Mehrabidavoodabadi, Abbas; Siekkinen, Matti; Ylä-Jääski, Antti

QoE-traffic Optimization Through Collaborative Edge Caching in Adaptive Mobile Video Streaming

Published in:
IEEE Access

DOI:
10.1109/ACCESS.2018.2870855

Published: 01/09/2018

Document Version
Publisher's PDF, also known as Version of record

Published under the following license:
CC BY

Please cite the original version:
Mehrabidavoodabadi, A., Siekkinen, M., & Ylä-Jääski, A. (2018). QoE-traffic Optimization Through Collaborative Edge Caching in Adaptive Mobile Video Streaming. *IEEE Access*, *6*, 52261-52276.
https://doi.org/10.1109/ACCESS.2018.2870855

Received August 15, 2018, accepted September 7, 2018, date of publication September 18, 2018, date of current version October 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2870855

QoE-Traffic Optimization Through Collaborative Edge Caching in Adaptive Mobile Video Streaming

ABBAS MEHRABI, (Member, IEEE), MATTI SIEKKINEN, (Member, IEEE), AND ANTTI YLÄ-JÄÄSKI, (Member, IEEE)
Department of Computer Science, Aalto University, FI-00076 Espoo, Finland

Corresponding author: Abbas Mehrabi (abbas.mehrabidavoodabadi@aalto.fi)

This work was supported in part by the Academy of Finland under Grant 278207 and Grant 297892, in part by the Tekes - the Finnish Funding Agency for Innovation, and in part by the Nokia Center for Advanced Research.

ABSTRACT Multi-access edge computing has been proposed as a promising approach to localize the access of mobile clients to the network edges, therefore, reducing significantly the traffic congestion on the backhaul network. Due to time-varying wireless channel condition, the video caching at the mobile edges for dynamic adaptive video streaming over HTTP (DASH) needs to be efficiently handled to alleviate the high bandwidth demand on the backhaul network and improve the quality of experience (QoE) of end users. We investigate the impact of collaborative mobile edge caching on joint QoE and backhaul data traffic by proposing the joint QoE-traffic optimization with collaborative edge caching which introduces the BFTR (backhaul/fronthaul traffic ratio) parameter adjustable by the mobile network operator. We then design a self-tuned bitrate selection algorithm with low complexity to solve the optimization problem and further propose an efficient cache replacement strategy called retention-based collaborative caching. Through simulation-based evaluations, we show a noticeable gain in the percentage of cache miss and specify some threshold for BFTR parameter after which the significant reduction in the data traffic with further improvement in average video bitrate is obtained using collaborative caching. Our findings help mobile edge system developers design an efficient collaborative caching mechanism for 5G networks.

INDEX TERMS Collaborative caching, dynamic adaptive video streaming over HTTP (DASH), fairness, integer non-linear programming, multi-access edge computing (MEC), NP-hardness, quality of experience.

I. INTRODUCTION

According to statistics, the majority of Internet traffic is generated by video streaming applications, such as Netflix, YouTube etc. [37]. By 2019, about 70% of the traffic in the Internet is expected to be from video streaming services [15]. In future 5G mobile networks, the low end-to-end communication delay and high available bandwidth enables the mobile clients to watch the videos with high quality [21], [15]. Toward enabling this objective, Multi-access Edge Computing (MEC) has been proposed in conjunction with network virtualization in which the video contents are localized at the network edges within the radio access network (RAN), therefore, reducing the delivery delay and alleviating the traffic congestion on the backhaul network [17].

At the network edge within the RAN, the time varying wireless channel quality and the clients' mobility significantly affects the video quality when multiple clients compete for the shared bandwidth simultaneously. As a solution to improve the quality of experience (QoE) of the viewers, dynamic adaptive video streaming over HTTP (DASH) protocols [38] are implemented on the clients' side which react to the varying network conditions by adapting dynamically to the most suitable video bitrate. Furthermore, with the advent of MEC, video caching and delivery at the network edge provides means for Internet service providers (ISPs) to economize by reducing backhaul and inter-ISP data traffic [19].

In dynamic adaptive video streaming, the video content caching at the network edge is more challenging since for each request from a mobile client, not only the requested chunk but also its specific bitrate must be available in the cache. To improve the cache hit rate, solutions based on joint bitrate adaptation and edge caching have recently emerged. Joint optimization of QoE and backhaul data traffic takes this approach one step further. The key insight is that allowing slight degradation in video quality may sometimes substantially reduce the data traffic on the backhaul network. In other words, by fetching cached chunks with somewhat lower bitrate than that sustainable by the network, we can reduce backhaul traffic by avoiding the download of not (yet) cached higher bitrate versions of the same chunks from the origin server. Further noticeable reduction in the backhaul traffic could be achieved when the neighborhood edge servers collaborate with each other to serve the requested chunks/bitrates in the case of their unavailability in the local caches.

In this paper, we aim to quantify the achievable gain from jointly optimizing the QoE of the clients with the network data traffic in collaborative mobile edge caching environments. We further explore under which circumstances the collaborative edge caching brings noticeable improvements in QoE-traffic compared to non-collaborative strategies. Our simulation results reveal that the network-assisted bitrate adaptation combined with collaborative edge caching indeed helps to improve the QoE of the clients and alleviate significantly the network traffic burden compared to other collaborative edge caching mechanisms.

A. CONTRIBUTIONS

Our main contributions are summarized as follows:

- We explore the potential of collaborative video caching in joint QoE-traffic optimization for edge-assisted DASH in MEC environments.
- We formulate the joint QoE-traffic optimization with collaborative edge caching as an integer non-linear programming (INLP) optimization problem which also introduces the backhaul/fronthaul traffic ratio (BFTR) parameter adjustable by mobile network operator (MNO).
- An online network-assisted bitrate adaptation algorithm with performance guarantee is designed for the optimization problem which can be easily deployed by MNOs. An efficient edge cache replacement heuristic called retention-based collaborative caching (RBCC) is also designed which takes into account not only the retention of the clients toward different video requests but also the inter-collaboration between the edge servers.
- The results of our simulation-based evaluation confirm that the network-assisted bitrate adaptation with collaborative edge caching indeed helps to improve the average bitrate of the clients with significant reduction in the backhaul network traffic.

B. PAPER ORGANIZATION

The remaining parts of the paper are organized as follows: Section II discusses the related works. In Section III, the system design for collaborative edge caching adaptive video streaming is introduced. The QoE and data traffic components are also discussed in this section. The joint QoE-traffic optimization problem is formulated in Section IV and the proposed bitrate allocation algorithm, the cache replacement heuristic and their complexity analysis are presented in Section V. Section VI is devoted to the results of our conducted simulations and finally, Section VII concludes the paper.

II. RELATED WORK

High available bandwidth and low end-to-end communication latency are the primary objectives of future mobile networks. Gupta and Jha [21] discuss in a comprehensive survey the architecture of 5G mobile network, the fundamental requirements and emerging technologies involved. Multi-access edge computing (MEC) proposed by European Telecommunication Standard Institute (ETSI) is an emerging solution to meet these requirements [15], [17]. In MEC, placing the content at the edge within the radio access network (RAN) has been proved to reduce the end-to-end latency and degrade the traffic on the backhaul network [19], [36]. A comprehensive survey on mobile edge computing, its advantages and comparison with the centralized cloud-based architecture are provided in [23] and [27].

In mobile video streaming, the bandwidth demand of the clients is expected to increase dramatically in the next generation of mobile networks [15], [17]. In on-demand video streaming, caching the most popular video contents with large size at the network edge can help to reduce significantly the high contention on the origin server [17]. In contrast to static HTTP-based video streaming, the edge caching is however more challenging in dynamic adaptive video streaming since for each request from a mobile client, not only the requested chunk but also its specific bitrate should be efficiently cached [36]. Zhang *et al.* [35] have proposed cooperative caching among mobile users with the help of edge computing facilities. Lei *et al.* [34] discuss the existing challenges in the practical implementation of mobile edge caching.

Another aspect of mobile edge computing and caching is designing solutions for enhancing the security for the mobile users [29], [30], [33]. Roman *et al.* [29] provide a comprehensive survey on the security challenges in mobile edge and fog computing environments. A jammer can interrupt the radio communication between the mobile users and edge servers and, therefore, prevent the clients from accessing the cache [30], [33]. Xiao *et al.* [30] propose an efficient reinforcement learning algorithm to protect mobile clients from jammer attacks in accessing the edge contents.

During the past years, several research efforts have been proposed for improving the QoE of the mobile clients in DASH video streaming [7], [8], [10], [13]. Seufert *et al.* [5] provides a comprehensive study on DASH quality adaptation and the major factors that both client and network have to take into account. Majority of the works on QoE consider the quality adaptation algorithms merely on the client side which in turn causes the suboptimal bitrate allocation and the under-utilization of the network resources when multiple streaming clients compete on the shared bandwidth. Although some research works have investigated the scalability of DASH strategies [1]–[4], the client-based adaptation heuristics may still lead to unfair bitrate allocation among the clients due to the lack of coordination among them in some situations such as the interleaving of their arrival and departure times. Furthermore, the major factors that directly impact the QoE of the clients have not been considered in these works.

Server and network assisted DASH (SAND-DASH) standard which has been recently published is a progress toward the collaboration between the mobile clients, servers and in-network elements [39]. The preliminary research works on SAND-DASH [9], [11], [12] fail to provide a concrete optimization framework for jointly maximizing the QoE of the clients and the fair allocation of the wireless network resources. Toward this step, Mehrabi *et al.* [18] design an optimization problem for jointly maximizing the QoE, the fair bitrate allocation among the competing clients as well as balancing the utilized resources among multiple edge servers. However, the assumption of the availability of all variations of the video chunks in all edge servers is not realistic in the practical situations when the cache size is constrained.

For addressing the challenges posed on edge-assisted DASH, the joint adaptive video streaming, caching and processing at the edge servers has been recently suggested. Pedersen and Dey [20] propose the joint optimization of adaptive video streaming, backhaul resource allocation, and video content caching at RAN. The proposed adaptive video streaming and caching model in this work assumes that the clients request to the chunks with lower bitrates may be served from the cache by transcoding from the available chunks with higher bitrates at the edges. Further reduction in cache miss can be achieved by the collaboration among the edge servers within a cluster [24], [22], [15]. Tran and Pompili [24] propose Octopus, a cooperative hierarchical caching for cloud radio access networks [24]. In addition to the clients access to the cloud-cache in the case of miss in the local cache, they utilize the potential of inter-communication among the neighborhood edge servers to serve the requested chunks in a collaborative manner. However, their system model does not take into account the clients mobility and it is assumed that the requested bitrates by the clients are initialized in the caches using a random distribution.

