Case for caching and Model Predictive Control quality decision algorithm for HTTP Adaptive Streaming: is cache-awareness actually needed?

Vitalii Poliakov, Lucile Sassatelli
Université Côte d'Azur, CNRS, I3S, France

Damien Saucez
Université Côte d'Azur, Inria, France

***Abstract*—Presence of in-network opportunistic cache is known to cause undesirable effects on Quality of Experience when using HTTP Adaptive Streaming (HAS). While these effects on rate-based and buffer-based HAS algorithms have been studied, the recent near-optimal Model Predictive Control (MPC) has not yet been considered. We evaluate the impact of cache presence on MPC and show that caches may cause instabilities in common set-ups. To tackle this issue, we consider an idealized cache-aware extension to MPC and discuss its implementation. Our simulations show that overall cache-unaware MPC benefits from in-network caching with only video quality instability being increased, while cache-awareness brings noticeable improvement in this regard.**

***Keywords*-HTTP adaptive streaming, video caching, QoE, Model Predictive Control.**

I. INTRO

HTTP adaptive video streaming (HAS) has caught a particular attention recently as an efficient way to stream video over network. Within this approach, videos encoded with different quality levels are divided into short segments (from 2 to 10 seconds long) each of them stored as a separate file. In this way, each segment can be requested with a different quality, thus allowing the video player to adapt video quality during the playback according to current network conditions and user preferences. A number of algorithms are proposed to make video quality decisions. For instance, *Rate-based* algorithms are using previous download bit-rate samples; *Buffer-based* approach ([1], [2]), instead, makes decisions relying on the playback buffer occupancy. Recently, Yin et al. [3] proposed a *Model Predictive Control* (MPC) algorithm for segment quality selection in HAS which outperforms previous solutions. MPC combines several QoE metrics in a principled way and solves an optimization problem in order to select video qualities for segments.

Storing each video segment as an HTTP object makes video content easily cacheable. This gives an opportunity to alleviate quickly growing network load caused by considerable increase in Internet video popularity. However, Lee et al. ([4]) have shown a particular issue using rate-based video quality selection with caches. A common network set-up can be simplified as shown at Fig. 1, where connectivity between Client and Cache has higher bit-rate than the other one. In this case, rate-based quality selection algorithm is shown to incorrectly estimate network conditions due to difference in path capacities to the cache and to the video server, thus making wrong quality decisions resulting in severe *Quality of Experience* (QoE) degradation. A number of studies have demonstrated their vision on solving this issue. Some propose traffic shaping to influence quality decisions on client [5], or predict client decisions in order to prefetch potentially necessary segments [6]. Other studies ([1], [2]) do not rely on network entities collaboration, and propose to use playback buffer for quality decisions as it is not directly affected by difference in path capacities.

We have previously studied direct caching impact on the rate-based and buffer-based algorithms and how do they compare to the theoretical optimal [7]. MPC has been shown to perform better compared to the formers in absence of cache but there is no evaluation on the impact of caches on this algorithm. The purpose of this paper, therefore, is to study the effect of cache presence on the MPC video-rate selection algorithm. We show that MPC is subject to increased video quality instability in presence of a cache, and demonstrate that cache-awareness might alleviate this.

The paper is structured as follows. Sec. II provides a description of MPC video quality selection algorithm, and studies the impact of cache on it. Sec. III proposes a way to improve MPC to be cache-aware. Sec. IV presents detailed evaluation of advantages of cache-aware MPC compared to the cache-unaware one. Finally, Sec. V concludes the work.

