A simple yet effective network-assisted signal for enhanced DASH quality of experience

Omitted due to double blind submission

ABSTRACT

We propose and evaluate simple signals coming from in-network telemetry that are effective to enhance the quality of DASH streaming. Specifically, in-network caching is known to positively affect DASH streaming quality but at the same time negatively affect the controller stability, increasing the quality switch ratio. Our contributions are to first (i) consider the broad spectrum of interaction between the network and the application, and then (ii) to devise how to effectively exploit in a DASH controller a very simple signal (i.e., per-quality hit ratio) that can be exported by framework such as Server and Network Assisted DASH (SAND) at fairly low rate (i.e., a timescale of 10s of seconds). Our thorough experimental campaign confirms the soundness of the approach (that significantly ameliorate performance with respect to network-blind DASH), as well as its robustness (i.e., tuning is not critical) and practical appeal (i.e., due to its simplicity and compatibility with SAND).

1 INTRODUCTION

According to Cisco VNI [2], video traffic will account for over 80% of all IP traffic by 2021, making video the predominant network application. In recent years, several *over-the-top* techniques emerged to efficiently deliver videos over the Internet, both proprietary such as Microsoft HSS and Apple HLS, as well as standard-based as in the case of Dynamic Adaptive Streaming over HTTP (DASH). At the same time, to reduce the pressure that video represent on the network infrastructure, we also observe a growing offer of *network frameworks*, such as Server and Network Assisted DASH (SAND) or Information Centric Networks (ICN), as well as *network functions*, such as in-network caching and multi-path forwarding, that are meant to effectively assist the video delivery.

Whereas the literature abounds with controllers[16], there is no systematic study of the controller interaction with network functions (but, see Sec.2 for an overview). In particular, we argue that it would be desirable for any network-controller mechanism to be as lightweight as possible, limiting the *amount* of information that it needs to collect from the network and disseminate to clients, as well as the *rate* at which the information needs to be disseminated. Additionally, it would be desirable for this signal to be easily pluggable within existing DASH controllers — i.e., naturally extending their logic by fitting the additional information provided by the network, as opposite to requiring a complete redesign around it.

In this paper, we tackle this challenge by considering a specific network function, namely *in-network caching*, that is known to have potential for relieving traffic load on the one hand, but that can possibly induce quality oscillations [18]. To reduce cache-induced oscillation, shaping is seldom used to reduce the rate toward the cache [15] (which practically limits its usefulness), or the assistance of a centralized resource manager is usually assumed [5] (which requires global instantaneous knowledge and decision of all cached content and is too heavyweight to have practical relevance). Specifically, our aim is to understand the feasibility of a SAND interaction that (i) uses the most lightweight network signal, (ii) requires only minimal changes to existing DASH logics, and can (iii) maximise cache usefulness and therefore increasing the average video quality, while (iv) avoiding cache-induced oscillations at the same time. Summarizing our main contributions:

- we systematically assess the impact of in-network caching when several ICN DAS players compete to watch a video, exploring the boundaries of the design space for network vs client interaction using various adaptation logics (Sec. 3)
- we propose and evaluate (Sect. 4) a network-aware evolution of an existing adaptation logic (specifically, AdapTech [3]), based on simple network signals (specifically, per-quality cache hit-rates) exported at low rate timescales (specifically, tens of seconds).

Our results, gathered over an experimental campaign worth *several weeks* of video streaming, show this to be a promising and viable direction. We plan to make our code available to the scientific community (which is impossible at submission time due to double-blind requirement), along with instructions to reproduce results of this paper at [1].

2 RELATED WORK

HTTP Adaptive Streaming (HAS) systems traditionally delegate bitrate selection and control to the client application, which takes independent decisions based on local estimates of available end-to-end bandwidth and awareness of buffer level dynamics. Recently, many studies have shown the benefits of network assistance in HAS to overcome the limitations of a purely client-driven scheme (see e.g. [3, 5–7, 9, 12, 17, 25]). The lack of direct knowledge about network status (congestion, bottleneck, cached content) as well as of coordination among concurrent flows may result in frequent quality/bitrate oscillations, sub-optimal bitrate decisions by the clients, unfairness among ongoing flows and inefficient overall utilization of network resources. Moreover, service differentiation and management policies cannot be guaranteed in purely client-driven HAS. As such, a dedicated MPEG standard has been developed to formalize the interaction between client and network elements under the name of Server and Network Assisted DASH (SAND)[27]. While there is work focusing on assistance of multiple network elements and more complex control systems (e.g., CDNs [8, 11, 21], SDN [4, 5, 12, 22] or network of caches in ICN [19, 24]), due to space constraint we focus here on work that consider the assistance of a single network element (i.e., server, intermediate cache, edge router), which is closer to our work. In particular, we restrain our attention to work that leverage nodes caching capabilities, be it on general purpose architectures [6, 10, 18, 20] or ICN-specific [13, 15, 23].