None of the above-mentioned papers propose an in-network adaptation solution that can guide the clients toward a fair and optimal bitrate allocation by utilizing their radio access link level information at the network edges. Furthermore, in contrast to the common least recently used (LRU) cache replacement strategy, which has been utilized in some of the prior work, the heuristics based on the statistical information about the clients' retention behavior can be designed for the collaborative edge caching scenarios in order to further improve the cache hit rate.

In scenarios with limited processing capabilities at the edges and high bandwidth capacity on the backhaul network, the encoding of the video chunks can be performed at the origin server while the caching and clients information processing are handled at the edges. With low access delay to the cloud server using ultra-fast communication lines in next generation of mobile networks [26], the joint optimization of QoE and data traffic is a promising solution to address the challenges of edge-assisted on-demand DASH video streaming. According to the best of our knowledge, the impact of collaborative edge caching on joint QoE-traffic optimization for edge-assisted DASH has not been yet addressed. To this end, we propose the joint QoE maximization, fairness among the competing clients and the data traffic (backhaul and fronthaul) minimization in collaborative mobile edge-assisted DASH video streaming scenarios. Using a self-tuning mechanism, we design an online greedy-based algorithm with performance guarantee to solve the joint optimization problem. Inspired by the research study in [32], we further propose an efficient edge cache replacement heuristic which takes into account not only the statistical information about the clients' retention pattern but also the inter-collaboration among the edge servers. Our simulation results show that the network-assisted bitrate adaptation combined with retention-based collaborative edge caching indeed helps to improve the QoE of the clients while obtaining the significant reduction in the network backhaul traffic.

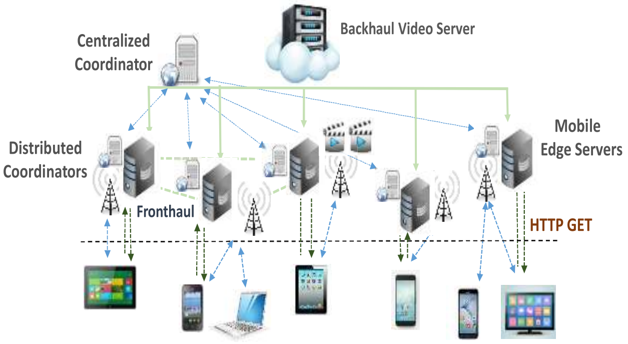


FIGURE 1. Collaborative caching at the network edges.

III. MOBILE EDGE ASSISTED DASH WITH COLLABORATIVE CACHING

A. SYSTEM OVERVIEW

Fig. 1 illustrates a schematic view of mobile edge assisted adaptive video streaming with collaborative caching. Associated with the edge servers, the cellular base stations allocate the available downlink radio resource blocks among the set of local clients based on the proportional fair (PF) policy. The clients radio access link level information and the video content caching are handled at the network edges. The set of neighborhood edge servers form the edge clusters in which they collaborate with each other to serve the request of mobile clients. We design in-network bitrate adaptation solution such that with the instantaneous radio link level information of the clients, the coordinators first resolve the clients to server mapping based on per chunk signal-to-noise ratio (SNR) values and then solve the joint QoE-traffic optimization problem according to the weighting adjustment by MNO. The most sustainable bitrates are allocated to the clients at each time slot based on the results of the optimization problem.

The decision on bitrate allocation to each chunk is made taking into account the existence of chunks in the caches in a down-to-up hierarchical manner. In other words, the chunk/bitrate of the client's requested video is searched in the cache of first the local and then the neighborhood edge servers in the case that it does not exist in the local cache. Upon the failure to find the chunk/bitrate in the edges, then the possibility of transferring from the origin server is investigated. The actual bitrate allocation is performed based on the maximization of joint QoE-traffic utility objective. After the bitrate allocation, the possibility of caching among multiple served chunks/bitrates for future clients' access is then investigated.

It should be noted that the edge-assisted bitrate adaptation is independent from the collaborative edge caching. In the former, clients and edge servers coordinate with each other in order to achieve optimal and fair bitrate allocation among the clients. In the latter, the edge servers collaborate with each other to improve the overall cache hit rate. In this work, we do not address the mechanisms required to forward clients' requests to the chosen edge or origin server. We also do not address security aspects of collaborative edge caching.

B. SYSTEM NOTATIONS

We consider the scheduling of S number of DASH mobile clients during $|T|$ time slots with the duration of each slot of Δt seconds. The contents of multiple videos are initially placed at the origin server such that each video is divided into consecutive chunks with fixed size of each chunk equal to C seconds and available in multiple discrete bitrates represented by set R. K edge servers are deployed in the system and associated with each server, the base station (eNodeB) allocates the downlink resource blocks to the subscribing clients according to PF policy. The available downlink resource blocks at edge server k in time slot t is represented by $W_k^{(t)}$. The arrival and departure times of client i are denoted by respectively A_i and D_i. Also, the media player of each client contains a video buffer with maximum capacity of B_i^{max} and the amount of video data at the buffer of the client at time slot t is represented by $B_i^{(t)}$. Depending on the client mobility, the receivable signal to noise ratio (SNR) of client i from the base station k at time slot t is denoted by $SNR_{ik}^{(t)}$. Furthermore, $r_{ik}^{(t)}$ represents the allocated bitrate to the current chunk of client i assigned to server k at time slot t. The binary variable $a_{ik}^{(t)}$ is defined such that $a_{ik}^{(t)} = 1$ indicates the allocation of client i to edge server k at time slot t.

In our system model, we assume that the server allocation is decided at the beginning of each new chunk and the current server remains unchanged if the client is downloading at the middle of the chunk. At the beginning of the new chunk, the client is first assigned to the base station from where it receives the highest SNR and the edge server associated to that BS. Then, the next video chunk delivered to the client is fetched either from the cache of the local (the edge server associated to the BS) or another edge server in the neighborhood, or from the origin server. Regardless of which server the content is fetched from, it passes through the edge server of the BS the client is associated to and gets transmitted over the air by that BS. In other words, that edge server acts as a proxy to the client.

The fixed cache capacity at each edge server is denoted by Q and the set of available chunks in the cache of edge server k at time slot t is denoted by $M_k^{(t)}$. Furthermore, two binary decision variables $dc_{ik}^{(t)}$ and $de_{ik}^{(t)}$ are also defined such that $dc_{ik}^{(t)} = 1$ indicates that the chunk/bitrate of the video watching by client i assigned to server k at time t is downloaded from the origin server. Similarly, $de_{ik}^{(t)} = 1$ states that client i downloads the chunk/bitrate of its requested video from the other edge servers else than server k at time slot t. It is noted that at each time slot t, the equality $dc_{ik}^{(t)} \cdot de_{ik}^{(t)} = 0$ holds meaning that the client cannot download its requested chunk simultaneously from the origin and the edge servers at the same time slot. The notations BT_i and FT_i (in Mb) denote the overall data traffic on respectively the origin server and the fronthaul network resulted during the whole duration of video streaming of client i. Furthermore, the parameter $BFTR$ (backhaul/fronthaul traffic ratio) is also defined which takes a value between zero and one and is adjustable by MNO. This parameter is included in the system to indicate the significant of data traffic on the fronthaul network compared to the traffic on the backhaul network from MNO point of view. It enables to access the locality of the traffic and, hence, the effectiveness of caching that happens at the edge of the network.

The list of system parameters and their descriptions have been summarized in Table 1.

C. QUALITY OF EXPERIENCE

In conventional video streaming systems, the quality of experience (QoE) of the clients is measured using the traditional subjective metrics such as the mean opinion score (MOS) or the objective metrics such as peak signal-to-noise ratio (PSNR). However, these metrics are not commonly used nowadays by video providers because they are difficult to measure in a large scale operational system, especially MOS which requires explicit participation of the users. As pointed out by several research works in DASH literature, the main factors that affect the QoE in dynamic adaptive video streaming are *video stalling*, *video quality*, *bitrate switching* and initial playback buffering time known as *initial/startup delay*.

1) VIDEO QUALITY

The quality that clients perceive during the streaming session is directly related with the streaming bitrate as the video chunks are streamed with high bitrate, the watching quality increases. We rely on the average bitrate that client i watches the video during the whole time duration of its streaming session which is given by the following equation:

$$AQ_i = \frac{C}{|D_i - A_i|} \sum_{\forall A_i \le t \le D_i} \sum_{1 \le k \le K} a_{ik}^{(t)} \cdot r_{ik}^{(t)} \qquad (1)$$

TABLE 1. System notations and their descriptions.

Notation	Description
C	Constant size of each video chunk (in seconds)
K, S, R	Number of edge servers, DASH clients and the discrete set of available video bitrates, respectively
$\|T\|, \Delta t$	Total number of scheduling time slots and the duration of each slot in seconds
$W_k^{(t)}, M_k^{(t)}$	Available resource blocks at base station k and the set of cached chunks at edge server k in time slot t, respectively
Q	Constant cache size
A_i, D_i	Arrival and departure times of client i
B_i^{max}	Maximum buffer capacity (in Mb) of client i
$B_i^{(t)}$	Buffer level of client i at time slot t
AQ_i	Average video quality for client i
L_i	Initial delay on the client i's buffer
E_i	Accumulated bitrate switching for client i
$SNR_{ik}^{(t)}$, $Thr_{ik}^{(t)}$, $\hat{Thr}_{ik}^{(t)}$	Received SNR, theoretical data throughput and effective throughput by client i from base station k at time slot t
F_i	Fairness of client i
$0 < BFTR < 1$	Backhaul/fronthaul traffic ratio parameter adjustable by mobile network operator (MNO)
$DT_i = BT_i + FT_i$	Overall data traffic (backhaul and fronthaul) caused during the whole duration of video streaming of client i
ρ, ω, γ	Adjustable weighting parameters for average quality, bitrate switching and fairness respectively
α	Weighting coefficient for controlling the importance of QoE and data traffic in the joint optimization
$a_{ik}^{(t)}$	Binary indicator for the allocation of client i to server k at time slot t
$dc_{ik}^{(t)}, de_{ik}^{(t)}$	Binary indication that the current chunk/bitrate of client i at server k in time slot t is downloaded from respectively the origin video server and the neighborhood edge servers
$r_{ik}^{(p)} \in R$	Discrete allocated bitrate to chunk index p of client i at server k

2) BITRATE SWITCHING

High bitrate switching also negatively impacts the satisfaction of the mobile clients. The accumulated bitrate switching (magnitude) of client i during its streaming session from the arrival to departure is obtained as follows:

$$E_i = \sum_{p=2}^{|D_i - A_i|/C} \sum_{\forall 1 \le k \le K} (a_{ik}^{(A_i+(p-1)\cdot C)} \cdot r_{ik}^{(p)} - a_{ik}^{(A_i+(p-2)\cdot C)} \cdot r_{ik}^{(p-1)} \qquad (2)$$

3) INITIAL BUFFER DELAY

Initial buffer delay denoted by L refers to the time duration from the arrival time of the client until the time that the data in the playback buffer reaches the maximum capacity. As a trade-off, longer initial buffer delay helps to reduce significantly the video stalling events during the streaming session. Although long delay slightly impacts QoE, most of the clients however tend to tolerate longer initial delay in order to experience smoother video without interruption [6]. Since the impact of delay on the QoE is not significant as the impact of video bitrate and switching, we ignore the buffer delay in our analytical model for the sake of simplicity.

