QoE-based Mobility-aware Collaborative Video Streaming on the Edge of 5G

Mehmet Fatih Tuysuz and Mehmet Emin Aydin

***Abstract*—Today's Internet traffic is dominated by video streaming applications transmitted through wireless/cellular interfaces of mobile devices. Although ultra-high-definition videos are now easily transmitted through mobile devices, video quality level that users perceive is generally lower than expected due to distance-based high latency between sources and end-users. Mobile edge computing (MEC) paradigm is expected to address this issue and provide users with higher perceived Quality of Experience (QoE) for latency-critical applications, deploying MEC servers at edges. However, due to capacity concerns on MEC servers, a more comprehensive approach is needed to meet users' expectations applying all possible operations over the resources such as caching, pre-fetching and task offloading policies depending on the data repetition or memory/CPU utilization. To address these issues, this paper proposes a novel collaborative QoE-based mobility-aware video streaming scheme deployed at MEC servers. Throughout the paper, we demonstrate how the proposed scheme can be implemented so as to preserve the desired QoE level per user during entire video sessions. Performance of the proposed scheme has been investigated by extensive simulations. In comparison to existing schemes, the results illustrate that high efficiency is achieved through collaboration among MEC servers, utilizing explicit window size adaptation, collaborative pre-fetching and handover among the edges.**

***Index Terms*—QoE, 5G, Mobile edge computing, MEC, Video streaming, DASH.**

I. Introduction

NOWADAYS, rapid developments in wireless and cellular networks, such as Wi-Fi and 5G have made it possible to transmit, analyze and manipulate massive amount of data/applications quickly, remotely and efficiently than ever before. These technologies have also been shifting the mobile computing concept from centralized cloud to distributed edge(s). In this context, Mobile Edge Computing (MEC) paradigm that is accelerated with 5G pushes computation, content delivery, storage and virtualization to the network edge (e.g., base stations or access points) [1]. Advances in Radio Access Technologies (e.g., Wi-Fi, WiMax, 3G/LTE/5G) have also let user-centric multimedia applications dominate the Internet traffic. Studies reveal that 70% of today's Internet traffic is generated by video streaming applications, such as Netflix and YouTube [2]. With the proliferation of advanced mobile devices, acceleration of 5G networks and MEC servers positioned at the network edge within the Radio Access Network (RAN), this rate is forecast to increase 75% by the end of 2020 providing lower end-to-end communication delays and higher available bandwidth [3].

Online streaming services assisted by Content Delivery Networks (CDN) solely through cloud data centers lead to reduced QoE levels due to the high latency that occurs during data transmission between the users and the cloud. In this regard, Dynamic Adaptive video Streaming over HTTP (DASH) protocol [4] is adopted as a solution in the literature. DASH basically allows users and video players to switch to the most suitable video quality/bitrate according to the varying network condition. This feature of DASH mostly results in improved QoE since play-out interruptions are reduced/avoided.

Motivated by the specific capabilities that the mobile edge computing paradigm can provide to the Video-on-Demand (VoD) streams (e.g. caching, pre-fetching and task offloading), and the fact that studies in the literature cannot actually preserve the QoE values desired by users, a novel QoE estimation and adaptation method deployed over a MEC server is proposed in this paper. The proposed scheme first estimates a session-based QoE level gathering or computing information from the client-side and the medium-side. The proposed scheme later aims at preserving user-preferred QoE level during the whole session per user, performing three key manipulations; (i) session-based explicit TCP window size adaptation between the sender and edge node, (ii) pre-fetching some of segments/chunks to the edge server or neighboring edge(s), and (iii) load balancing through handover operation. In case the QoE cannot be preserved for a session despite the aforementioned manipulations, the proposed scheme simply adjusts the best suited video quality for that session through Adaptive Bit Rate (ABR) operation based on channel condition without harming QoE levels of other video sessions.