II. MPC

A. Background on MPC

Yin et al. have identified that the problem of segment video quality selection in *HTTP Adaptive Video Streaming* (HAS) can be formulated as an optimal control problem and proposed the *Model Predictive Control* (MPC) algorithm for selecting the video quality of segments in HAS [3]. The central point of the MPC algorithm is a linear optimization problem, consisting of maximizing the *Quality of Experience* (QoE) value under video playback-related constraints, including buffer dynamics and expected segment download time. Yin et al. define the QoE as the function as in Eq. (1).

$$QOE_1^K = \sum_{k=1}^{K} Q_k - \lambda \sum_{k=1}^{K-1} |Q_{k+1} - Q_k| - \mu \sum_{k=1}^{K} dBp_k - \mu_s T_s \tag{1}$$

This function is a weighted mixture of the following QoE related metrics: **Video Quality** Q, **Buffer Deficit** dBp, **Startup Delay** T_s and **Video Quality Instability** $|Q_{k+1} - Q_k|$. From these metrics, we can derive more informative ones which can be used for a complex QoE assessment:

- **Average Video Quality**, the ratio of sum of all the qualities selected for segments during playback to the number of segments;
- **Total Stall Time**, the cumulative duration of buffer deficit over K segments (or simply the total time playback was blocked during playback);
- **Startup Delay**, the time elapsed between the request for playback and the playing of the first video frame.
- **Average Quality Instability**, the ratio of sum of all quality differences between two consecutive segments to number of segments minus one.
- **Average Buffer Level**, the ratio of sum of all playback buffer levels over number of segments.

Coefficients λ, μ and μ_s are providing a mechanism to tune the expression so as to adapt it to particular QoE objectives. The effects imposed by changing these coefficients (*weights*) are related to the summand(s) having larger weight, i.e., to the QoE metric(s) in focus. For example, combination $(\lambda = 1, \mu = 10, \mu_s = 100)$ will presumably minimize startup delay and stall time on the expense of the video quality and its instability. We will focus on this weight configuration in the rest of the paper as users are generally most distracted by stalls, as shown in [8].

The analytical expression Eq. (1) allows to make a decision about qualities of all video segments at once as a result of optimization. Such an approach, however, can hardly be implemented in reality due to common unavailability of accurate link capacity predictions for the entire movie duration, so authors have called upon using a short *look-ahead window* (e.g., 10 seconds long) as it is possible to develop accurate predictor for such short windows. The MPC algorithm hence is a two-phase algorithm where the first phase is a *Prediction* and the second is an *Optimization*. During prediction phase the download bit-rates for the next segments in the look-ahead window are anticipated; these predictions are used as input to the optimization phase. To ensure QoE maximization, the MPC algorithm forms a control loop where results of decisions are used to decide the quality of future segments. To avoid solving a linear optimization problem before downloading each segment, Yin et al. [3] propose efficient pre-computation and data representation as a potential solution.

B. MPC in presence of a cache

Various studies show that network caches may cause undesirable effects on the video playback QoE in the Internet ([4], [7]). While MPC is extensively evaluated and proven to be nearly optimal in absence of cache, little is known about its ability to deal with network caches.

To understand how MPC behaves in presence of cache, let us assume a reference network consisting of a client and a video server, connected by the intermediate network cache, as depicted on Fig. 1.

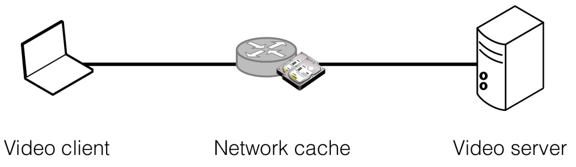
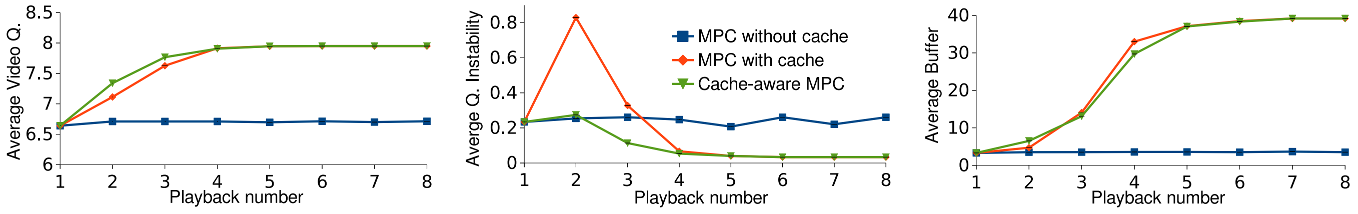