4) STALLING RATIO

Video stall event refers to the time when the client's buffer gets empty (playback freezes) due to streaming the video on the player with high bitrates under low achievable throughput. To quantitatively express the stalling in the optimization problem, we need to derive first the relation for the buffer filling level of the client at each time slot.

$$B_i^{(t)} = \begin{cases} B_i^{(t-1)} + \hat{Thr}_{ik}^{(t)} \cdot \Delta t, & A_i \le t \le A_i + L_i \\ B_i^{(t-1)} + (\hat{Thr}_{ik}^{(t)} - r_{ik}^{(p)}) \cdot \Delta t, & A_i + L_i < t \le D_i \end{cases} \qquad (3)$$

where Δt is the fixed duration of each time slot and $\hat{Thr}_{ik}^{(t)}$ represents the effective (achievable) throughput by client i allocated to server k at time slot t. It is noted that in the startup phase which lasts for the time duration of initial buffer delay, there is no video streaming on the client's player. At a given time slot, the effective throughput of each client is obtained based on its theoretical throughput, Thr, and the number of other simultaneous clients at that time slot.

In order to ensure that no stalling event happens during the whole time duration of video streaming, the following constraint should be satisfied at each time slot:

$$0 \le B_i^{(t)} \le B_i^{max}, \quad \forall A_i \le t \le D_i \qquad (4)$$

5) FAIRNESS

As mentioned earlier in the system model, the downlink radio resources are allocated to the clients according to the PF policy at the base station. The client-side adaptation heuristics allocate the most sustainable bitrates to the clients based on their share of throughput (wireless channel quality). However, in some situations, the client-based solutions may not fairly allocate the bitrates resulting in unfairness among the competing clients. To avoid unfair situations, for each current client, we strive to allocate the best sustainable bitrate which has the least difference with the average of bitrates allocated to the other simultaneous active clients at the same server. More precisely, the objective of fair bitrate allocation is to minimize the overall deviation of the allocated bitrates to each client i from the average during its whole streaming session given by the following summation:

$$F_i = \sum_{t=A_i}^{D_i} \sum_{1 \le k \le K} a_{ik}^{(t)} \cdot |r_{ik}^{(t)} - \bar{r}^{(t)}| \qquad (5)$$

where $\bar{r}^{(t)}$ is the average bitrate of other simultaneous clients at time slot t. The minimization of (5) should also satisfy the instantaneous available resource blocks at the base station.

D. BACKHAUL AND FRONTHAUL DATA TRAFFIC

As mentioned in the system model, the allocation of bitrates are decided based on the existence of the chunk/bitrate of the requested video in down-to-up hierarchical caches. If the requested chunk is not available in the local cache, it is then searched in the cache of the neighborhood edges taking into account the data traffic that it will add to the fronthaul network within the edge cluster. If it is not also found in the neighborhoods, then, transferring the chunk from the cloud server with the cost of adding to the backhaul traffic is investigated. Considering two decision variables $dc_{ik}^{(t)}$ and $de_{ik}^{(t)}$, the overall backhaul and fronthaul data traffic caused during the video streaming of client i are therefore obtained using the following summations:

$$BT_i = \sum_{t=A_i}^{D_i} \sum_{1 \le k \le K} a_{ik}^{(t)} \cdot dc_{ik}^{(t)} \cdot \Delta t \cdot r_{ik}^{(t)} \tag{6}$$

$$FT_i = \sum_{t=A_i}^{D_i} \sum_{1 \le k \le K} a_{ik}^{(t)} \cdot de_{ik}^{(t)} \cdot \Delta t \cdot BFTR \cdot r_{ik}^{(t)} \tag{7}$$

And, the overall data traffic resulted from the video streaming of client i is given by $DT_i = BT_i + FT_i$. System model assumes that there is no traffic incurred when the requested chunks are retrieved from the local caches. It is noted that there is no basically a trade-off between the backhaul and fronthaul data traffics since reducing the traffic on the backhaul network does not necessarily mean increasing the data traffic on the fronthaul network. In fact, efficient optimal heuristics can be designed for updating the cache contents which minimize both the backhaul and fronthaul data traffic.

IV. JOINT QoE-TRAFFIC OPTIMIZATION

In this section, we formulate the problem of jointly maximizing the QoE, fairness and minimizing the overall data traffic for each individual client. The optimization problem is formulated considering the discrete time slotted DASH scheduling with fixed duration Δt for each time slot. Three adjustable weighting parameters ρ, ω, γ are defined to control the importance of each component video quality, bitrate switching and fairness in QoE term. The weighting parameter α is defined for adjusting the balance between QoE of the clients and the video data traffic (backhaul and fronthaul). With the relations (1), (2), (4), (5), (6) and (7), the joint QoE-traffic optimization in collaborative edge caching is defined as the following integer non-linear programming (INLP) optimization problem:

$$\underset{dc,de,r}{Maximize}\ U_i = \alpha(\rho AQ_i - \omega E_i - \gamma F_i)\Delta t - (1-\alpha)DT_i \tag{8}$$

Subject to:

$$\sum_{j \in S} a_{jk}^{(t)} \cdot \lceil \frac{r_{jk}^{(\lceil t/C \rceil)}}{Thr_{jk}^{(t)}} \rceil \le W_k^{(t)}, \quad \forall 1 \le k \le K, \ \le t \le |T| \tag{9}$$

$$0 < B_i^{(t)} \le B_i^{max}, \quad \forall A_i \le t \le D_i \tag{10}$$

$$\begin{aligned} dc_{ik}^{(t)} &= \mathbb{I}((v_i, \lceil (t-A_i)/C \rceil, r_{ik}^{(t)}) \notin M_{k'}^{(t)}, \quad \forall 1 \le k' \le K), \\ &\forall 1 \le k \le K, \quad A_i \le t \le D_i \end{aligned} \tag{11}$$

$$\begin{aligned} de_{ik}^{(t)} &= \mathbb{I}((v_i, \lceil (t-A_i)/C \rceil, r_{ik}^{(t)}) \notin M_k^{(t)} \\ &\wedge \exists \quad 1 \le k' \le K \ni (v_i, \lceil (t-A_i)/C \rceil, r_{ik}^{(t)}) \in M_{k'}^{(t)}), \\ &\forall 1 \le k \le K, \quad A_i \le t \le D_i \end{aligned} \tag{12}$$

$$a_{ik}^{(t)} \cdot dc_{ik}^{(t)} \cdot de_{ik}^{(t)} = 0, \quad \forall 1 \le k \le K, \ A_i \le t \le D_i \tag{13}$$

$$a_{ik}^{(t)} = \begin{cases} a_{ik}^{(t-1)}, & t \ mod \ C \ne 1 \\ 1, & t \ mod \ C = 1 \wedge k = arg \ max\{SNR_{ik}^{(t)}\} \\ 0, & \text{Otherwise} \end{cases} \tag{14}$$

$$\begin{aligned} &r_{ik}^{(p)} \in R, \quad dc_{ik}^{(t)}, de_{ik}^{(t)} \in \{0, 1\}, \\ &\forall 1 \le k \le K, \quad A_i \le t \le D_i, \ 1 \le p \le (D_i - A_i)/C \end{aligned} \tag{15}$$

In the above optimization problem, variables $r_{ik}^{(t)}$ and $dc_{ik}^{(t)}$ and $de_{ik}^{(t)}$ are the integer and binary decision variables, respectively, and, the values of other parameters are known in advance. Constraint (9) ensures that at each time slot, the available resource blocks at the BS are allocated to the requested clients according to their wireless link quality. Inequality (10) ensures avoiding the stalling at each time slot as mentioned in the previous section and constraint (11) states that the client downloads the chunk/bitrate of its video from the origin server at time slot t if it is not available in the cache of any local or neighborhood edge servers. Constraint (12) states that the client downloads its chunk/bitrate from a neighborhood edge server within the same cluster if its request does not exist in the local cache. Constraint (13) ensures that the client cannot download the requested chunk/bitrate of its video from the origin and the edge server simultaneously at the same time slot. Constraint (14) determines the edge server to which the client is mapped at each time slot and finally, the relations in (15) specify the range of decision variables.

V. ONLINE SCHEDULING ALGORITHM

The existence of integer decision variables makes the optimization problem (8)-(15) NP-hard and, therefore, the exhaustive search strategies do not scale. Furthermore, the problem (8-(15) is formulated in an offline manner assuming that all the information about the clients and edge servers are available in advance, which is not practically feasible. We design an online greedy-based algorithm with low computational complexity which also requires the minimum need for parameter tuning making it suitable for practical deployment. The proposed algorithm utilizes a self-tuned bitrate selection procedure which allocates the most suitable bitrates to each client and dynamically adjusts the weighting parameters in QoE term of the objective function. However, the flexibility of adjusting the weighting of QoE and data traffic along with the setting of BFTR parameter are left to the MNO itself depending on its desired operational point. Pseudo-code of the algorithm called Collaborative Cache-aware Bitrate Allocation Algorithm (CCBAA) is shown in Algorithm 1.