In line with the aforementioned explanations, main contributions of the paper are summarized as follows,

- While other studies aim to provide best-suited QoE value by decreasing/increasing the streamed video bitrates according to fluctuating channel condition, our primary goal is to preserve user preferred QoE level per session during the whole video play time even in case of channel fluctuations, performing key manipulations as mentioned earlier.
- Within the scope of this study, collaborative pre-fetching support is obtained from neighboring edge server(s). To the best of our knowledge, it is the first collaborative pre-fetching scheme utilized for video streaming among edge servers. Additionally, in contrast to other studies, this paper also provides collaborative load balancing through handover operations according to the mobility and location of the user.
- In contrast to other studies, we do not use any packet snif-

Manuscript submitted April 30, 2019.

M. F. Tuysuz is with Department of Computer Engineering, Harran University, Sanliurfa, Turkey (e-mail: ftuysuz@harran.edu.tr).

M. E. Aydin is with Department of Computer Science and Creative Technology, University of the West of England, Bristol, UK (e-mail: Mehmet.Aydin@uwe.ac.uk).

fer to compute session-based video buffer sizes, initial play-out delays and re-buffering intervals. Instead, our lightweight scheme monitors the throughput upon arrivals of acknowledgement (ACK) packets whether the difference between consumption and delivery is decreased below a certain threshold.

• Efficiency increase is achieved by applying the scheme only on edge server(s) without making any changes on the client-side. The work can therefore be easily installed and deployed.

II. Background and Related Works

5^{th} generation networks, *a.k.a.* 5G, is the latest generation of cellular mobile communication technologies that is expected not only to contain very-high carrier frequencies with enormous bandwidth and extraordinary numbers of antennas, but also to bind radio spectrum together with LTE and Wi-Fi providing ubiquitous coverage and improved user experience.

DASH is an advanced video delivery protocol, widely used by Content Service Providers (CSPs). With DASH, videos are first encoded/compressed into various representations that have different bitrates. Representations are divided into segments/chunks with identical time intervals. Information regarding the representations, their bitrates, segment intervals and URLs are kept in a manifest file called Media Presentation Description (MPD). In this way, video players can request/receive MPD files, select a bitrate at the beginning and switch to different bitrates during the video play according to fluctuating channel condition to provide users with a better QoE.

Four essential metrics assist estimating the QoE level of a DASH session [5]; (i) *video quality*, the selected bitrate for the video session, (ii) *initial play-out gap*, the time interval elapsed until the video starts playing after the first click, (iii) *re-buffering*, the duration of video freezing during a session, (iv) *bitrate switching*, the number of bitrate switches during a session. Studies presented in the literature mainly use some/all of these metrics to estimate the QoE levels in real-time for video streams. In this context, state of the art solutions focusing on QoE estimation/improvement are discussed below.

• *Video quality adaptation*: There have been works [6]–[11] dynamically adapt video quality through the ABR feature of DASH protocol to limit/avoid re-buffering. These works utilize achieved/estimated throughput or available buffer size of video sessions at the client-side. For instance, Li et al. in [6] utilize historic per-segment downlink throughput to forecast upcoming throughput and choose the best suited video quality accordingly. Liu et al. in [7] adjust video rates detecting bandwidth variations using a smoothed HTTP throughput measured based on the segment fetch time (SFT). Note that none of these works utilize MEC facilities and they simply require additional message exchanges between users and networks.

There are also solutions utilizing MEC and QoE indicators for video quality adaptation, such as the work that estimates aforementioned four QoE metrics of video streams in real-time through a packet sniffer [8], the work that utilizes a time-slot system with a look-ahead window for computing the switching cost among edges [9], the work that proposes an integer non-linear programming optimization for bitrate adaptation jointly maximizing the QoE and proportional fairness [10], the work that presents a 5G-QoE scheme to handle the QoE modeling for UHD video flows in 5G networks [11]. Note that these MEC-based solutions mostly adapt to lower video bitrates to minimize re-buffering interval as a common practice in case of channel fluctuations. Yet, such a strategy is not desired for most of users connecting to the Internet that primarily aims at providing low latency and high-bandwidth.