Submitted for peer review,
2018. 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

The fact that in-network caches may induce quality oscillations at the clients and thus a degradation of the QoE has been observed in [18]: the incorrect estimation of the available bandwidth at the client, due to a possible overestimation of the available bandwidth in case of retrieval from the cache, can lead to bitrate oscillations in case of cache miss. To tackle this issue, [18] introduces ViSIC, an intelligent cache which assists the bitrate adaptation by performing *shaping* of traffic from the cache. Common to [6, 10, 20] is to consider a multiuser HAS distribution and propose to optimize the QoE of all clients, either by means of a proxy-assisted in-network adaptation [10, 20], or by means of an ILP optimisation (in a centralized or decentralized way) [6]. In the above work, an optimal *target bitrate* is computed and then notified to the clients (either by modifying the HTTP requests at the proxy [10], or by a modified HAS logic that enforces clients selection of the optimal target bitrate [6, 20]). In all the above cases, the capability to adapt rate at the client is traded-off for a global optimization of resources' allocation: taking the purely client-driven and network-driven solutions as references, we instead investigate the space of solutions that combine network-awareness and user-centric QoE, augmenting client-driven rate adaptation with clues provided by the network.

A cache-assisted HAS approach is developed in [23] under the assumption of Scalable Video Coding (SVC): an intermediate ICN cache monitors all the requests, that it either forwards or drops (sending a NACK to the client) with the intent of *steering* client requested quality. Whereas SVC plays nicely with caches, it is however not currently widely deployed and we thus disregard it in what follows. In [13], the authors illustrate the downsides of video delivery over ICN in presence of in-network caching: decreased cache hit and quality oscillations arise when the client-cache path is significantly better than the client-server one. However, under the assumption that a distinction can be made between a content served by the server vs the cache, the client can then manage two *separate rate estimators* (i.e., for the server vs cache), which we also consider in this work. To reduce bitrate oscillations the intermediate cache further performs live *transcoding*: however, transcoding is a heavyweight operation requiring significant amount of computing resources, which is contrary to our design goal. Recently, [15] suggested *shaping* at the cache to avoid oscillations. The shaping is done to guide the client to the next decision: if the next segment is present in cache, at a given quality, the shaping will be done to match the bitrate of this quality – which as previously observed diminishes adaptiveness to variable network and client conditions, and that we thus avoid in our work.

To broaden the scope of previous work, we assess in a more systematic fashion whether (and how) network-assistance can be beneficial also in non-controlled network environments (like the mobile access) and in presence of purely reactive caching, with the objective of preserving dynamic bitrate adaptation at the client.

3 TO CACHE OR NOT TO CACHE?

While caching can be beneficial to DASH by *increasing the average quality* (e.g., as typically the bandwidth to the edge-cache is larger than that toward the end-server), it may also negatively impact performance *by increasing the quality switch ratio* (e.g., in case of a cache miss, which can further lead to oscillations). To understand which conditions lead to performance impairment, and how to avoid it, it is important to systematically study the design space of DASH interaction with in-network caches.

3.1 Design space

We consider a reference baseline scenario (without in-network caches) and contrast it with in-network caching scenarios (with proactive vs reactive policies), considering both network-blind (buffer vs rate based) and network-aware adaptation (with either a strict or soft interpretation of the network feedback).

3.1.1 Application: DASH adaptation logic. We consider two representative examples of network-blind DASH controller.

Buffer-Based Algorithm (BBA). Introduced in [14], this simple and robust algorithm is agnostic to the network conditions, and its decisions are only driven by the buffer state. In a nutshell, BBA[1] defines two buffer thresholds, $B_{\min}$ and B_{upper} and uses the buffer level $B(t)$ at the client to take decisions. If $0 \leq B(t) \leq B_{\min}$, the quality selected is the lowest. If $B(t) \geqslant B_{\text{upper}}$, the quality selected is the highest one. When $B_{\min} \leq B(t) \leq B_{\text{upper}}$, a linear mapping is done from the buffer level to the selected quality.