Algorithm 1 Collaborative Cache-Aware Bitrate Allocation Algorithm (CCBAA) (Run by the Coordinators)

```
1:  Input: $|T|, K, R$ : Number of scheduling time slots, number of edge servers, set of available discrete bitrates on the origin server.
2:  Output: Binary allocation $dc_{ik}^{(t)}$, $de_{ik}^{(t)}$ and integer bitrate allocation $r_{ik}^{(t)}$ for each client $i$, edge server $1 \le k \le K$ and time slot $1 \le t \le |T|$, $totalUtility$, $totalTraffic$
3:  $M_k^{(t)} = \emptyset \quad \forall 1 \le k \le K, \forall 1 \le t \le |T|$;
4:  for each time slot $1 \le t \le |T|$ do
5:      for each client $i$ such that $A_i \le t \le D_i$ do
6:          Allocate client $i$ to server $1 \le k \le K$
              according to (14)
7:          if $t = A_i$ then
8:              Initialize $BufferStatus$, $DT_i$ and $L_i$
9:          if $(t - A_i)$ mod $C \ne 1$ then
10:             Allocate client $i$ to the same server and
                with the same bitrate as with time slot $t - 1$;
11:             Update $B_i^{(t)}$, $DT_i$;
12:             if $BufferStatus = False$ And
                      $B_i^{(t)} = B_i^{max}$ then
13:                   $BuffetrStatus = True$; $L_i = t - A_i$;
14:         if $(t - A_i)$ mod $C = 1$ then
15:             Call Subroutine Self-tuned Bitrate Selection;
16:         if $t = D_i$ then
17:             $totalUtility = totalUtility + U_i$
18:             $totalTraffic = totalTraffic + DT_i$
19: Return $totalUtility$,$totalTraffic$;
```

A. COLLABORATIVE CACHE-AWARE BITRATE ALLOCATION ALGORITHM

In discrete time slotted scheduling, the CCBAA algorithm first resolves the clients to servers mapping at each time slot using the relation (14) (line 6). At each time slot, a client is mapped to the server (base station) with maximum receivable downlink SNR value if the client is about to download a new chunk and it remains allocated to the same server as in the previous time slot if a chunk download is in progress. Buffer status of the client is also initialized if the client arrives to the system at the current time slot (lines 7-8). Then, the same bitrate as the bitrate of the previous time slot is allocated if the client is currently downloading a chunk. Otherwise, if a new chunk download is about to start, the self-tuned bitrate selection procedure (Subroutine 1) is called to determine the best possible bitrate for the new chunk (lines 9-15).

1) SELF-TUNED BITRATE SELECTION

As part of the algorithm, the self-tuned bitrate selection procedure is executed if the client is about to download the new chunk and hence, the most suitable bitrate should be decided for the new chunk.

As demonstrated in Subroutine 1, it first chooses the highest available bitrate if the client is downloading the first chunk of the video (lines 1-3). For the consecutive chunks, the procedure chooses the largest sustainable bitrate (in decreasing order of set R) that results in fewer bitrate switches and higher fairness than observed with the previous chunk. In other words, the chosen bitrate for the current chunk should result in a switching less than some threshold δ_S and a fairness value greater than or equal to the given threshold δ_F. Switching threshold δ_S is derived (line 4) knowing the fact that the highest bitrate switchings between the chunks happen when the bitrates are allocated merely based on the buffer status of the client [18] and fairness threshold δ_F is given at the deployment phase.

Among those bitrates that satisfy the base station resource allocation constraint (9) and both switching and fairness thresholds (lines 5-7), the one which maximizes the utility value (8) is chosen as the bitrate for the current chunk of the client. Note that for each bitrate, the evaluation of utility objective (8) is based on the availability of the chunk/bitrate in down-to-top hierarchical caches as well as the weighting parameters in the objective function (lines 8-17). If there is no available bitrate in set R which satisfies both thresholds, the availability of those bitrates which satisfy the switching threshold δ_S is investigated. Among those eligible bitrates, the one which maximizes the utility objective is selected as the bitrate for the current chunk (lines 18-22). Finally, if no such bitrate is available, the best sustainable bitrate that maximizes the utility value is chosen as the bitrate for the current chunk (lines 23-26). In this way, the bitrates are allocated to the client such that they satisfy the switching and fairness thresholds while maximizing the client utility.

Afterwards, the values of the weighting parameters in QoE term are automatically computed at the current time slot (line 27). More precisely, the weight of the video quality (ρ) is determined based on how far is the selected bitrate from the highest available one in set R. The switching weight (ω) is determined based on how far is the selected bitrate from the one which results in no switching with the bitrate of the previous chunk. Similarly, the fairness weight (γ) is determined based on the distance between the selected bitrate with the average bitrate of other simultaneous clients. The utility value of the client, its buffer status and the associated backhaul/fronthaul data traffic are also accordingly updated (lines 28-35). After the bitrates are allocated to the chunks, the cache replacement heuristic is executed by the edge servers in the current time slot if some of the allocated chunks/bitrates are downloaded from either the origin or the neighborhood edge servers.

It is noteworthy to mention that although the cache replacement heuristic is executed by the edge servers, the chunk eviction at the edge cache implicitly impacts the performance of CCBAA algorithm in terms of resulting QoE-traffic as we show later in simulation results.

Subroutine 1 Self-tuned Bitrate Selection

```
1:  if $t - A_i \leq C$ then
2:      Allocate the highest available bitrate;
3:      Update $BufferStatus$, $B_i^{(t)}$, $DT_i$
4:  Compute $estThr$ and threshold $\delta_S$;   $maxUtility = -\infty$;
5:  for each bitrate $r \in R$ in decreasing order do
6:      if allocation of $r$ satisfy (9) AND
          $r \leq max(estThr, \hat{Thr}_{ik}^{(t)}, B_i^{(t)})$ then
7:          if $|r - r_{ik}^{(t-1)}| \leq \delta_S$ AND
              $1 - |r - \bar{r}|/(R_{max} - R_{min}) >= \delta_F$ then
8:              $Data = 0$;
9:              if $(v_i, \lceil\frac{t-A_i}{C}\rceil, r) \notin M_{k'}^{(t)}, \forall 1 \leq k' \leq K$ then
10:                 $Data = \Delta t \cdot r$;
11:             if $(v_i, \lceil\frac{t-A_i}{C}\rceil, r) \notin M_k^{(t)}$ AND
12:               $\exists 1 \leq k' \leq K \ni (v_i, \lceil\frac{t-A_i}{C}\rceil, r) \in M_{k'}^{(t)}$
13:                 then $Data = BFTR \cdot \Delta t \cdot r$;
14:             Compute weightings $\rho, \omega$ and $\gamma$;
15:             $QE = \rho r - \omega|r - r_{ik}^{(t-1)}| - \gamma|r - \bar{r}|$;
16:             if $\beta QE - (1 - \beta)Data > maxUtility$ then
17:               $maxUtility = \beta QE-(1-\beta)Data; r_{ik}^{(t)} = r$;
18: if $r_{ik}^{(t)} = 0$ then
19:     for each bitrate $r \in R$ in decreasing order do
20:         if allocation of $r$ satisfy (9) AND
              $r \leq max(estThr, \hat{Thr}_{ik}^{(t)}, B_i^{(t)})$ then
21:             if $|r - r_{ik}^{(t-1)}| \leq \delta_S$ then   $Data = 0$;
22:             Perform the same operations as in lines (9)-
    (17);
23: if $r_{ik}^{(t)} = 0$ then
24:     for each bitrate $r \in R$ in decreasing order do
25:         if allocation of $r$ satisfy (9) AND
              $r \leq max(estThr, \hat{Thr}_{ik}^{(t)}, B_i^{(t)})$ then   $Data = 0$;
26:             Perform the same operations as in lines (9)-
    (17);
27: Update weighting parameters $\rho, \omega\ \gamma$ at time slot $t$;
28: Compute $AQ_i, E_i, F_i$ and $U_i$ up to time slot $t$ accord-
    ing to respectively (1), (2), (5) and (8);   Update $B_i^{(t)}$
29: if $B_i^{(t)} = B_i^{max}$ AND $BufferStatus = False$ then
30:     $BufferStatus = True$;   $L_i = t - A_i$;
31: if $(v_i, \lceil(t - A_i)/C\rceil, r_{ik}^{(t)}) \notin M_{k'}^{(t)}, \forall 1 \leq k' \leq K$ then
32:     $dc_{ik}^{(t)} = 1$;   $BT_i = BT_i + \Delta t \cdot r_{ik}^{(t)}$;
33: if $(v_i, \lceil(t - A_i)/C\rceil, r_{ik}^{(t)}) \notin M_k^{(t)}$ AND
      $\exists 1 \leq k' \leq K \ni (v_i, \lceil(t - A_i)/C\rceil, r_{ik}^{(t)}) \in M_{k'}^{(t)}$
    then
34:             $de_{ik}^{(t)} = 1$;   $FT_i = FT_i + \Delta t \cdot r_{ik}^{(t)}$;
35: $DT_i = BT_i + BFTR \cdot FT_i$;
36: Return $U_i$,$DT_i$;
```

B. RETENTION-BASED COLLABORATIVE CACHING (RBCC) HEURISTIC

Varying wireless link quality and the bitrates request during different time periods makes the proactive cache updating more challenging [25]. Since the future arrival/departure or the requested chunks by the clients are not known in advance, we design a probabilistic collaborative heuristic that relies on two separate sources of information: 1) Video viewing statistics 2) current clients' bitrate allocation history. To efficiently utilize the limited cache size, the retention of the clients helps to have some prior knowledge on their viewing behavior. The clients' bitrate history is also used to prioritize between different representations of specific segments. Each time there is need for cache replacement, the proposed cache updating strategy in this work computes a probability value when not only the local clients but also the clients associated to the neighborhood edge servers request a specific video chunk using these two sources of information. The pseudo-code of the proposed cache replacement heuristic called retention-based collaborative caching (RBCC) is in Subroutine 2.

We do not address the challenge of efficient computing of per-video retention but instead expect it to be handled by the video service provider who has access to all the viewing statistics (e.g., YouTube provides *audience retention*). We note that the accuracy of RBCC heuristic is dependent on how well the retention really reflects user behavior, which in turn is affected by the number of samples. Hence, the larger number of views a computed retention is based on, the more accurate predictor of future viewer's behavior we can expect it to be. Some results on the effect of number of views with such a predictor, although for a slightly different problem, are presented in [32]. The authors conclude that performance is fairly good already after 50 views and after 100 views the performance improvements are marginal.