• *Pre-fetching at the edge*: Many solutions are also proposed regarding pre-fetching of video streams on client or network-side. Approaches utilized on client-side includes downloading multiple segments in parallel through multi-path TCP [12], and a recommender that predicts videos likely to be watched by users [13]. Note that client-side video pre-fetching strategies do not wisely consider end-to-end path between Radio Access Networks (RAN) and the cloud, hence experience sub-optimal TCP performance. On the other hand, Approaches utilized on the network-side includes deploying pre-fetching proxies at wireless access points [14] or cellular base stations [15] with rigid pre-defined pre-fetching policies, or pre-fetching video segments from cloud to the MEC server maintaining a progress gap ahead of the user's actual request progress [16]. Since a collaborative pre-fetching among network entities/edges is not considered in these works, pre-fetching on solely one network edge may not perform well, or even may damage the QoE level, in dense networks due to possible buffer overflows.

Although aforementioned solutions basically provide users with higher QoE levels, there are in fact several metrics not sufficiently discussed among these solutions, such as explicit window size adaptation, pre-fetching from neighboring edges, and handover to another network within coverage. Throughout the paper, we show that by manipulating the aforementioned metrics, it is possible to preserve the UE-preferred QoE level during the whole video session per user as long as the edge server is able to collaborate with other edges.

III. Problem Formulation and QoE Modeling

Let the entire system has u users, each of which is connected to the same BS with an edge server, streaming video through the cloud where r distinct video representations are available as seen in Fig. 1. Suppose there are $e-1$ neighbouring edge servers in vicinity that users are able to handover or pre-fetch some data in case certain conditions are met. Here, as in [17], the binary variable $x_{i,j,k}$ represents whether user i accesses video representation k over the edge server j, where $i \in [1,u], j \in [1,e], k \in [1,r]$. Assume b_k and QoE_k denote the video bitrate of the representation k and its QoE value, respectively. Also assume each user downloads a single video representation at any time through a single radio interface, each edge server has a specific assignment capacity of $C_j, j \in [1,e]$ and each individual radio link between user i and the edge server j has a specific link capacity of l_i^j.

Our primary goal is to preserve the UE-preferred QoE level of each video session throughout the entire video play duration without harming the QoE levels of other sessions. To achieve QoE preserving, the proposed scheme that is implemented on the MEC server should keep track of both

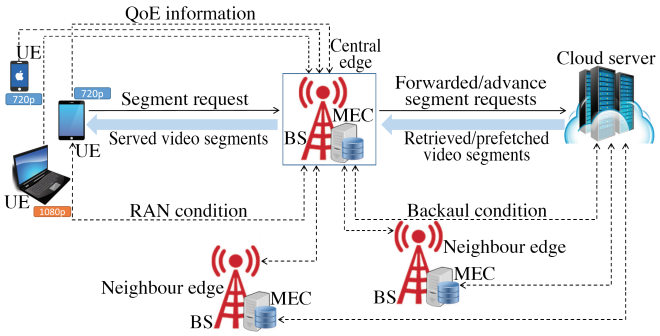


Fig. 1: Proposed System Architecture

initially assured and meanwhile computed video quality level per user ($QoE_{k,asr}$ and $QoE_{k,cmp}$) and the sum of bitrates of connected users per edge network ($\sum_{i=1}^{u}\sum_{k=1}^{r} x_{i,j,k}b_k$) due to capacity calculations of the edge(s).

Consequently, the scheme is expected to provide sessions with no less QoE levels than their initial UE-preferred values;

$$x_{i,j,k}QoE_{k,cmp} \geq x_{i,j,k}QoE_{k,asr}, \quad \forall i \in [1,u], \forall j \in [1,e], \forall k \in [1,r] \tag{1}$$

subject to the constraints below,

$$\sum_{i=1}^{u}\sum_{k=1}^{r} x_{i,j,k}b_k \leq C_j, \forall j \in [1,e] \tag{2}$$

$$x_{i,j,k}b_k \leq l_i^j, \forall i \in [1,u], \forall j \in [1,e], \forall k \in [1,r] \tag{3}$$

$$\sum_{k=1}^{r}\sum_{j=1}^{e} x_{i,j,k} \leq 1, \forall i \in [1,u] \tag{4}$$

$$x_{i,j,k} \in \{0,1\}, \forall i \in [1,u], \forall j \in [1,e], \forall k \in [1,r] \tag{5}$$

where, the constraint given in Eq.(2) ensures the capacity of edge server j is not surpassed by the assignment load, the constraint given in Eq.(3) ensures a video representation is not allocated to user i on the edge j in case its bitrate cannot be accommodated by the user's link capacity, the constraint given in Eq.(4) ensures a user is allocated with one video representation at most, and the constraint given in Eq.(5) ensures each device utilizes at most one edge network for video streaming as it receives one video representation at most.