AdapTech. Introduced in [3], AdapTech is an hybrid adaptation logic, relying on both rate estimation and buffer level. In a nutshell, AdapTech defines two buffer thresholds, $B_{\min}$ and B_{steady}, which define three zones in the buffer: ① the *panic* zone, when $0 \leq B(t) \leq B_{\min}$, ② the *buffering-state* zone, $B_{\min} < B(t) \leq B_{\text{steady}}$ and ③ the *steady-state* zone, $B_{\text{steady}} < B(t) \leq B_{\text{ax}}$. When the player is in the ① panic zone, it selects the lowest quality $q = 0$ in order to avoid rebuffering events. In the ② buffering-state zone, if the throughput estimation for the last video segment is higher than the bitrate of the next quality ($BW \geq b_{q+1}$), then the quality is increased. Conversely, if the throughput estimation for the last video segment is lower than the actual bitrate, $BW < b_q$ then the quality is decreased. At last, in the ③ steady-state zone, the quality is never decreased, in order to avoid overreaction to negative spikes in the available bandwidth. The quality can be increased if over the last T seconds, the average throughput estimate is higher than the next bitrate $\widehat{BW} > b_{q+1}$ and the throughput of the last download is higher than the bitrate of the next higher quality $BW > b_{q+1}$.

3.1.2 Network: Cache policies. We consider three scenarios, and devise two modes of interaction between network and application.

Baseline. No caches are considered in the network. On the client side, network-blind BBA and AdapTech logics are used.

Proactive placement, no cache replacement. A proactive placement is performed at the cache: one quality is fully cached at the router (i.e., all the data packets of all the segments of the selected quality are cached, and there is no cache replacement). Moreover, the router advertises the quality that is in cache to the clients via annotations in the MPD file. *Network-blind* clients resorts to their unmodified BBA and AdapTech adaptation logics to select the quality of the next segment. Conversely, *network-aware* clients can interact with two modes: ① a *strict* one, that forces the client to

[1] We use the BBA-0 algorithm, referred to as BBA in the remainder of this paper