Once the cache update procedure is executed at each edge server, the heuristic computes *weight* and *value* for each set of chunk/bitrate of different requested videos at the current time slot. The weight (occupied space in the cache) of video chunk is equal to the multiplication of chunk size and its allocated bitrate. The value of a chunk is a unitless quantity describing the probability that it will be requested in the future. This value is computed considering all those clients streaming from either the same edge server (lines 4-10) or from other edge servers (lines 11-17) and are currently downloading earlier chunks of the same video. In other words, these clients will request the chunk in question, or another one with same index but different bitrate, unless they abandon viewing the video before that point of time. The calculation needs to consider both the index of chunk as well as its bitrate: For each of those clients, the heuristic first computes the likelihood that client will be still active in its streaming session when it reaches the point of video corresponding to the chunk whose value is being computed. Second, it measures how frequently the bitrate of that chunk has been accessed by each of these other clients during the past.

Mathematically speaking, consider updating the cache contents at edge server $1 \leq k \leq K$. For every client i allocated to server k at time slot t ($a_{ik}^{(t)} = 1$), to compute

Subroutine 2 Retention-Based Collaborative Caching (RBCC) Heuristic (Run by the Edge Server)

```
for each edge server $1 \leq k \leq K$ do
    if $dc_{ik}^{(t)} = 1$ or $de_{ik}^{(t)} = 1$, for at least one client $i : a_{ik}^{(t)} = 1$ then
        Set $V, I, W, P =$ Lists of respectively the videos, chunk indexes, the weights and the updated values of the existing chunks in cache;
        for each each client $i$ such that $a_{ik}^{(t)} = 1$ do
            if $v_i \notin V$ OR $\lceil (t - A_i)/C \rceil \notin I$
                OR $r_{ik}^{(t)} \cdot C \notin W$ then
                Create set $S_L$ from (16);
                Compute $P_{Lreach}(j, i, k, t)$ and
                    $P_{Lacc}(j, i, k, t)$
                from (19) and (20) for each client $j \in S_L$;
                Compute $P_{Lcache}(i, k, t)$ value for
                    client $i$
                according to relation (17);
                Append video $v_i$, chunk index
                $\lceil (t - A_i)/C \rceil$ and its weight $C \cdot r_{ik}^{(t)}$
                to lists respectively $V, I$ and $W$;
                Append $P_{Lcache}(i, k, t)$ value to $P$
        for each each client $i'$ such that $a_{ik'}^{(t)} = 1 (k' \neq k)$
    do
            if ($v_{i'} \notin V$ OR $\lceil (t - A_{i'})/C \rceil \notin I$
                OR $r_{i'k'}^{(t)} \cdot C \notin W$) AND
            $(v_{i'}, \lceil (t-A_{i'})/C \rceil, r_{i'k'}^{(t)}) \notin M_s^{(t)})(\forall s \neq K')$ then
               Create set $S_N$ from (22);
               Compute $P_{Nreach}(j, i', k', t)$ and
               $P_{Nacc}(j, i', k', t)$ for each client $j \in S_N$;
               Compute $P_{Ncache}(i', k', t)$ value
               client $i'$ according to relation (23);
               Append video $v_{i'}$, chunk index
               $\lceil (t - A_{i'})/C \rceil$ and its weight $C \cdot r_{i'k'}^{(t)}$
               to lists respectively $V, I$ and $W$;
               Append $P_{Ncache}(i', k', t)$ value to $P$
        Sort list $P$ and accordingly lists $V, I, W$ in
              decreasing order of caching values;
        $sum\_of\_weights = 0$;
        for $index = 1$ to $size(P)$ do
            $sum\_of\_weights =$
                        $sum\_of\_weights + W(index)$;
            if $sum\_of\_weights > Q$ then
                Break;
            $M_k^{(t)} = M_k^{(t)} \cup$
                    $(V(index), I(index), R(index))$;
```

the value of its current chunk with index $\lceil \frac{t-A_i}{C} \rceil$, the heuristic first creates the following list of potential clients which are allocated to the same server as client i and streaming the earlier chunks of the same video:

$$S_L = \{j | a_{jk}^{(t)} = 1, v_j = v_i, \lceil \frac{t - A_j}{C} \rceil \leq \lceil \frac{t - A_i}{C} \rceil \} \qquad (16)$$

The likelihood of caching the chunk of client i allocated to server k at time slot t, $P_{Lcache}(i, k, t)$, is then computed using the following union probability:

$$P_{Lcache}(i, k, t) = P((\cup L_{(j,i,k,t)} \quad \forall j \in S_L) \cup L_{(new,i,k,t)}) \qquad (17)$$

where for each client $j \in S_L$ at time slot t, the notation $L_{(j,i,k,t)}$ denotes the event that client j has perceived the bitrate of client i's chunk during the past time slots and will be still active in its session until it reaches the client i's chunk. In order to consider the effect of clients' arrival/departure interleaving, we also take into account the arrival of a new client in the caching probability computation. The similar event $L_{(new,i,k,t)}$ is also considered for the new arrival. The probability of the corresponding event for each client $j \in S_L$ is derived using the following multiplication:

$$P(L_{(j,i,k,t)}) = P_{Lreach}(j, i, k, t) \times P_{Lacc}(j, i, k, t) \qquad (18)$$

where the first term, $P_{Lreach}(j, i, k, t)$, is the probability that the client $j \in S_L$ will not abandon the stream before reaching the chunk of client i. The second term, $P_{Lacc}(j, i, k, t)$, states how frequently the bitrate of considered chunk of client i has been accessed by client j during the past time slots.

We note that in the computation of the caching probability, we assume that all clients in set S_L stream the video continuously without interruption, however, our methodology can be easily adapted to the case when some of the clients may jump form one part of the video to the later parts. This adaptation can be achieved by dynamically updating the set of clients in S_L at each time slot and computing the caching value, accordingly. Assuming that the retention function P_{act}, which specifies the probability of a newly arrived client to view different parts of the video, is known and provided by the origin video server, the probability $P_{Lreach}(j, i, k, t)$ is therefore estimated using the following relation:

$$P_{Lreach}(j, i, k, t) \approx 1 - (P_{act}(\lceil \frac{t - A_j}{C} \rceil) - P_{act}(\lceil \frac{t - A_i}{C} \rceil)) \qquad (19)$$

If a given client is currently downloading a chunk of video before the chunk index of client i, it is evident that the closer it is to the chunk index of client i the higher probability that client j will request that other chunk in the future. The second term in (18) is obtained as follows:

$$P_{Lacc}(j, i, k, t) = \frac{\sum_{t'=A_j}^{t} a_{jk}^{(t')} \cdot \mathbb{I}(r_{jk}^{(t')} = r_{ik}^{(t)})}{\sum_{t'=A_j}^{t} a_{jk}^{(t')}} \qquad (20)$$

where the identity function $\mathbb{I}(r_{jk}^{(t')} = r_{ik}^{(t)}) = 1$ if the allocated bitrate to client j on server k at time slot t' is equal to the bitrate of client i at time slot t and $\mathbb{I}(r_{jk}^{(t')} = r_{ik}^{(t)}) = 0$, otherwise.

Similarly, the likelihood that the new arrival reaches the current chunk of client i is given by:

$$P_{Lreach}(new, i, k, t) = 1 - (1 - P_{act}(\lceil (t - A_i)/C \rceil)) \qquad (21)$$

Since a new arrival has not yet started the video streaming, we estimate the probability that the new arrival has accessed the same bitrate as client i during the past time slots with the equal probability $P_{Lacc}(new, i, k, t) \approx 1/|R|$, where R is the set of available bitrates at the origin server.

Now, to consider the effect of collaboration among the edge servers in the cache updating heuristic (the operations in lines 11-17), similarly, for every client i' allocated to edge server $k' \neq k$, the following list of clients is first created:

$$S_N = \{j | a^{(t)}_{jk'(k' \neq k)} = 1, v_j = v_{i'}, \lceil \frac{t - A_j}{C} \rceil \leq \lceil \frac{t - A_{i'}}{C} \rceil \} \tag{22}$$

The likelihood of caching the chunk of client i' allocated to server $k'(k' \neq k)$ at time slot t, $P_{Ncache}(i', k', t)$, is then computed using the following union probability:

$$\begin{aligned} P_{Ncache}(i', k', t) = P((\cup N_{(j,i',k',t)} \quad \forall j \in S_N) \\ \cup N_{(new,i',k',t)}) \end{aligned} \tag{23}$$

Since the existence of any chunk/bitrate in the neighborhood caches helps to reduce the backhaul data traffic by the factor of $BFTR$, therefore, this weighting value is considered in the computation of $P(N_{j,i',k',t})$ and $P(N(new, i', k', t))$ as follows:

$$\begin{aligned} P(N_{(j,i',k',t)}) &= BFTR \times P_{Nreach}(j, i', k', t) \\ &\quad \times P_{Nacc}(j, i', k', t) \end{aligned} \tag{24}$$

$$\begin{aligned} P(N_{(new,i',k',t)}) &= BFTR \times P_{Nreach}(new, i', k', t) \\ &\quad \times P_{Nacc}(new, i', k', t) \end{aligned} \tag{25}$$

Quantities P_{Nreach} and P_{Nacc} for each client $j \in S_N$ and the new arrival are computed in a similar manner as for the clients $j \in S_L$ given in relations (19), (20) and (21).

We assume the events $L_{(j,i,k,t)}$ and $L_{(new,i,k,t)}$ in relation (17) and similarly the events $N_{(j,i',k',t)}$ and $N_{(new,i',k',t)}$ in relation (23) to be independent ignoring the complex interdependencies among the clients for the sake of analytical simplicity. At each edge server, the heuristic then sorts the current chunks in decreasing order of their computed caching values P_{Lcache} and P_{Ncache} (line 18). In the sorted order, the chunks are then inserted into the cache until the sum of the weights of the chunks exceeds the cache capacity (19-24).

It is noteworthy to mention that although the **for** loop in Subroutine 2 is taken over the edge servers, it does not imply that the edge servers update their cache contents independent from each other. In other words, they exchange explicitly the information about the status of neighborhood clients when making the caching decisions.

C. COMPLEXITY ANALYSIS

In this section, we derive the worst case time complexity of algorithm CCBAA and the cache replacement heuristic RBCC, separately. At each time slot and for each client, the client to server mapping part in Algorithm 1 takes worst case time of $O(K)$, where K is the number of edge servers. Obviously, the most computational tasks in Algorithm 1 is taken by running the subroutine Self-tuned Bitrate Selection at each time slot. In the following, we analyze the worst time complexity of Subroutine 1 in one time slot.