Video streams are formed as a set of continuous chunks that contain s seconds of video parts. Chunks are encoded with different bitrates (b_k) for various video representations. At any time, a video chunk v_i (where $i \in [1,z]$) can be downloaded with a bitrate $b_k(i)$ (where $k \in [1,r]$). Assume $d\left(b_k(i)\right)$ is the size of the chunk v_i at bitrate $b_k(i)$ and Dw_i is the average download speed. If the chunk v_i is started to be downloaded at time t, downloading duration becomes $d\left(b_k(i)\right)/Dw_i$.

Note that buffers (B) allocated for video sessions revolve as chunks are being downloaded and video is being played. Hence, an allocated buffer increases by s seconds in case a new chunk is downloaded and decreases as it is being watched. The buffer dynamics can be formulated as in [9] as follow,

$$B_{i+1} = \left(\left(B_i - \frac{d\left(b_k(i)\right)}{Dw_i}\right)_+ + s\right)_+ \tag{6}$$

here the notation $(y)_+ = \max\{y,0\}$ assures positiveness of the term. Also note that re-buffering occurs in case $B_i < d\left(b_k(i)\right)/Dw_i$ since the buffer becomes empty while the video player is still downloading the chunk v_i.

Considering both the video bitrate quality rate $q\left(b_k(i)\right)$ and the QoE value are mapped between 0 and 5 as in MOS[1], the average video quality Q_{avg} over z chunks becomes,

$$Q_{avg} = \frac{1}{z}\sum_{i=1}^{z} q\left(b_k(i)\right) \tag{7}$$

Total video re-buffering time $T_{\text{re-buffering}}$ and the re-buffering ratio R_r over z chunks can be expressed as follow,

$$T_{\text{re-buffering}} = \sum_{i=1}^{z}\left(\frac{d\left(b_k(i)\right)}{Dw_i} - B_i\right)_+ \tag{8}$$

$$R_r = \frac{T_{\text{re-buffering}}}{T_{\text{duration}}}; \tag{9}$$

In addition, the frequency of bitrate switching f_s and the proportion of play-out gap g_p considering the whole video session can be expressed as $\frac{z_{\text{switch}}}{z}$ and $\frac{T_{\text{gap}}}{T_{\text{duration}}}$, respectively. Consequently, the QoE value of any video session is expressed below as a weighted sum of the aforementioned four metrics.

$$QoE = Q_{avg} - \alpha R_r - \beta f_s - \gamma g_p \tag{10}$$

where T_{duration} is the total time of the video session including the first play-out gap T_{gap}, video play time $s \times z$ and re-buffering time. Finally, α, β, and γ are the weight parameters[2] regarding re-buffering ratio, bitrate switching frequency, and the play-out gap, respectively. Also note that QoE level is indirectly related to QoS metrics, such as achieved throughput, packet loss rate, latency, jitter and out-of-order delivery in a network. In this context, each metrics of the Eq.(10) (Q_{avg}, R_r, f_s, g_p) is actually affected by the QoS metrics.

IV. Architecture Overview

A. Bandwidth-aware Explicit Window Size Adaptation

TCP congestion control mechanism is negatively affected by hostile attitude of wireless channels and mobile nature of wireless users [19]. In this context, split connection approaches that host a mediator (e.g., agent) in between sender and receiver such as TCP snoop [20] are proposed in the literature to increase the transmission performance of hybrid networks of wired and single-hop wireless links.

Let $B^e(t)$ denote the empty buffer size of a session on the edge server at time t, which is $B^e(t) = B^{total} - B^o(t)$, where B^{total} and $B^o(t)$ are the total and occupied buffer sizes at time

[1]Mean opinion score (MOS) is a measure used for the Qos/QoE computations, representing overall quality of a stimulus or system.