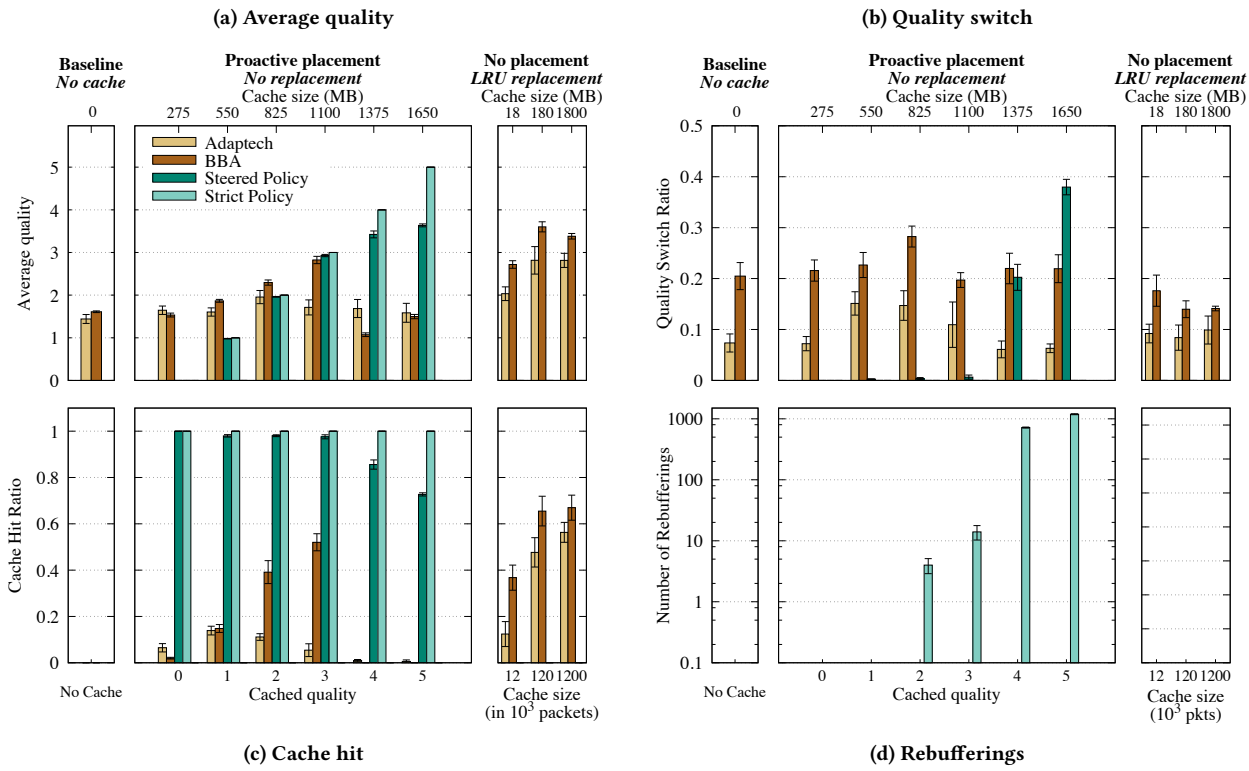

Figure 1: Design space at a glance: plots (a)-(d) show a key performance metrics, with different subplots for different scenarios, i.e., baseline (left), proactive placement without replacement (middle) and no placement with LRU replacement (right). The bottom x-axis reports the quality cached at the router (in the proactive placement scenario) or the cache size in number of data packets (in the LRU replacement scenario), while the top x-axis reports the equivalent cache size in MB (both scenarios).

download segments at the quality advertised by the network, regardless of its state (buffer level and rate estimation); ② a *steered* one, that follows the router indication unless his buffer level is below a given threshold (B_{min} = 20% in our experiments), in which case the client downloads at the lowest quality $q = 0$ (i.e., panic mode as in AdaptTech).

No proactive placement, LRU replacement. No proactive placement is performed and the cache uses a LRU replacement as cache management policy. On the client side, network-blind BBA and AdapTech logics are used.

3.2 Results at a glance

3.2.1 Scenarios. For the sake of simplicity, we consider six emulated clients connected (via Wi-Fi 802.11n or Ethernet) to an intermediate router (possibly equipped with a transparent cache of controlled size S) connected to a video server via an Ethernet link (of controlled capacity C). All the nodes (i.e., clients, router, servers) are ICN-enabled (using the ICN stack of the Linux Foundation CICN project [26]), and each client runs an instance of Viper (the default dual-stack TCP/IP and ICN video player of CICN). We set the buffer capacity to 20 video segments, and uses the defaults parameters of the adaptation algorithms early indicated. The server hosts the Tears of Steel video encoded at 6 different qualities, identified by their bitrates (i.e., 3, 6, 9, 12, 15 and 18 Mbps).

Without loss of generality, we report here a case where we set the capacity C = 60 Mbps, and connect the clients to the intermediate router using a WiFi 802.11n link: clients are close to the access point, so that the WiFi channel capacity is about 100 Mbps. In order to prevent PIT aggregation from occurring at startup in our experiments (which would lead to a multicast tree to naturally form in ICN), we introduce stochastic arrivals (with average interarrival of two seconds). For each scenario, each player downloads once the video and we gather 95% confidence interval over 10 runs.

3.2.2 Experimental results. We contrast in Fig. 1 the wide boundaries of the design space with the usual metrics: the average quality perceived by all clients (Fig. 1a), the number of rebuffering events (Fig. 1d), the ratio of quality switches (the number of quality switches divided by the number of segments downloaded, Fig. 1b) and the cache hit ratio (whenever relevant, Fig. 1c). Each subfigure is further divided in three plots, one for each scenario: baseline (left), proactive placement (middle) and reactive LRU cache (right). Whenever relevant, the cache size is reported in bytes (top x-axis) and the placed quality or the equivalent number of data packets are also indicated (bottom x-axis).

Baseline. In the *baseline* scenario, when no cache is available and there is no PIT aggregation, the bottleneck for each six clients is the router-server link: the fair-share is about 10 Mbps, so that the highest viable quality is $q = 2$ (bitrate $b_2 = 9$ Mbps). However, due to the segment size fluctuation, the average quality is around 1.5 for both BBA and AdapTech.

Proactive placement. For *Network-blind* clients, placement may be inefficient when there is a mismatch between the available resources and the cached qualities: these will be unlikely to be requested by the clients, either because too high qualities are cached and there is not enough bandwidth to request them (e.g., $b_4 = 15$ and $b_5 = 18$ Mbps exceed the fair-share of the WiFi access to the cache), or because the cached quality is too low in terms of bitrate compared to the bottleneck capacity (e.g., $b_0 = 3$ and $b_1 = 6$ Mbps are lower than the fair-share of each client on the bottleneck link between the router and the original server). As such, placing either the lowest or the highest qualities (qualities 0,1,4 and 5) does not result in measurable gain compared to the baseline scenario, while the cache hit is low (up to 15% for quality 1, close to 0% for the other qualities). Rather, wrong placement can even worsen the user QoE: placing $q = 1$ induces quality oscillations (the quality switch ratio for AdapTech nearly doubles), while the other metrics are not impacted.