The computation of estimated throughput *estThr* during one previous chunk (with size C seconds) results in the worst case complexity of $O(C \cdot S)$, where S is the number of clients in the system. Also, the estimation of switching threshold δ_S demands for $O(|R|)$ time where, $|R|$ is the size of available bitrates at the origin server. For every available bitrate, the assignment of data traffic variable *Data* according to the cache status of the local and the neighborhood edge servers leads to the worst case time complexity of order $O(K)$ and computing each of the weighting parameters in the QoE term (ρ, ω, γ) requires $O(|R|)$ time. Updating the QoE weighting parameters at the current time slot (line 43) again requires $O(|R|)$ time in the worst case. Finally, updating the data traffic values and the corresponding binary variables (lines 47-50) takes the computation time of $O(K)$ in the worst case. Putting the above complexities together results in the following worst case time complexity for running the Subroutine 1 at each time slot:

$$\begin{aligned} T_{Subroutine1} &= O(C \cdot S + |R| + |R|(K + |R|) + |R| + K) \\ &= O(C \cdot S + |R| + (|R| + 1)(K + |R|)) \end{aligned} \tag{26}$$

Now, within $|T|$ time slots and with S number of clients, the execution of Algorithm 1 including the clients to server mapping results in the following worst case time complexity:

$$\begin{aligned} T_{CCBAA} &= O(|T| \cdot S \cdot (K + T_{Subroutine1})) \\ &= O(|T| \cdot S \cdot (K + C \cdot S + |R| \\ &\quad + (|R| + 1)(K + |R|))) \end{aligned} \tag{27}$$

In the following, we derive the worst case complexity of running RBCC heuristic at one edge server. In the worst case, the cache replacement is performed at $|T|$ time slots. At each time slot, the heuristic first updates the caching value of the current existing chunks in the local cache. In the worst case, the maximum number of available chunks in the cache is $\frac{Q}{C \cdot R_{min}}$, where Q is the fixed cache size and R_{min} is the smallest available bitrate in set R. For each of these chunks, the computation of P_{Lcache} takes $O(S(|T| + 1))$ time in the worst case. This is because for S clients, the estimation of P_{Lreach} requires $O(1)$ time when the retention function P_{act} is known in advance, and, the computation of P_{Lacc} takes $O(|T|)$ time in the worst case.

The heuristic then computes the caching value for all the clients allocated to the local and the neighborhood edge servers. In the worst case with overall S clients, this process takes $O(S \cdot S \cdot (|T| + 1))$ time. Also, there are $\frac{Q}{C \cdot R_{min}}$ chunks already in the cache in the worst case and, with S number of other chunks (number of clients), the sorting process takes therefore $O((\frac{Q}{C \cdot R_{min}} + S) log(\frac{Q}{C \cdot R_{min}} + S))$ time. Further, inserting the sorted chunks in the cache demands for the time of order $O(\frac{Q}{C \cdot R_{min}} + S)$. Now, putting the above time complexities all together within $|T|$ time slots results in the following worst case time complexity for RBCC heuristic:

$$T_{RBCC} = O(|T|((\frac{Q}{C \cdot R_{min}} + S)(S(|T|+1) + log(\frac{Q}{C \cdot R_{min}} + S) + 1))) \quad (28)$$

VI. SIMULATION RESULTS

In this section, we present simulation results conducted for evaluating the performance of the proposed system. Our objectives are the following: 1) To compare our bitrate adaptation combined with collaborative edge caching solution against another network-assisted and collaborative edge caching approach in terms of both QoE and data traffic, 2) to evaluate the performance of the proposed RBCC cache replacement heuristic through comparison with other existing methods, and 3) evaluate the impact of BFTR parameter in collaborative caching on both average video bitrate and data traffic with respect to non-collaborative caching.

A. SIMULATION SETUP

The simulations are conducted with a network setup in SimuLTE simulator [40] using the radio access link level traces of 100 UEs (mobile clients) and 10 eNodeBs (base stations). The edge servers associated with these base stations form one single edge cluster i.e. each edge server has all the other servers as its neighborhood. With time slot duration of $\Delta t = 1s$, the total time duration of $300s$ (time slots) is considered for the simulation. All the clients have the constant speed of 8.33 m/s adopted from [16] with linear mobility and their downlink SNR values during different time slots are obtained according to LTE downlink throughput specifications reported in [41]. The chunks of four videos with different popularities are available initially at the origin server in ten different bitrates $[15, 17, 22, 26, 30, 35, 38, 43, 45, 50\ Mbps]$. Each video lasts for $270s$ (time duration of the video) while each chunk is $5s$. Unless explicitly mentioned, the mobile clients arrival time is a random value chosen from the uniform interval $[1, 30s]$ and also, the linear curve is used to represent the clients retention toward different videos.

Constant buffer size of $250Mb$ is considered for all the clients and the cache size at each edge server is fixed at $Q = 2Gb$. With total of $5MHz$ available bandwidth at each time slot, the number of 28 downlink LTE resource blocks are available per slot at each base station. The weighting value $\alpha = 0.5$ is considered in joint QoE-traffic optimization and unless explicitly mentioned, the value of parameter BFTR is set to 1/3. Furthermore, the fairness threshold of $\delta_F = 0.5$ is considered in the simulations. At each simulation part, the average of the results taken over 20 runs of simulation with confidence interval of 95% is presented.

B. QoE-TRAFFIC COMPARISON

1) COMPARISON TO NON-COLLABORATIVE CACHING

We first compare the collaborative and non-collaborative edge caching strategies in terms of QoE metrics and the backhaul/fronthaul data traffic. For each client, the resulting average video bitrate and data traffic are plotted in Fig. 2a to Fig. 2b. Fig. 2a shows that mobile clients perceive higher video bitrate with collaborative caching by exploring not only the local but also the neighborhood caches. Although the collaborative edge caching adds some traffic to the fronthaul network, it also reduces significantly the backhaul traffic (Fig. 2b), as intended. An average improvement of 12% in video bitrate and the reduction of 53% in backhaul data traffic is achieved with collaborative edge caching compared to non-collaborative strategy. *We conclude that with this parameterization, the collaborative edge caching is more effective in reducing backhaul data traffic than increasing video bitrate.*

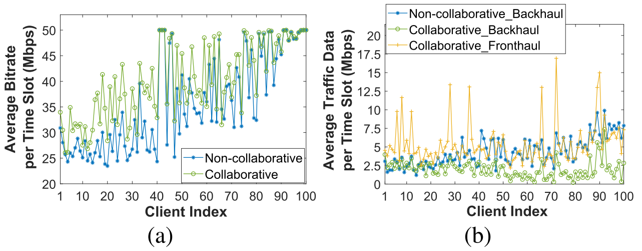


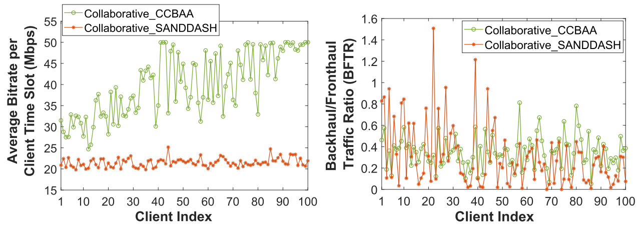
FIGURE 2. Comparison between collaborative and non-collaborative edge caching in terms of (a) average video bitrate (b) average data traffic.

We note that the collaborative edge caching causes slightly more bitrate switching compared to non-collaborative caching. The reason is that in some situations, the clients download the chunks from the neighborhood edges in order to substantially reduce the backhaul traffic although it leads to slightly increased bitrate switching.

2) COMPARISON TO ANOTHER SAND-DASH SOLUTION

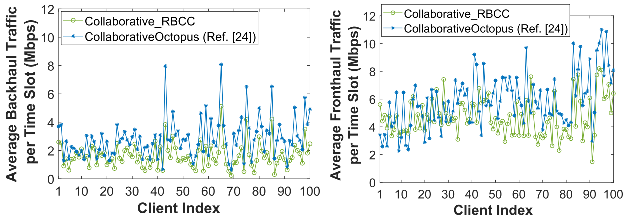
We have also compared the proposed edge-assisted adaptation algorithm CCBAA with another SAND-DASH solution which was adapted from [9]. The network-assisted solution in this work solves a QoE-driven utility maximization problem subject to the limited available bandwidth to determine the allocated bitrates to the set of competing clients. For the purpose of fair comparison, we adapt this solution using the same objective function (8) with $\alpha = 0.5$ and the equal weighting values $\rho = \omega = \gamma = 1/3$ in the QoE term. A simple greedy algorithm is then employed to determine the allocated bitrate to each client at each time slot. Cache updating procedure RBCC is also used for updating the cache contents at the edge servers.

The results of the comparison of CCBAA to the other SAND-DASH solution are shown in Fig. 3. As Fig. 3a shows, our algorithm yields roughly 85% higher average video bitrate than the other algorithm, which is due to the fact that our algorithm uses a self-tuning mechanism which strives to allocate the highest bitrates at each time slot in the best possible way of satisfying the switching and fairness thresholds. However, using our algorithm leads to about 16% (in average)



(a) (b)

FIGURE 3. **Comparison between CCBAA algorithm and another SAND-DASH solution. (a) Average bitrate. (b) BFTR value.**



(a) (b)

FIGURE 4. **Comparison between the proposed collaborative caching and Octopus in terms of backhaul/fronthaul traffic. (a) Average backhaul data. (b) Average fronthaul data.**

increase in the backhaul/fronthaul traffic ratio per client as shown in Fig. 3b. Taking into account the QoE-traffic trade-off, the significant improvement obtained in average video bitrate compared to the smaller increase in BFTR however confirms the superiority of our algorithm with respect to the other network-assisted solution.

It is worth pointing out that our solution can be parameterized such that it generates smaller BFTR values compared to the other SAND-DASH solution, while sacrificing slightly the average video bitrate per client.

We have shown in [18] that the network-assisted bitrate adaptation part of the solution outperforms client-based DASH reference heuristics rate based adaptation (RBA) and buffer based adaptation (BBA). We expect that the collaborative edge-assisted bitrate adaptation and caching solution proposed in this work brings the noticeable improvement in average bitrate with small increase in BFTR value when compared to strategies that combine client-based adaptation heuristics with non-collaborative edge caching.