[2]Based on the QoE computations carried out by other studies [9], [18], and the impact of specific parameters on the quality level, the parameters α, β, and γ are assigned as 3, 0.1, 0.02, respectively throughout the simulations.

t, respectively. Assume $W_{rcv}(t)$ is the user's advertised empty window size that is embedded in the ACK at time t. This buffer size advertised by ACK frame corresponds to the buffer maintained at the transport layer of the UE. Yet, the available buffer size at the edge server is also important due to the nature of split TCP video sessions. Hence, the proposed scheme computes an estimated window size $f(B^e(t))$ for video sessions as a function of sessions' bandwidth requirement: bitrate of video chunks and throughput rate averaged over the last p IP packets successfully received by the user. This value is then used to mark up/down the user's advertised empty window size in the ACK depending on the number of packets received by the user; whether the number is higher than necessary or lower than enough packets. Consequently, average throughput rate over the last p IP packets can be calculated as in [21],

$$\Theta_{\text{avg}} = \frac{1}{p} \sum_{i=L_p-p+1}^{L_p} \frac{S_{\text{packet}}(i)}{T_{\text{ACK}}(i) - T_{\text{fetch}}(i)} \tag{11}$$

where L_p is the index of the last packet successfully received by the user, $T_{\text{ACK}}(i)$ is the time at which ACK_i is received by the edge server, $T_{\text{fetch}}(i)$ is the time when the packet i enters into the edge server's queue, and $S_{\text{packet}}(i)$ is the packet size for the packet i. In this respect, to preserve a QoE level of a video session, the average throughput calculated above must be bigger than the throughput required for the bitrate selected by the user/player at any time of that video session: $\Theta_{\text{avg}} \geq \Theta_{\text{req}}(b_k)$ where $k \in [1, r]$. Consequently, advertised window size of the session is modified as follows,

$$W'_{rcv}(t) = \begin{cases} \min\left(W_{rcv}(t), f\left(B^e(t)\right)\right) & \text{if } \Theta_{\text{avg}} \geq \delta\Theta_{\text{req}}(b_k) \\ \max\left(W_{rcv}(t), f\left(B^e(t)\right)\right) & \text{otherwise} \end{cases} \tag{12}$$

where δ is the weight parameter[3]. Let's consider an ideal channel condition scenario where u identical TCP sessions with zero RTTs are active (radio link propagation is assumed to be negligible). In this context, the window size transmitted to the source at time t for the session i will be set to $W_i(t) = B^e(t) = B^{total} - B^o(t)$ where $i \in [1, u]$. Therefore, we can assume that the buffer occupied by each session is equal to its window size W_i. That is $B_i^o(t) = W_i(t)$. Considering the same scenario, total buffer occupancy at time t is set as $B^o(t) = uB_i^o(t)$. Hence, under ideal condition, it is expected that the system will converge to the state of $B_i^o(t) = B^{total} - uB_i^o(t)$. Thus, the expected buffer occupied by each TCP session in steady state becomes $B_i^o = \frac{B^{total}}{u+1}$ [22]. Since video sessions are not identical and have different throughput requirements based on the bitrates utilized, buffer requirement of each session must be set based on the proportion of required throughput per session to the total amount of throughput required for all sessions as follows,

$$B_i^o(t) = \frac{\Theta_{\text{req}}(b_k(i))}{\sum_{i=1}^{u} \Theta_{\text{req}}(b_k(i))} B^o(t) = \frac{\Theta_{\text{req}}(b_k(i))}{\sum_{i=1}^{u} \Theta_{\text{req}}(b_k(i))} \frac{uB^{total}}{u+1} \tag{13}$$

[3]Within the scope of this paper, a preliminary study was performed on the values that the parameter δ can take, by setting the δ as 1, 1.5, 2, and 2.5, respectively. In all simulation scenarios where different number of users and different ratios of channel density were tested, it was found that δ values between 1 and 2 always give the best window size values for video sessions. In this context, δ value is set constant as 1.5 throughout the simulations.

Accordingly, the estimated empty window size is set as $f(B^e(t)) = B_i^o(t)$. As seen from the Eq. (12), while a session that already obtains higher throughput than its bitrate advertises its empty window size as the minimum value between $W_{rcv}(t)$ and $f\left(B^e(t)\right)$, a session that obtains less throughput than its bitrate advertises its empty window size as the maximum value between the $W_{rcv}(t)$ and the $f\left(B^e(t)\right)$.