Network-aware strategies, that are informed of the cached qualities, do not necessarily benefit from proactive placement either: if the cached quality is too low (qualities 0 and 1), the average quality of all clients is below the baseline scenario. Conversely, when the cached quality is too high (qualities 4 and 5), we see an improvement of the average quality, but at the costs of either rebufferings (strict policy), or quality switches (steered policy). A well dimensioned proactive placement (e.g., which can be the result of an optimization problem) exacerbates the tradeoff between average quality and quality switch for both *network-blind* and *network-aware* clients. For instance, placing $q = 2$ or 3 in our scenario induces a cache hit increases for BBA (respectively 40% and 50%), along with a slight increase of both the average quality and of the quality switch ratio[2]. Even with well dimensioned proactive placement, AdapTech still suffers from cache-induced quality oscillations: this can be inferred by a low cache hit ratio, a higher quality switch ratio and no significant improvement of the average quality. It is worth noting that network-aware policies with proactive placement yield better results in terms of cache hit ratio and better (than AdapTech) or comparable (to BBA) results for the average quality, but do not eradicate oscillations.

Reactive LRU caches. Finally, when a LRU replacement is done at the cache, we observe an improvement compared to the other scenarios: the average quality and cache hit ratio are higher, while the quality switch ratio remains more or less the same. It is worth noting that the size of the cache has some influence on the performance of the clients: when the cache has a small size (18 MB), the average quality is less than the one observed with a bigger cache (180 MB or 1800 MB).

[2]This is due to BBA's linear mapping from the buffer level to the quality requested: when $q = 2$ is cached, it will be quickly retrieved from the cache, thus the client will eventually request $q = 3$, which will take longer to download, depleting the buffer and causing the player to request $q = 2$, and so on.

4 NETWORK-AWARE DASH PROPOSAL

From Sec. 3, we learn that caching increases average quality but possibly induce quality oscillations. Proactive placement is cumbersome (as it should carefully take into account available and time-varying resources) whereas LRU caches are simpler and thus appealing. At the same time, we observe that advertising the cached quality to guide the client choice can improve the overall QoE: while advertisement is natural in the proactive placement case, benefits should arise also under reactive LRU caches. Indeed, LRU caches are driven by the controller requests, which would thus benefit from informed assistance from the network to increase cache efficiency and reduce oscillations: the simplest possible signal that an LRU cache can track (at low overhead) and export (through SAND, at low rate) is the *average per-quality hit-ratio*. We now show how network-aware DASH clients can turn this simple indication to a useful knob to refine their decision process by simply performing throughput estimations on a per-path basis.

4.1 NA^2: Network-Aware AdapTech

We enable clients to differentiate the source of each ICN Data packet by using a path label. This allows to dicriminate server vs cache traffic, so that clients can keep track of throughput estimations on a per-path basis: one estimation for the throughput toward the cache and one toward the server. Furthermore, we enable the cache to periodically advertise to the clients a per-quality pair of signals: the average hit-ratio and number of samples (ICN Interest packets). This advertisement can be done using SAND, but it can also be achieved by updating the MPD on the fly at the cache in cases where the MPD is periodically updated (e.g., MPD live). By combining theses informations, the client can make an educated choice on the quality of the next segment to download. Algorithm 1 describes Network-Aware AdapTech (NA^2), a modified version of AdapTech taking into account the in-network assistance provided by both path-labelling and cache advertisements.

Like AdapTech, NA^2 divides the buffer in three zones: the ① *panic* zone, the ② *growing* zone and the ③ *steady* zone, delimited by two thresholds: B_{panic} and B_{steady}. In the ① panic zone, the lowest quality is selected in order to quickly fill the buffer as to avoid rebuffering events that are harmful to the user QoE. In the ② growing and ③ steady zones, selection is a two-step process: first we compute the *feasibility* of each considered quality and second, we select the highest feasible quality. A quality q is *feasible* if the downloading rate BW is higher than the associated bitrate b_q, i.e., the segment is downloaded faster than viewed. Specifically, the rate is multiplied by a conservative slack factor δ to account for size variations across segments, and the instantaneous BW or average $\widetilde{BW}$ rates are used depending on the buffer state.