Figure 1. Reference environment

In this network, the link between the client and the cache has considerably larger bit-rate capacity than the one between the cache and the server; such a generalization can be reasonable for modern mobile networks with edge caches. The video to be streamed is encoded into several qualities, where the highest quality has an average bit-rate greater than cache-server link bit-rate, but lower than the other link. Based on [7], we expect that in these conditions segments fetched by MPC will be stored in the cache. During the next playback of the same video, those cached segment qualities will most likely be fetched again, but now from the cache hence with a greater download rate. In this case, MPC will increase its network conditions estimation and fetch segments of higher qualities than in the previous playback, which in turn will also be cached. This process will repeat over several consecutive playbacks until all the segments of the highest quality will become cached, allowing MPC to take full advantage of the faster link and thus to provide better QoE. Our study aims at understanding the behaviour of MPC during this transient phase.

We have implemented the reference environment in our Python simulator using segment quality sizes from real HTTP Live Streaming-ready video (MPEG-2, H.264, Big Buck Bunny[1]). The weight coefficients in MPC objective function are: $\lambda = 1$, $\mu = 10$ and $\mu_s = 100$. In the simulator, we repeated 5 times a sequence of 8 consecutive playback runs of the same video where the cache is flushed before the first run of a sequence. Each run is started after the end of the previous one. This approach mimics independent plays of "not so popular" short videos, i.e., very unlikely to be played by several clients simultaneously. Fig. 2 validates the progressive caching of segments with repetitions of playbacks to eventually reach the highest video quality after 5 playback repetitions. The x-axis shows the video playback run while the y-axis is the average value of the metric of interest over the 5 repetitions with the 95% confidence interval, namely *average video quality*, *average quality instability*, and *average buffer level*. We omit to depict evolution of the stall time as our weight configuration mostly eliminates those.

[1]Available at https://peach.blender.org



(a) Average Video Quality (b) Average Quality Instability (c) Average Buffer Level

Figure 2. MPC performance depending on cache presence and cache-awareness; Client-Cache link is fixed to 2Mbps, Cache-Server link is fixed to 4Mbps

With Fig. 2, one can note that MPC is able to take advantage of the presence of a cache in terms of average video quality played at the client but that it comes at the expense of a higher average quality instability during transience, which could potentially decrease the QoE as users are sensitive to quality fluctuations.

Fig. 2 also points out large average buffer levels differences with and without caches. Playback buffer level is not a subject for optimization, therefore put out of focus in favor of the optimized QoE metrics. Due to this fact, we can see that average buffer levels during playback do not exceed 4 seconds without cache. This means that playback is very likely to stall in case of a major perturbation in the network, which is a major limitation of the objective function shown at Eq. (1). Presence of a cache improves this figures already from the second consecutive run.

III. Cache-awareness

In case of weight configuration focusing on minimizing stalls and startup time, MPC with cache-unaware predictor manages to provide overall high quality playback under cache presence. However, increased average quality instability is observed at the early stages of the transient phase when the cache is being populated. This instability phenomenon may seem minor as it only happens during a short transient phase. However, in case of cache contention the "not-so-popular" videos might remain cached only for a limited amount of time and hence suffer from this phenomenon.

The reason for such an instability in presence of network caches lies in the way cache-unaware MPC determines which quality to use for the segments to retrieve. As shown in Sec. II, MPC indirectly needs to know the download bit-rate of the next segment in order to optimize the maximal average segment quality that can be requested to the server to fulfill playback buffer occupancy requirements. With cache-unaware predictor, this approach yields nearly optimal as long as all the segments share the same network bottleneck (i.e., downloaded with about the same download rate). However, this assumption doesn't hold true when a cache is put between the client and the server. In this situation, as the cache-unaware MPC is unable to determine which segments will be retrieved from the network cache and which ones will be retrieved from the server, it ends up over-estimating the download bit-rate to retrieve segments. It will thus request segments for a quality higher than what can really be retrieved as in the transient phase high quality segments are not all in the cache yet. MPC control loop will then reduce the download rate prediction for the following segments as it measures the actual rate while retrieving segments. However, the lower quality segments requested by MPC might then be retrieved from the cache, which will fool MPC that will again overestimate the download rate and thus request high bit-rate segments that cannot be retrieved as they are not cached yet. This download rate prediction error is at the origin of the video quality instability observed for MPC in case of the presence of a network cache.