B. QoE-aware Cooperative Pre-fetching

Pre-fetching data, which is the operation of transferring video streams already cached at the edge server(s) to the user, enables fully utilizing the front-end throughput between the edge server and the user. Deciding on how many and which chunks to be pre-fetched ahead of the user's progressing request during a session is the key factor to preserve the video bitrate per user. Yet, there is a trade-off required to be balanced: pre-fetching chunks less than the actual requirement of consumption may lead to a higher risk of re-buffering, while pre-fetching more than required would result in redundant traffic at the backhaul and unnecessarily waste buffer space at the edge server. Therefore, pre-fetching progress and the frequency of the edge server must be at least as fast as the user's request progress, while a bit faster is preferred, in general [16].

Suppose that each video chunk, v_i (where $i \in [1, z]$), contains one second of video parts. Also suppose that the user's last request was for the chunk v_{i+1} and the round[4] length is π seconds. This means the user is expected to retrieve the chunk v_{i+2} to chunk $v_{i+\pi+1}$ in the next round. These chunks will be requested from the cloud server in case the buffer at the edge server does not have these chunks (v_{i+2} to $v_{i+\pi+1}$). In this case, the next chunks ($v_{i+\pi+2}$ to $v_{i+2\pi+1}$) can be pre-fetched by the edge server. Consequently, the expected number of chunks, N_{pf}, to be pre-fetched in a round length, π, by the edge server can be calculated considering the empty buffer size, B^e, and the buffer size required for each chunk, $B^{\text{req}}(b_k)$, that has the bitrate b_k as

$$N_{\text{pf}} = \frac{B^e \pi}{B^{\text{req}}(b_k)} \tag{14}$$

In any time, if there are chunks pre-fetched at the edge server and are not yet played by the UE, high amount of latency reduction can be achieved transferring these chunks in the next rounds to the UE directly from the edge server. This also makes it possible for video sessions to preserve their video bitrates even if cloud-edge path cannot always sustain the required bandwidth. Yet, if backhaul throughput at the cloud-edge path is always (or for a long time) less than the required throughput of the selected bitrate, there will be no chunks left in the UE's buffer after a while and re-buffering will start since downloading chunks will take more time than their playing time (See Eq.(6)). In this case, pre-fetching through single edge server will also not work in long term. In this context, pre-fetching over multiple edges can prevent further reduction of the available buffer size of the edge server.

[4]Task assignments for video flows are executed for a pre-defined interval, which is defined as a *round* [23]

Within the scope of this paper, it is aimed that the number of chunks pre-fetched to the edge server(s) is not less than the number of chunks consumed by the video player in any round to preserve the UE-preferred video bitrate per session. Consequently, pre-fetching chunks with the UE-preferred bitrate is possible as long as the following holds,

$$T_\pi \geq \frac{\sum_{j=1}^{e} d\left(b_k\right)/Dw_j}{e} + s \tag{15}$$

where T_π is the time remaining to play-back the next chunk to be pre-fetched, which is not yet in the buffer, $d\left(b_k\right)$ is the size of the chunk at bitrate b_k, Dw_j is the downloading speed of the edge server j and s is the chunk duration. In this regard, to preserve QoE levels of video sessions, proposed pre-fetching scheme decides how many edge servers are required to make the average throughput obtained in each round bigger than the throughput required for the bitrate selected per session.