Network-awareness kicks in zones ② and ③. If there are not enough samples for this quality ($N_q < T_{samples}$), the informations provided by the cache are not significant and therefore a conservative choice is made, by using the estimated throughput to the server to compute the quality's feasibility. If there are enough samples, the average per-quality cache-hit ratio P_q is segmented in three zones: ④ cold ($P_q \leq P_{Low}$), ⑤ warm ($P_{Low} < P_q \leq P_{High}$) and ⑥ hot ($P_q > P_{High}$) cache. In the ④ cold zone, it is likely that segments of quality q are not cached, and will be downloaded from the server,

Algorithm 1 NA^2: Network-Assisted AdapTech

$B(t)$ ▷ Buffer level
BW_S, BW_C ▷ Server and cache instant throughput
$\widehat{BW}_S, \widehat{BW}_C$ ▷ Server and cache average throughput
$\{P_i\}_{1\le i\le M}, \{N_i\}_{1\le i\le M}$ ▷ Per quality cache hit and samples no.
q, b_q ▷ Current quality and associated bitrate

if $B(t) \le \mathrm{B}_{\mathrm{panic}}$ **then** ▷ ① Panic
  $q \leftarrow 1$
else if $B(t) \le \mathrm{B}_{\mathrm{steady}}$ **then** ▷ ② Growing
  $q \leftarrow \underset{i\in[\![q-1,q+1]\!]}{\mathrm{argmax}}$ isFeasible(i, BW_S, BW_C, growing)
else if $B(t) > \mathrm{B}_{\mathrm{steady}}$ **then** ▷ ③ Steady
  if isFeasible(q+1, $\widehat{BW}_S, \widehat{BW}_C$,steady) && CSU **then**
    $q \leftarrow q + 1$
return q

function isFeasible($q, BW_S, BW_C, state$)
  if $(N_q < T_{Samples})$ || $(P_q \le T_{Low})$ **then** ▷ ④ Cold
    return $(BW_S \times \delta > b_q)$
  else
    if $P_q \le T_{High}$ **then** ▷ ⑤ Warm
      if *state* = growing **then**
        return $(BW_S \times \delta > b_q)$ && $(BW_C \times \delta > b_q)$
      else
        return $(BW_S \times \delta > b_q)$ || $(BW_C \times \delta > b_q)$
    else ▷ ⑥ Hot
      return $(BW_C \times \delta > b_q)$

therefore the estimated throughput to the *server* is used to *conservatively* compute the quality's feasibility. In the ⑥ hot zone, it is likely that the quality is cached and thus the estimated throughput to the *cache* is used. Finally, in the ⑤ warm zone both estimates are used: in the ② growing buffer state, the main objective is to fill the buffer, therefore a conservative choice is made (i.e., the quality has to be feasible for both paths), while in the ③ steady buffer state, the buffer level is high enough to allow for a more optimistic choice (i.e., the quality has to be feasible for at least one of the two paths).

As in AdapTech, we restrict the magnitude of a quality switch to one. As a result, in the ② grow and ③ zones, we consider only the current quality and the ones directly below and above it (when possible). Note that, just like AdapTech, in the steady zone, we can only increase the quality if for at least T seconds, the network conditions allow us to switch to a higher quality: at that point, the can-switch-up (CSU) flag is set to $True$.

4.2 Experimental evaluation

4.2.1 Sensitivity analysis. To assess the impact of the different parameters of our algorithm on the user QoE, we use the early described topology, in which all six clients are connected to the router using an Ethernet link with a capacity of 30 Mbps. The router is connected to the server via an Ethernet link of capacity C = 30 Mbps. To avoid PIT aggregation, we introduce stochastic arrivals (with average interarrival of 6 seconds). We set the cache capacity of the router to 1.8 GB (corresponding to 1.2 M Data packets). We use

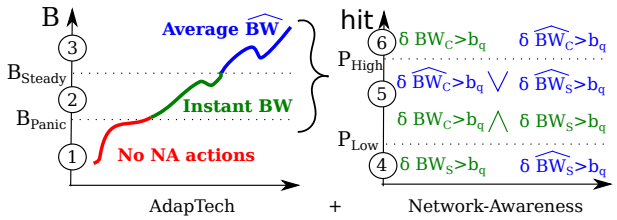

Figure 2: Synoptic of AdapTech (left) + Network-Aware (right) decisions

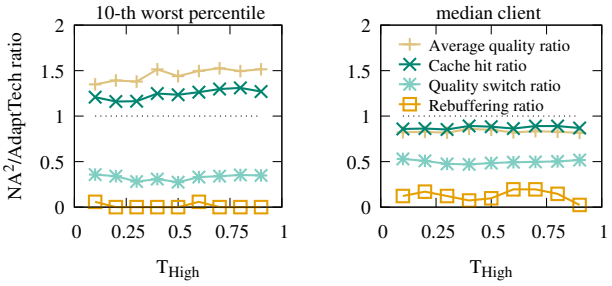

Figure 3: Sensitivity analysis: ratio of Network Aware vs Network blind performance for the 10-th worst percentile of clients (left) and for the median client (right).

default AdapTech values for $B_{panic} = 10s$, $B_{steady} = 20s$, $\delta = 0.8$. For Network-awareness, we advertise the cache hit-ratio every 30 seconds, and require to have collected at least $T_{Samples} = 10^4$ packets. Assuming that the available bandwidth to the server is lower than the one to the cache, the higher T_{Low} is, the more conservative our algorithm is. We thus set $T_{Low} = 0.1$ to avoid being too conservative and we vary T_{High} from 0.1 to 0.9. For each value, we repeat experiments 40 times, for a total of over 100 hours work of experiments.