3) COMPARISON TO ANOTHER COLLABORATIVE CACHING SOLUTION

Next, we compare our solution to the hierarchical collaborative edge caching algorithm called Octopus presented in [24]. Octopus uses a client-based heuristic for the bitrate selection among the clients and the requested chunk/bitrate of the clients can be fetched from the neighborhood caches when it is not available in the local cache. Furthermore, the cache contents at the edge servers are dynamically updated using the LRU policy without considering collaboration between the edge servers in the caching process. To have a fair comparison under the same setup, we adapt the bitrate allocation of Octopus to our network-assisted bitrate adaptation solution while using the same cache replacement heuristic LRU without edge collaboration.

The comparison in terms of backhaul and fronthaul data traffics is shown in Fig. 4a and 4b, respectively. Incorporating RBCC cache replacement heuristic in our solution indeed helps to reduce the backhaul and fronthaul data traffic in average about respectively 42% and 20% compared to Octopus. Considering the QoE-traffic trade-off, our algorithm yields almost the same average video bitrates than Octopus.

C. PERFORMANCE EVALUATION OF RBCC HEURISTIC

Next, we evaluate the performance of the proposed cache replacement heuristic and compare it to other cache updating strategies. The heuristics are compared in term of the percentage of cache miss per client chunk, i.e., the ratio between the number of times that the allocated chunks/bitrates to the all clients are fetched from the origin server and the overall number of allocated chunks/bitrates. We evaluate the following four different cache replacement strategies (the optimal ones are obviously impossible to implement for real systems).

- **Non-collaborative Retention-based Caching (RBC)**: This strategy uses the same retention-based cache updating logic as in this work but without the inter-collaboration between the edge servers in the caching process. Also, the clients fetch the chunk/bitrates from either the local edge or the origin server.
- **Non-collaborative Optimal**: Without collaboration among the edge servers in caching and content fetching, the optimal approach assumes the availability of one time slot ahead clients status at each slot in an offline manner. More precisely, at each time slot, the optimal approach caches the best chunks/bitrates which are immediately requested by the clients in next time slot.
- **Retention-based Collaborative Caching (RBCC)**: The proposed cache replacement strategy in this work which accounts for the possibility of collaborative content fetching and, also takes into account the values of the requested chunks from the neighborhood clients for caching at each local edge server.
- **Collaborative Optimal**: Collaborative optimal approach caches the best possible chunks/bitrates at each time slot assuming the availability of one time slot ahead information about the requests of not only the local but also the neighborhood clients.

1) IMPACT OF ARRIVAL TIME

The results of comparing the heuristics in term of cache miss percentage for different arrival intervals of the clients are plotted in Fig. 5a. For different arrival intervals, the collaborative caching performs better than all the non-collaborative ones, even the offline optimal. On average, the proposed heuristic reduces cache misses by 55% and 38% compared to Non-collaborative_RBC and Non-collaborative_Optimal,

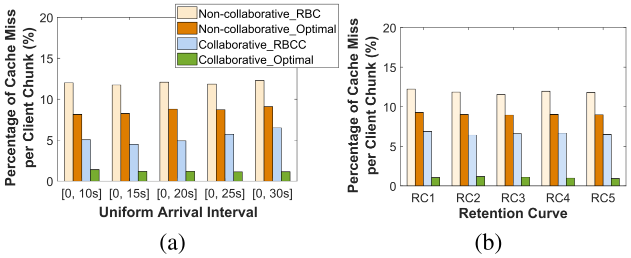


FIGURE 5. Comparison between cache replacement heuristics in term of percentage of cache miss per chunk for different (a) Clients arrival time interval (b) Retention curves.

respectively. Optimal collaborative caching further reduces the cache miss by average about 77% compared to our heuristic. The gap between the optimal collaborative and our heuristic in term of cache miss tends to be bigger than the gap between the Non-collaborative_RBC and Non-Collaborative_Optimal. This in turn reveals the interesting fact that relying on the statistical information for caching is more accurate with respect to the optimal one when there is no collaboration among the edge servers. In fact, the consideration of the requests from neighborhood clients when updating the cache contents at the local edge server may lead to the replacement of some of the chunks which will be most likely requested by the local clients in the future time slots. This in turn causes higher percentage of cache miss under the collaborative cache updating due to under/over estimation of the future bitrates request.

2) ROBUSTNESS

We further study the performance of Collaborative_RBCC heuristic from the robustness point of view. To measure its robustness when the clients show different retentions toward the requested videos, we generate randomly different retention curves (RCs) and then compare the four above-mentioned cache replacement scenarios in term of cache miss. RCs are generated using the polynomial function $P(t) = at^2 + bt + c$ by changing the curvature of the polynomial and determining the corresponding coefficients a, b and c. Here, $P(t)$ specifies the probability that the client leaves its session at time slot t. The results for five different retention curves (RC1-RC5) are shown in Fig. 5b. It is noted that the sharpness of the polynomial gets larger as the curve index increases, for instance, the probability that the clients depart from their streaming session is higher under the retention curve RC5 than the curve RC4.

The results are qualitatively similar to those where arrival time was varied. On average, our heuristic reduces the cache miss by 44% and 27% compared to Non-collaborative_RBC and Non-collaborative_Optimal, respectively. The optimal collaborative strategy further reduces the cache miss by 84% compared to our heuristic.

3) QoE-TRAFFIC BAG PLOT VISUALIZATION

We now visualize the resulting QoE and backhaul data traffic for four different caching strategies using Bag plots [42] with Tukey's half-space median representation. Matlab libraries LIBRA [43] are used to draw the plots shown in Fig. 6a–6b. The bag of the collaborative_RBCC is narrower, which shows that the video streaming of the majority of the clients generates less data traffic on the origin server compared to non-collaborative caching (Fig. 6a). Improvement in average video bitrate with collaborative caching is also confirmed from the plots.

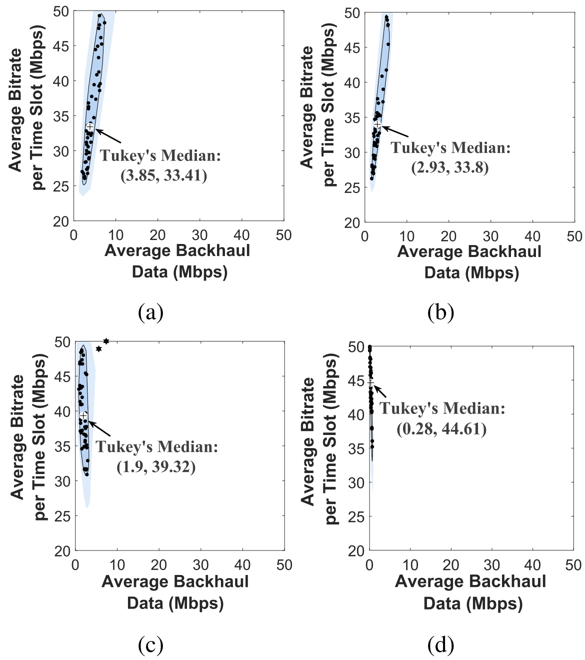


FIGURE 6. Comparison between non-collaborative and collaborative cache replacement in term of QoE/traffic using Bag plot with half space Tukey's median representation. (a) Non-collaborative_RBC. (b) N-collaborative_Optimal. (c) Collaborative_RBCC. (d) Collaborative_Optimal.

D. REMARKS ON THE OPTIMALITY

As mentioned in Subsection V, the optimization problem (8)-(15) belongs to the class of NP-hard problems and therefore, the proposed edge-assisted adaptation algorithm CCBAA is a suboptimal solution for the problem. As our simulation results in [18] show the proposed edge-assisted adaptation solution provides an approximation factor of about 1.2 in terms of average bitrate with respect to the optimal solution under the assumption that every edge server holds all the video chunks/bitrates (unlimited size fully populated edge caches).

Furthermore, the results in Subsection VI-C show that our RBCC cache replacement heuristic updates the cache contents such that the percentage of cache miss is in average about three times worse than the percentage of cache miss when using the optimal collaborative solution. This in turn implies that our algorithm provides an approximation factor of about 3 in term of data traffic minimization.

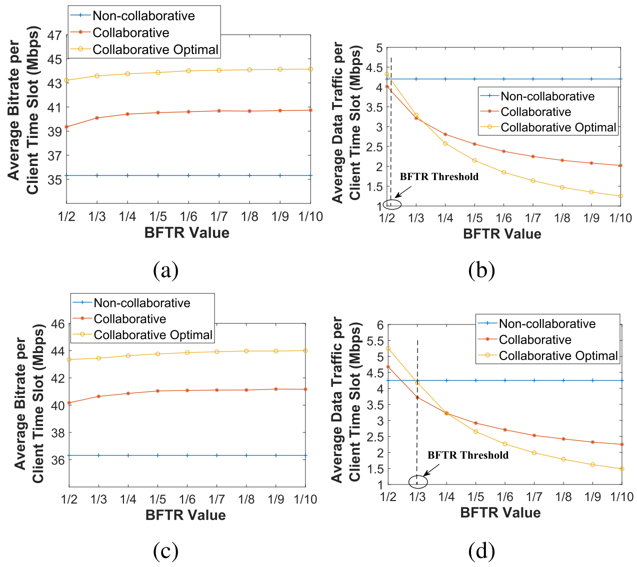


FIGURE 7. Impact of BFTR parameter on (a) Average video bitrate and (b) Average data traffic for different arrival intervals and (c) Average video bitrate and (d) Average data traffic for different retention curves.

E. BFTR THRESHOLD EVALUATION

In this part, we study the impact of the BFTR parameter value on both average video bitrate and the data traffic for different arrival intervals and retention curves. We plot the average video bitrate and data traffic for five different arrival intervals and for BFTR ranging from 1/2 to 1/10 in Fig. 7a and 7b. Similarly, for different BFTR values, the average of video bitrate and data traffic for five retention curves are shown in Fig. 7c and Fig. 7d, respectively.

As confirmed from the results, the collaborative edge caching yields the improvement in average video bitrate for different values of BFTR as well as under the varying arrival intervals and clients' retention pattern. As mentioned in subsection VI-B, the reason is due to the possibility of exploring the chunk/bitrate of the requested video in the cache of not only the local but also the neighborhood edge servers with the collaborative caching. Further improvement is obtained with optimal collaborative strategy in which the knowledge about the future request of the clients is available in advance. As observed from Fig. 7a for different arrival intervals, the average of about 14% and 24% improvement in video bitrate is obtained with respectively collaborative and optimal collaborative caching compared to non-collaborative strategy. Considering different retention curves, as it is noticed form Fig. 7c, the average of about 12% and 20% improvement in video bitrate is achieved using respectively the collaborative and optimal collaborative caching compared to non-collaborative. The improvement in average bitrate using collaborative strategies under varying system parameters (clients' arrival, retention, BFTR parameter) obviously reveals the generality of the results.

