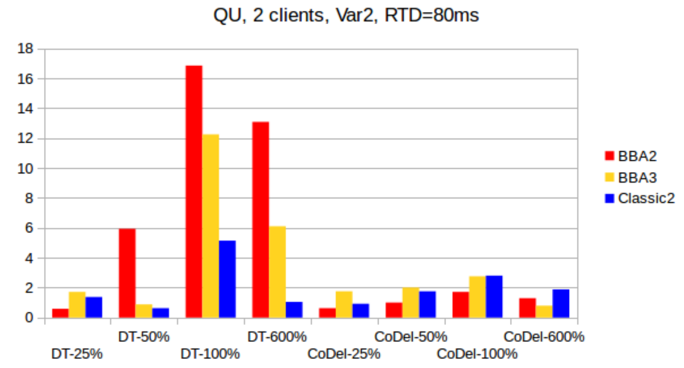

Fig. 14: Quality unfairness, 2 clients, variable bandwidth (profile 2).

V. Conclusion

Thanks to testbed experiments, we have examined the multimedia streaming performance in a home access scenario. We have exposed the interplay between the video rate adaptation logic and the buffer sizes and policies (DT, ARED, CoDel), and the impact on the QoE metrics (video rate, re-bufferings, instability, unfairness) and network metrics (utilization, RTT inflation and loss patterns).

First, for a single client, the QoE metrics are not significantly impacted by the buffer sizes and management. The network metrics are however, and we unveil how AQMs help the HAS flows to obtain a better link utilization. A main finding is that AQMs (and CoDel in particular), move the bulk of TCP losses from the problematic initial burst and ACK-trailing phases to the more favorable ACK-clocking phase.

Second, we have identified why large delays entailed by large DT-buffers are harmful: (i) the extra RTTs spent without data transfer but required to carry out the HTTP exchange

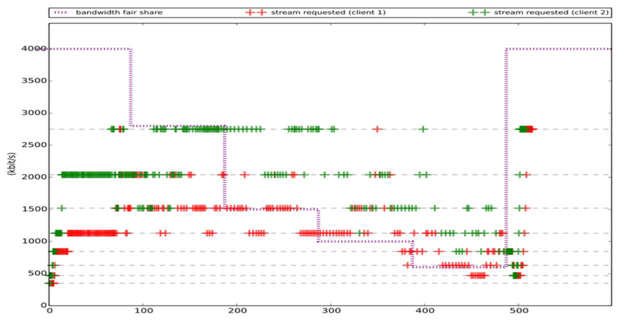

Fig. 15: Time series of both clients' bit rates, BBA2, DT-100%. Pink dashed line is the per-client bandwith, green and red crosses are the rates requested by each client.

(not yet solved for HAS but partly addressed in HTTP/2) may lead to a wrong rate estimation (specifically detrimental for Classic), and (ii) *cwnd* gets higher hence more harmful bursts and larger delay, hence more harmful trailing phases.

Third, we have showed that, with DT and moderate to large buffers, the quality fairness degrades, specifically for BBAs which perform up to 150% worse than Classic in the 2-client constant and variable bandwidth cases, which are typical household streaming conditions. Again, CoDel is able to alleviate the problem for the different HAS algorithms.

A take-away message is therefore that implementing AQM (CoDel does not require tuning, unlike RED) at the access router is a simpler means to improve QoE and network metrics than striving to smooth out the HAS bursts by pacing at the client [5] or middleboxes [11]. Finally, we point out that these results also fit within the current trend of UDP-transport of video flows owing to their need of congestion control. In particular, QUIC [18], being deployed globally by Google, encompasses HTTP/2 and Forward Error Correction mechanisms along with Cubic congestion control.

References

[1] Cisco, "VNI Global IP Traffic Forecast, 2014 - 2019."
[2] Over the Top GStreamer, YouView TV. GStreamer Conference 2014, "http://gstreamer.freedesktop.org/conference/2014/speakers.html."
[3] 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 *ACM SIGCOMM*, Aug. 2014.
[4] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac, "Interactions between HTTP adaptive streaming and TCP," in *ACM NOSSDAV*, Jun. 2012.
[5] A. Mansy, B. Ver Steeg, and M. Ammar, "SABRE: A client based technique for mitigating the buffer bloat effect of adaptive video flows," in *ACM MMSyS*, Feb. 2013.
[6] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," Tech. Rep., 2001.
[7] K. Nichols and V. Jacobson, "Controlling Queue Delay," *ACM Queue*, vol. 10, no. 5, May 2012.
[8] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with FESTIVE," in *ACM CoNEXT*, Dec. 2012.
[9] H. Luo, "Improve delay performance of wireless video streaming with active queue management," in *IEEE Int. Conf. on Software Eng. and Service Sc. (ICSESS)*, Jun 2012, pp. 777–780.
[10] X. Liu, A. Men, and P. Zhang, "Enhancing TCP to Improve Throughput of HTTP Adaptive Streaming," *Int. J. of Future Generation Comm. and Netw.*, vol. 7, no. 1, 2014.
[11] C. Ben Ameur, E. Mory, and B. Cousin, "Evaluation of gateway-based shaping methods for http adaptive streaming," in *IEEE ICCW*, Jun. 2015.
[12] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when HTTP adaptive streaming players compete for bandwidth?" in *ACM NOSSDAV*, Jun. 2012.
[13] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *ACM SIGCOMM*, Aug. 2015.
[14] Article's Testbed, "https://github.com/serl/hls-bba-testbed."
[15] US Census Bureau, "https://www.census.gov/hhes/families/files/graphics/HH-6.pdf."
[16] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *ACM MMSyS*, Feb. 2011.
[17] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *ACM SIGCOMM*, 2015, pp. 325–338.
[18] J. Iyengar and I. Swett, "QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2," in *IETF draft*, Jun 2015.