To study the *cache-awareness* of MPC we therefore consider a download rate predictor having an exact knowledge of whether segments are coming from cache or not, thus giving an upper bound in algorithm performance. In contrast to predictor in our cache-unaware MPC model, where the entire download rate prediction vector (of length equal to lookahead horizon) is in fact filled with just one EWMA (Exponentially Weighted Moving Average) value, our cache-aware MPC tracks two different EWMAs. One is for segments coming from the cache, and the other one is for segments coming from the server. Based on its knowledge of cache state, our idealized cache-aware predictor is therefore able to predict the expected download rate for each future segment based on its cache or server origin and optimize its segment quality selection according to its objective function. In this way, the cache-aware predictor takes advantage of the cache presence while leaving a room for fluctuating download rates, which it is not aware of.

IV. Evaluation of cache-awareness

To obtain an upper bound of our cache-aware MPC, we simulated a perfect cache-awareness. For this, the simulator records which segments/qualities have been requested during each consecutive playback run and keeps this information between runs, thus providing perfect cache-aware predictor with a complete cache state.

To evaluate our cache-aware predictor we implemented a trace-driven simulator in Python. It uses segment sizes from a real HLS-ready video (Big Buck Bunny) at the input of the optimization in order to compute its runtime parameters such as playback buffer occupancy and video quality decisions. The video consists of 300 chunks of 2 seconds each, and is encoded into 8 qualities with average bitrate of 350, 470, 630, 850, 1150, 1520, 2040, and 2750 kbit/sec.

The cache is simulated according to Fig. 1. In the simulator it is operating by assigning download bit-rates of Client-Cache or Cache-Server links to particular qualities of particular segments. These download bit-rates are, in turn, used for calculating the actual playback buffer after reception of each segment as a function of previous playback buffer occupancy, selected segment quality size, and the mentioned download bit-rate.

We simulate link fluctuations with a mobile dataset [9], which has a collection of download bit-rate traces from Telenor HSPA network experienced in a number of scenarios (e.g., city bus commute, intercity car trip). To make it suitable for our simulations, we have joined the download bit-rate traces from several dataset scenarios having close environment, with the resulting tracefile having 1213 samples. We have only selected bitrate traces larger than 60000 Bytes per second. [2] In order to achieve the required average bit-rate, each trace sample is being multiplied by a factor in the simulation process. This factor is specifically calculated to set the average of all trace samples to a certain value. In the simulation, we apply this trace by assigning its samples one by one to each quality of each segment, so that each of the latter is associated with a unique download rate sample, with which it will be downloaded in simulation. Three different connectivity configurations are considered in our simulator:

- ***f*** (fixed), where both links have fixed capacity: Client-Cache is 4 Mbps, Cache-Server is 2 Mbps. Note that the Cache-Server link capacity is lower than average bit-rate of the maximum video quality encoding.
- ***cv*** (client variation), where Client-Cache link is fluctuating with an average rate of 4Mbps while the Cache-Server link is stable at 2 Mbps.
- ***sv*** (server variation), where Cache-Server link is fluctuating with an average rate of 2 Mbps while the Client-Cache link is stable at 4 Mbps.

For the reference simulation, the following parameters are selected: maximum playback buffer is 40 seconds; look-ahead window is 5; video quality instability weight is 1; buffer deficit weight is 10; startup delay weight is 100. this configuration is equivalent to the one recommended by Yin et al. [3] with only exception of large startup delay weight used for fixing the latter to the same value (0) across all the simulations. Each algorithm is being run eight consecutive times (runs), within each of them cache state is kept. This experiment of eight runs is repeated five times for statistical analysis, with cache being flushed before first run of a new repetition. This parameters resemble ones used in Sec. II-B.