Fig. 3 presents the ratio of the usual metrics of our algorithm vs baseline AdapTech for the 10-th worst percentile of clients (left) and the median client (right). We see that our algorithm significantly reduces the number of *quality switches* (by about a factor of 2 in both cases) and drastically cut the *rebuffering rate* (by about one order of magnitude for both cases). In the case of the median client, this is done as expected at the expense of the average quality, which reduces by about 15%. Interestingly, for the worst 10-percentile of clients, the average quality actually increases by 50%, which is again a sizeable improvement. Finally, note that results are very stable irrespectively of the exact (T_{Low}, T_{High}) parameterization: we can observe the upper bound of average quality is obtained at (0.1, 0.35) and the lower bound of quality switch at (0.1, 0.5).

4.2.2 Comparison with Network-blind baseline. We finally vary the cache capacity of the router between 90MB and 1.8 GB (respectively corresponding to 60k and 1.2M packets). For each cache size, we run 20 experiments for each adaptation logic: network-blind AdapTech, NA^2 (upper bound), and NA^2 (lower bound), for yielding a total of 120 experiments. The results are presented in Fig. 4, in terms of average quality, quality switch ratio, cache hit ratio and

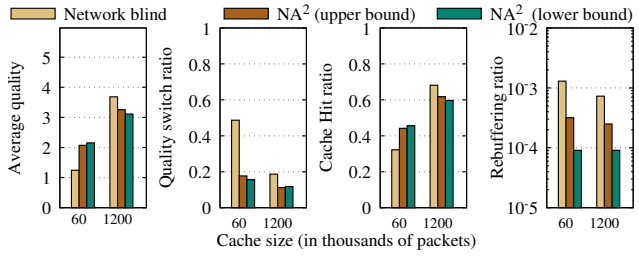

Figure 4: Evaluation: Comparison of Network blind vs Network Aware AdapTech, for two NA^2 settings: upper bound of average quality $(T_{Low}, T_{High}) = (0.1, 0.35)$ and lower bound of quality switch ratio $(T_{Low}, T_{High}) = (0.1, 0.5)$.

rebuffering probability (the ratio of the number of rebufferings over all experiments over the number of segments downloaded). The gold bars present the results for network-blind AdapTech, the brown ones present the results for NA^2 (upper bound) and the green one presents the results for NA^2 (lower bound).

For both cache sizes , we confirm that NA^2 sizeably (drastically) reduces the quality-switch (rebuffering) ratio. Particularly, when the cache is small (60k packets), NA^2 outperforms a network blind AdapTech, both in average quality and in quality switches. This is due to a better utilisation of the cache (notice the the hit rate increase): with network-blind AdapTech, quality oscillations happens which pollutes the cache. When the cache is small, this pollution is critical because it replaces segments that can be useful for the other clients. Our network-aware approach circumvents this pollution by giving more informations to the client, which can make educated choices for the quality of the next segment and thus preventing quality oscillations.

With a bigger cache (1.2M packets), the pollution is still present, but is less critical because it does not replace useful segments. As a result, the average quality is higher, for both network-aware and network-blind algorithms. The average quality observed for our network-aware approach is slightly lower than compared to network blind AdapTech, which is necessary to prevent cache-induced quality oscillations. Shortly, NA^2 is simple and robust, providing sizeable benefits for DASH QoE.

5 CONCLUSION

In this paper, we propose a simple signal such as the per-quality cache hit-rates from cache, coupled with a per-path throughput estimation, which is effective for (i) avoiding the cache-induced oscillations (reduction of the quality switch ratio), while (ii) maintaining a comparable average quality (increasing worst case quality but necessarily reducing quality for more aggressive clients), and (iii) increasing the cache hit ratio.

While the quantitative results showed in this paper are gathered with a specific network-aware evolution of AdapTech, we argue that network-assistance such as the one we propose is beneficial to all rated-based adaptation logics: part of our future work aims at systematically leveraging such signals in multiple controllers [16].