On the other side, although the collaboration among the edge servers reduces the data traffic on origin server, it however adds to the data traffic on the fronthaul network. Depending on the BFTR value set by MNO, the average data traffic caused by collaborative caching may become worse than non-collaborative in some situations. For each value of BFTR parameter, the average data traffic per client time slot is obtained as $(1/S)\sum_{i=1}^{S} DT_i/|D_i - A_i|$, where S is the total number of clients. As the results in figures 7b and 7d show, there is a threshold on BFTR value after which the average data traffic per client time slot (for different arrival interval and retention curves) gets below the average data traffic caused by non-collaborative caching. As observed from the result, this average BFTR threshold value is very close to 1/2 for the case of different arrival intervals and is very close to 1/3 when different retention curves are considered. Therefore, the average BFTR threshold value of $min\{1/2, 1/3\} = 1/3$ is suggested such that after this threshold, the significant reduction in data traffic with further improvement in average video bitrate is guaranteed. *This threshold value indeed helps the edge caching designers to judge about the effectiveness of collaborative caching in DASH video streaming scenarios.*

It should be noted that this threshold value depends on parameters such as the weighting values in joint QoE-traffic optimization problem or even the size of the cache at the edge server. The reason is that as long as the volume of the created data traffic on the fronthaul network increases due to the frequency of cache misses at the local edge servers (by varying these impacting parameters), the BFTR threshold value is expected to decrease.

VII. CONCLUSION

In this paper, we investigated the impact of collaborative caching on joint QoE-traffic optimization in edge-assisted dynamic adaptive video streaming over HTTP (DASH) for particularly mobile edge computing (MEC) environments. To this end, we formulated the joint maximization of QoE and bitrate fairness and minimization of data traffic (backhaul and fronthaul) as an integer non-linear programming (INLP) optimization problem which introduces the backhaul/fronthaul traffic ratio (BFTR) parameter adjustable by mobile network operator (MNO). Due to NP-hardness of the problem formulation, we then design an online performance guaranteed algorithm with low complexity and minimum need for parameter tuning to solve the joint optimization problem. The algorithm utilizes a self-tuning mechanism for adjusting the weighting parameters in the optimization problem which in turn makes it easy for practical deployment by MNOs. We also design an efficient edge cache replacement heuristic which takes into account not only the retention behavior of the mobile clients toward different video request but also the collaboration among the edge servers.

Through simulations using the radio access link level traces of mobile clients, we show that our network-assisted bitrate adaptation and collaborative edge caching solution indeed helps to improve the QoE of the clients and reduce significantly the backhaul traffic compared to other network-assisted and collaborative edge caching mechanisms. Our simulation results also suggest the existence of some thresholds on BFTR parameter after which the significant improvement in average video bitrate and network data traffic are obtained using the collaborative edge caching. This threshold can help the edge-assisted DASH system developers to design efficient collaborative edge caching mechanisms for 5G mobile networks.

REFERENCES

[1] S. Petrangeli, J. Famaey, M. Claeys, S. Latré, and F. De Turck, "QoE-driven rate adaptation heuristic for fair adaptive video streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 12, no. 2, pp. 1–15, Oct. 2015.

[2] D. Bethanabhotla, G. Caire, and M. J. Neely, "Adaptive video streaming for wireless networks with multiple users and helpers," *IEEE Trans. Commun.*, vol. 63, no. 1, pp. 268–285, Jan. 2015.

[3] N. Bouten, S. Latre, J. Famaey, W. Van Leekwijck, and F. De Turck, "In-network quality optimization for adaptive video streaming services," *IEEE Trans. Multimedia*, vol. 16, no. 8, pp. 2281–2293, Dec. 2014.

[4] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proc. ACM MobiCom*, Sep. 2013, pp. 389–400.

[5] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2015.

[6] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: Between the devil and the deep blue sea," in *Proc. IEEE Int. Workshop Quality Multimedia Exper. (QoMEX)*, Aug. 2012, pp. 1–6.

[7] 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 *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 187–198.

[8] T. Mangla, N. Theera-Ampornpunt, M. Ammar, E. Zegura, and S. Bagchi, "Video through a crystal ball: Effect of bandwidth prediction quality on adaptive streaming in mobile environments," in *Proc. 8th ACM Int. Workshop Mobile Video*, May 2016, pp. 1–6.

[9] Z. Li, S. Zhao, D. Medhi, and I. Bouazizi, "Wireless video traffic bottleneck coordination with a DASH SAND framework," in *Proc. IEEE Vis. Commun. Image Process.*, Nov. 2016, pp. 1–4.

[10] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[11] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming," in *Proc. 7th ACM Int. Conf. Multimedia Syst. (MMSys)*, May 2016, pp. 1–12.

[12] E. Thomas, M. O. van Deventer, T. Stockhammer, A. C. Begen, M.-L. Champel, and O. Oyman, "Applications and deployments of server and network assisted DASH (SAND)," in *Proc. Int. Broadcast. Conv. (IBC)*, 2016, pp. 1–8.

[13] C. Wang, A. Rizk, and M. Zink, "SQUAD: A spectrum-based quality adaptation for dynamic adaptive streaming over HTTP," in *Proc. 7th ACM Int. Conf. Multimedia Syst. (MMSys)*, May 2016, pp. 1–12.

[14] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1435–1445, Oct. 2013.

[15] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[16] A. E. Essaili, Z. Wang, E. Steinbach, and L. Zhou, "QoE-based cross-layer optimization for uplink video transmission," *ACM Trans. Multimedia Comput. Commun.*, vol. 12, no. 1, pp. 1–22, Aug. 2015.

[17] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[18] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Edge computing assisted adaptive mobile video streaming," *IEEE Trans. Mobile Comput.*, pp. 1–17, Jun. 2018, doi: 10.1109/TMC.2018.2850026.

[19] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[20] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing," *ACM/IEEE Trans. Netw.*, vol. 24, no. 2, pp. 996–1010, Apr. 2016.

[21] A. Gupta and E. R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, Jul. 2015.

[22] Q. Ding, H. Pang, and L. Sun, "SAM: Cache space allocation in collaborative edge-caching network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[23] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[24] T. Tran and D. Pompili, "Octopus: A cooperative hierarchical caching strategy for cloud radio access networks," in *Proc. IEEE Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2016, pp. 154–162.

[25] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, Sep. 2016.

[26] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[27] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, Mar. 2017.

[28] H. Ahlehagh and S. Dey, "Video-aware scheduling and caching in the radio access network," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, Oct. 2014.

[29] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog *et al.*: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2018.

[30] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 116–122, Jun. 2018.

[31] S. Almajali, H. B. Salameh, M. Ayyash, and H. Elgala, "A framework for efficient and secured mobility of IoT devices in mobile edge computing," in *Proc. IEEE Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2018, pp. 58–62.

[32] M. Siekkinen, M. A. Hoque, and J. K. Nurminen, "Using viewing statistics to control energy and traffic overhead in mobile video streaming," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1489–1503, Jun. 2016.

[33] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[34] L. Lei, X. Xiong, L. Hou, and K. Zheng, "Collaborative edge caching through service function chaining: Architecture and challenges," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 94–102, Jun. 2018.

[35] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 80–87, Jun. 2018.

[36] C. Ge, N. Wang, S. Skillman, G. Foster, and Y. Cao, "QoE-driven DASH video caching and adaptation at 5G mobile edge," in *Proc. ACM Conf. Inf.-Centric Netw. (ICN)*, Sep. 2016, pp. 237–242.

[37] (2016). *Sandvine: Global Internet Phenomena Report*. [Online]. Available: https://www.sandvine.com/trends/global-internet-phenomena/

[38] *Dynamic Adaptive Streaming Over HTTP (DASH)*, document ISO/IEC 23009, 2017.

[39] *Dynamic Adaptive Streaming Over HTTP (DASH)—Part 5: Server and Network Assisted DASH (SAND)*, document ISO/IEC 23009-5, 2017.

[40] (2015). *SimuLTE Simulator*. [Online]. Available: http://simulte.com/

[41] (2009). *LTE Guideline*. [Online]. Available: http://www.etsi.org/deliver/etsi_tr/136900_136999/136942/08.02.00_60/tr_136942v080200p.pdf

[42] (2018). *BagPlot Description*. [Online]. Available: https://en.wikipedia.org/wiki/Bagplot

[43] (2010). *LIBRA Libraries Manual*. [Online]. Available: https://wis.kuleuven.be/stat/robust/papers/2010/HubertVerboven_LIBRA_WIRE_2010.pdf

ABBAS MEHRABI received the B.Sc. degree in computer engineering from the Shahid Bahonar University of Kerman, Iran, in 2008, the M.Sc. degree in computer engineering from Azad University, South Tehran, in 2010, and the Ph.D. degree from the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju, South Korea, in 2017. He is currently a Post-Doctoral Fellow at the Department of Computer Science, Aalto University, Espoo, Finland. His main research interests include quality of experience optimization and resource allocation for multimedia services in mobile edge computing environments, energy efficient mobile computing, and scheduling/planning problems in smart grids. He was one of the recipients of the Ph.D. Graduation Award from the Gwangju Institute of Science and Technology.

MATTI SIEKKINEN received the M.Sc. degree in computer science from the Helsinki University of Technology in 2003, and the Ph.D. degree from the EURECOM/University of Nice Sophia-Antipolis in 2006. He is currently a Senior Researcher with the University of Helsinki and Aalto University. His research on multimedia systems combines techniques from multimedia signal processing, mobile networking, cloud computing, system analysis, machine learning, and HCI.

ANTTI YLÄ-JÄÄSKI received the Ph.D. degree from ETH Zurich in 1993. From 1994 to 2009, he was with Nokia in several research and research management positions, with a focus on future Internet, mobile networks, applications, services, and service architectures. Since 2004, he has been a tenured Professor with the Department of Computer Science, Aalto University. His current research interests include mobile cloud computing, mobile multimedia systems, pervasive computing and communications, indoor positioning and navigation, energy efficient communications and computing, and Internet of Things.