A. Cache-aware predictor evaluation

First of all, we have simulated perfectly cache-aware predictor with the reference parameters in order to directly compare its results with cache-unaware MPC implementation. Simulations (presented by the curve labeled "Cache-aware MPC" in Fig. 2) have shown that cache-awareness can significantly alleviate average quality instability compared to the cache-unaware MPC model, while the rest of the shown metrics are not experiencing any noticeable improvement. In order to more deeply observe the effect

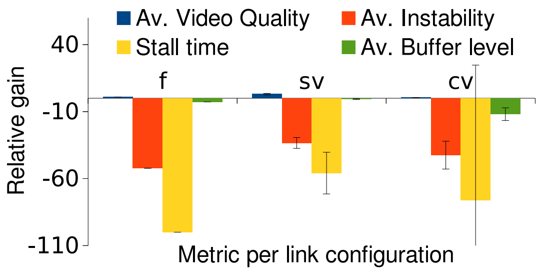


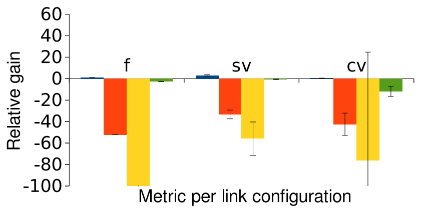
Figure 3. Relative advantage of cache-aware MPC to cache-unaware MPC in a presence of a cache, for reference parameters

of cache-awareness, we have simulated communication link capacity fluctuations. Fig. 3 depicts relative gain in percents in total over transient phase of QoE metrics of perfect cache-awareness compared to cache-unaware MPC with a confidence interval of 95%, i.e., how proportionally larger is the value of former compared to latter. It can be seen from the curves that, alike to the configuration with fixed links, cache-aware model improves noticeably the average quality instability performance (i.e., decreasing its value) but does not introduce any significant advantage for average video quality and average buffer level. Additionally, Fig. 3 shows an important difference of *total stall time*; though relative advantage of cache-aware predictor in total stall time is considerable, the absolute values are often negligible, especially in case of *cv* and *f* connectivity configurations. The problem of low playback buffer, mentioned in Sec. II-B, becomes noticeable when network connectivity is configured as *sv*, which makes Cache-Server capacity prediction very difficult. Fig. 4 shows the stall times over consecutive runs in the mentioned case. As can be seen, stalls are not completely suppressed with cache-awareness; rather, cache-awareness eliminates stalls after less consecutive runs – as compared to the cache-unaware algorithm.

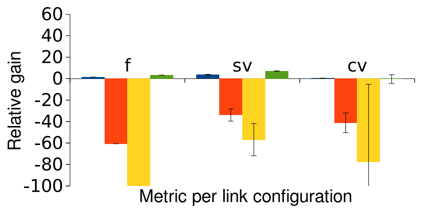
B. Sensitivity analysis

In order to study how does cache-awareness compared to the cache-unaware MPC under different setup parameters,

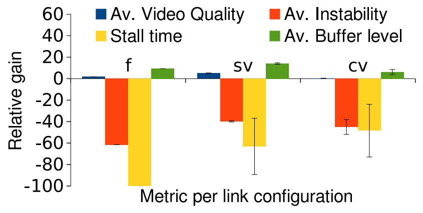
[2]This is done to alleviate inherent drawback of cache behaviour modeling in our simulation. As download bit-rate is assigned to the particular quality of a segment, accepting low bit-rate traces may cause unrealistic situation when large segment, e.g., of 500 kB, will be assigned a very low download bit-rate, e.g., of 10 kBps. In this case the download time will be simulated as 50 seconds, during which the communication link bit-rate would have become larger in real world so segment would have downloaded faster.



(a) Maximum buffer 40, window 5

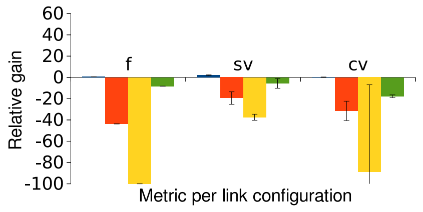


(b) Maximum buffer 20, window 5

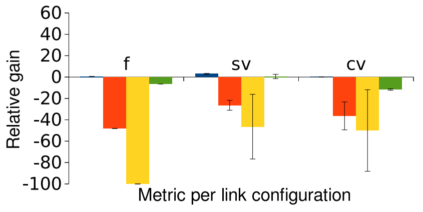


(c) Maximum buffer 10, window 5

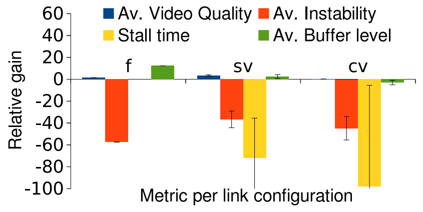
Figure 5. Sensitivity to maximum buffer capacity