Additionally, we plan to further extend the class of signals that the network can export. Particularly, another appealing signal is binary feedback piggybacked from the cache to assert whether the *next* segment in the same quality is cached. On one hand, such feedback would require additional overhead at the cache (extra lookup for content that has not been requested yet) and would also need a refined timing: if the feedback is too early, the segment could be evicted from the cache, and if the feedback is too late, it will reach the client after his decision. On the other hand, such feedback would help further to increase the user QoE.

ACKNOWLEDGEMENT

Omitted due to double blind submission

REFERENCES

[1] –hidden due to double blind review–.
[2] Cisco Visual Networking Index: Forecast and Methodology, 2016-2021, 2017.
[3] S. Akhshabi, S. Narayanaswamy, et al. An experimental evaluation of rate-adaptive video players over http. *Signal Processing: Image Communication*, 27(4):271, 2012.
[4] A. Bentaleb, A. Begen, et al. Sdndash: Improving qoe of http adaptive streaming using software defined networking. In *ACM Multimedia (MM'16)*. 2016.
[5] D. Bhat, A. Rizk, et al. Network assisted content distribution for adaptive bitrate video streaming. In *ACM on Multimedia Systems (MMSys'17)*. 2017.
[6] Bouten, N. et al. In-network quality optimization for adaptive video streaming services. *IEEE Transactions on Multimedia*, 16(8):2281, 2014.
[7] Cofano, G. et al. Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming. In *ACM Multimedia Systems (MMSys)*. 2016.
[8] S. D'Aronco, L. Toni, et al. Price-Based Controller for Utility-Aware HTTP Adaptive Streaming. *IEEE MultiMedia*, 24(2):20, 2017.
[9] F. Dobrian, A. Awan, et al. Understanding the impact of video quality on user engagement. *Commun. ACM*, 56(3):91, 2013.
[10] Essaili, A.E. et al. QoE-based traffic and resource management for adaptive HTTP video delivery in LTE. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(6):988, 2015.
[11] A. Ganjam, F. Siddiqui, et al. C3: Internet-scale control plane for video quality optimization. In *USENIX NSDI*. 2015.
[12] Georgopoulos,P. et al. Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming. In *ACM SIGCOMM, FhMN Workshop*. 2013.
[13] R. Grandl, K. Su, et al. On the interaction of adaptive video streaming with content-centric networking. In *International Packet Video Workshop (PV)*. 2013.
[14] T. Y. e. a. Huang. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc. of ACM SIGCOMM*, pages 187–198. 2014.
[15] R. Jmal, G. Simon, et al. Network-assisted strategy for DASH over CCN. In *IEEE International Conference on Multimedia and Expo (ICME)*. 2017.
[16] T. Karagkioules, C. Concolato, et al. A comparative case study of http adaptive streaming algorithms in mobile networks. In *NOSSDAV*, pages 1–6. ACM, 2017.
[17] J. Kleinrouweler, S. Cabrero, et al. Delivering stable high-quality video: an SDN architecture with DASH assisting network elements. In *ACM Multimedia Systems (MMSys)*. 2016.
[18] D. Lee, C. Dovrolis, et al. Caching in http adaptive streaming: Friend or foe? In *ACM NOSSDAV*. 2014.
[19] W. Li, S. M. A. Oteafy, et al. Dynamic adaptive streaming over popularity-driven caching in information-centric networks. In *IEEE ICC*. 2015.
[20] Mok, R.K.P. et al. QDASH: a QoE-aware DASH system. In *ACM Multimedia Systems (MMSys)*. 2012.
[21] M. K. Mukerjee, D. Naylor, et al. Practical, real-time centralized control for cdn-based live video delivery. In *ACM SIGCOMM*. 2015.
[22] H. Nam, K. H. Kim, et al. Towards QoE-aware video streaming using SDN. In *IEEE GLOBECOM*. 2014.
[23] Posch, D. et al. Using In-Network Adaptation to Tackle Inefficiencies Caused by DASH in Information-Centric Networks. In *Proc. of ACM VideoNext Workshop*. 2014.
[24] B. Rainer, D. Posch, et al. Investigating the Performance of Pull-based Dynamic Adaptive Streaming in NDN. *Journal on Selected Areas in Communications*, 34(8):2130, 2016.
[25] S. Schwarzmann, T. Zinner, et al. Towards a framework for comparing application-network interaction mechanisms. In *Proc. of IEEE ITC 28*. 2016.
[26] The Linux Foundation. Fast Data project (fd.io) Community ICN (CICN). https://wiki.fd.io/view/Cicn, 2017.
[27] E. Thomas, T. Deventer, M.O. ad van Stockhammer, et al. Enhancing mpeg dash performance via server and network assistance. In *Proceedings of IET and IBC*, pages 48–53. IET, 2015.