(a) Maximum buffer 40, window 2

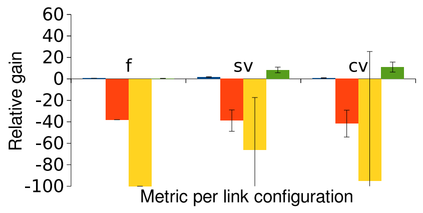


(b) Maximum buffer 40, window 3

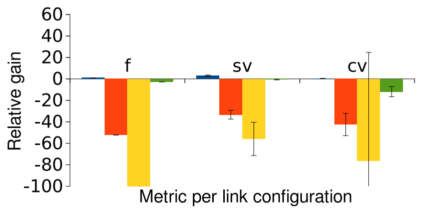


(c) Maximum buffer 40, window 10

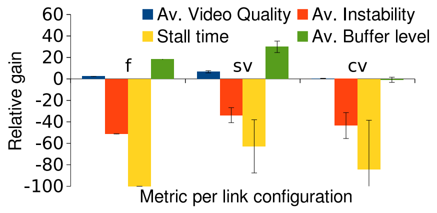
Figure 6. Sensitivity to look-ahead window length



(a) Maximum buffer 40, window 5, 4 qualities



(b) Maximum buffer 40, window 5, 8 qualities



(c) Maximum buffer 40, window 5, 16 qualities

Figure 7. Sensitivity to number of video qualities

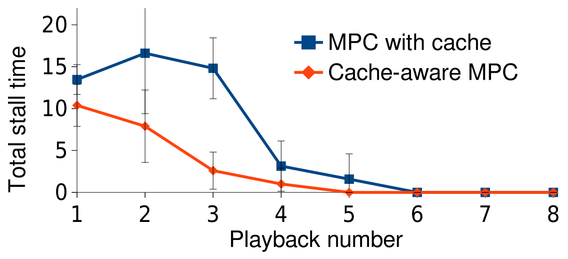


Figure 4. Total stall time over runs for *sv* connectivity configuration and reference maximum buffer, look-ahead window and weights

we perform a sensitivity analysis. The following varying parameters are examined:

- Maximum playback buffer: 10, 20, 40 sec;
- Look-ahead window: 2, 3, 5, 10 segments;
- Number of qualities the video is encoded into: 4, 8, 16.

Each combination of these parameters has been explored. The results are represented in the same manner as in Fig. 3.

1) Maximum buffer: Small playback buffers help the cache-aware predictor to reveal its strengths. As can be seen from Fig. 5, not all the parameters are clearly improved, but the relative gain in average buffer level shows a trend of increasing with lower maximum buffer. On the other hand, the absolute performance over the transient phase is better with larger buffers. This can be underpinned by the fact that very small buffers give less margins to manoeuvre for the cache-unaware algorithm when facing severe link capacity fluctuations due to caching. On the contrary, large enough buffers (able to keep 10 – 20 video segments) make the cache-unaware MPC more robust to these fluctuations, thus lowering numerical advantage of a cache-aware predictor. Finally, results from large buffers ($>$ 30 segments) are not significantly different from the previous ones.

2) Look-ahead window: Fig. 6 gives an insight regarding sensitivity of cache-aware advantage to different look-ahead windows. It can be noted that larger windows do allow cache-aware predictor to yield even better results compared to cache-unaware MPC. Average video instability and average buffer level are noticeably better with window of 10 than with window of 2, while average video quality and total stall time remain statistically the same. In absolute values, however, the performance of the algorithms are being improved with growing look-ahead window, especially comparing the shortest ones (2 to 3 segments long).

3) Number of qualities in video encoding: As the original video has been encoded into 8 qualities and because relation of video file sizes to their qualities is non-linear, we used logarithmic interpolation to make synthetic segment sizes for 4 and 16 qualities levels.

Generally, the advantage of cache-awareness grows with growing number of qualities, in configuration with reference parameters (Fig. 7). All the metrics are being improved with increasing number of qualities, but effect on average quality instability and total buffer level shows particular improvement. In absolute values, increasing the number of qualities is observed to induce more average instability and larger total stall time. The reason for this is that larger number of qualities provides a capability of a more fine-grained quality selection which might decrease video quality instability with cache-awareness, whereas the cache-unaware MPC would need to jump between more quality levels as to handle the fluctuating segment download rate.

C. Practical issues of determining cache state

Accurately predicting the cache state (i.e., which segment qualities were cached before) is a complicated task without directly collaborating with cache. To achieve this without the latter, we propose to use *intra-video popularity profiles* (where each segment is assigned a probability of being watched) leveraging the user retention distribution, and to estimate quality decisions during previous video playback.

Our considerations are built upon the fact that average video quality grows gradually over playback runs, thus suggesting that segment quality decisions are usually close between two consecutive runs. If the download bit-rate of segment K is noticeably larger than of previous segments', it is likely that it comes from a cache. Using the intra-video popularity profile, we can estimate the probability of next segment to be cached. If the profile suggests (comparing it with thresholds) that segment $K+1$ will be cached, but its download bit-rate is too low (i.e., likely that it came from video server), then we might presume that previous client's playback buffer could have depleted hence it could have backed off to a lower quality. In this case we can try to estimate which quality it was by performing optimization with the same runtime parameters except for playback buffer, which we believe was zero for the previous client. Such an approach can potentially yield good results, in case if the intra-video popularity profile is accurate and algorithm's threshold for cached/not cached decisions are adjusted upon each wrong decision. Improvement notwithstanding, discussed ideas will significantly increase algorithm complexity. We therefore let its implementation and evaluation for future work.

V. Conclusion

In-network caching is one of the effective methods to tackle the problem of quickly growing video traffic but various studies show that caches may have a negative impact on Quality of Experience when HTTP adaptive video streaming is used. In this paper, we assess the impact of network caches on the cache-unaware MPC quality selection algorithm. We demonstrate that, in cases when the MPC objective function is focused on eliminating stalls, it takes advantage of caching in terms of video quality on the expense of its instability. The reason for the latter is that cache-unaware MPC is not able to accurately predict download bit-rate when cache is present. In order to identify whether cache-awareness can confront this drawback, we propose an upper bound for cache-aware predictor to MPC that knows the exact cache state at any point of time. Though significantly increasing algorithm complexity in a practical implementation, such a predictor has shown to bring noticeable improvement by reducing video quality instability.

References

[1] 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," *ACM SIGCOMM Comp. Comm. Review*, vol. 44, no. 4, pp. 187–198, 2015.

[2] H. Yunfeng, "Cache-friendly Rate Adaptation for Dynamic Adaptive Streaming over HTTP (DASH)," May 2015.

[3] 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 SIGCOMM*. ACM, 2015, pp. 325–338.

[4] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in http adaptive streaming: Friend or foe?" in *ACM NOSSDAV*, 2014, p. 31.

[5] C. Kreuzberger, B. Rainer, and H. Hellwagner, "Modelling the impact of caching and popularity on concurrent adaptive multimedia streams in information-centric networks," in *IEEE ICMEW, June 2015*. IEEE, 2015, pp. 1–6.

[6] P. Juluri and D. Medhi, "Cache'n dash: Efficient caching for dash," in *Proceedings of the 2015 ACM SIGCOMM*. ACM, 2015, pp. 599–600.

[7] V. Poliakov, L. Sassatelli, and D. Saucez, "Impact of caching on http adaptive streaming decisions: towards an optimal," in *IEEE Infocom, Student Workshop*, 2016.

[8] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *Multimedia (ISM), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 494–499.

[9] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 2013, pp. 114–118.