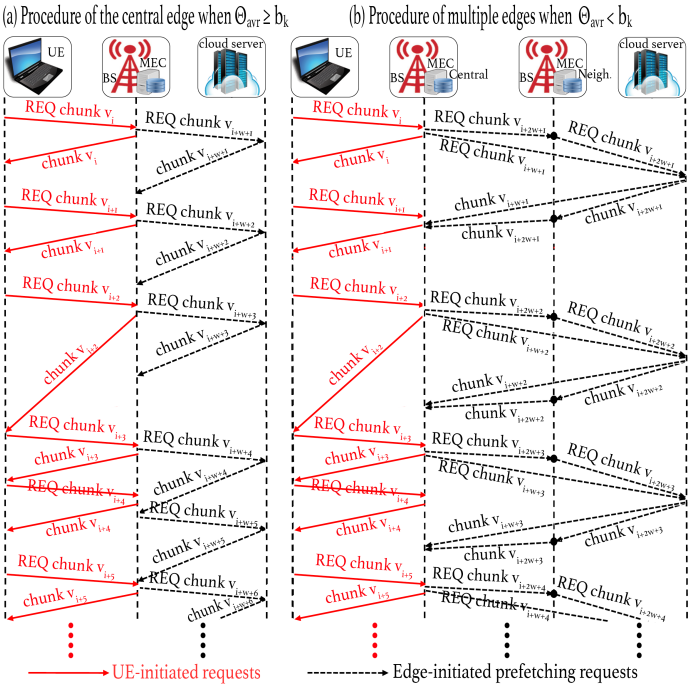


Fig. 2: Illustration of the proposed pre-fetching scheme

The collaborative pre-fetching scheme is illustrated in Fig. 2. While Fig. 2a demonstrates the procedures taken by the central edge server in case the $\Theta_{\text{avg}} \geq \Theta_{\text{req}}(b_k)$ is achieved through a single edge, Fig. 2b demonstrates the procedures taken by both central edge and other collaborative edge(s). In both scenarios, while the chunks already cached at the edge server(s) are directly sent to the UE after a request risen from the UE, the chunks to be needed in the next round(s) are also pre-fetched from the cloud with the proposed collaborative scheme, which helps prevent possible re-buffering problem by setting the number of pre-fetched chunks based on the number of chunks requested.

C. QoE-aware Load Balancing through Handover

In case, one of the BSs is highly loaded, a portion of its load can be transferred to different BSs with a handover operation. Initial step in the handover operation is the discovery of available BSs in vicinity and the easiest way to do it is periodic scanning or instant message exchanges between MEC servers deployed at BSs. Frequent message exchanges would be useful in case UEs are highly mobile. Yet, it may result in substantial overhead in a static environment. In order to reduce the aforementioned overhead, we propose a smart discovery and handover scheme, as in [24], that initiates the handover operation only if a MEC server uses its capacity above a certain threshold, τ_c, or if the signal strength value of a session falls below a pre-defined threshold, τ_s. This way, even in the presence of capacity or signal strength-related issues, it is aimed to preserve UE-preferred QoE levels, associating UEs with a proper BS that has high signal strength.

When τ_c is exceeded or τ_s of a session drops below a threshold, the proposed scheme first discover the set of neighboring edges that has enough capacity by requesting context-aware information from BSs in the vicinity, such as their positions, available buffer sizes and channel utilization ratios. The proposed scheme then lists video sessions in ascending order according to their signal strength . Afterwards, the scheme creates a distance-based connectivity map for the session at the top of the list based on the location of BSs and the signal strength values of the UE received from each BS. Handover procedure is initiated in case there is another BS (with sufficient capacity) that is expected to provide higher signal strength for the session. This operation is executed for all sessions that fall below the signal strength threshold. In case the capacity still exceeds the τ_c after the above-mentioned executions, the handover operation continues until either the capacity falls below the threshold again or the list is empty.

In case a handover operation is decided to be initiated, the pre-fetched chunks at the central server are first forwarded to the edge server that the session will be connected to, before the handover is initiated. This allows the session to consume the chunks already present in the UE's buffer while the handover process is in progress, and to pre-fetch the chunks of the next round(s) quickly from the new edge server after the handover.

D. The Proposed Solution

A simplified flow chart of the proposed scheme is shown in Fig. 3. The scheme enables edge server responsible for handling requests, admissions, QoE estimations and quality adaptations of video streams. As seen from Fig. 3, a new TCP split connection is first opened per session that has been entered to the medium. Thus, RAN-based channel fluctuations and user-based mobility issues are limited per new session, making the TCP congestion control works properly. Next, QoE modeling is performed per new session according to the user bitrate preference. This way, the QoE level to be preserved, and the amount of throughput required, $\Theta_{\text{req}}(b_k)$, for the selected bitrate per session until the end of the video play is determined.

Following the QoE modeling, the scheme reads the amount of throughput required per session, computes average throughput values obtained in the previous round, and initializes the pre-fetching scheme explained in Section III(b), computing the expected number of chunks to be pre-fetched in the next

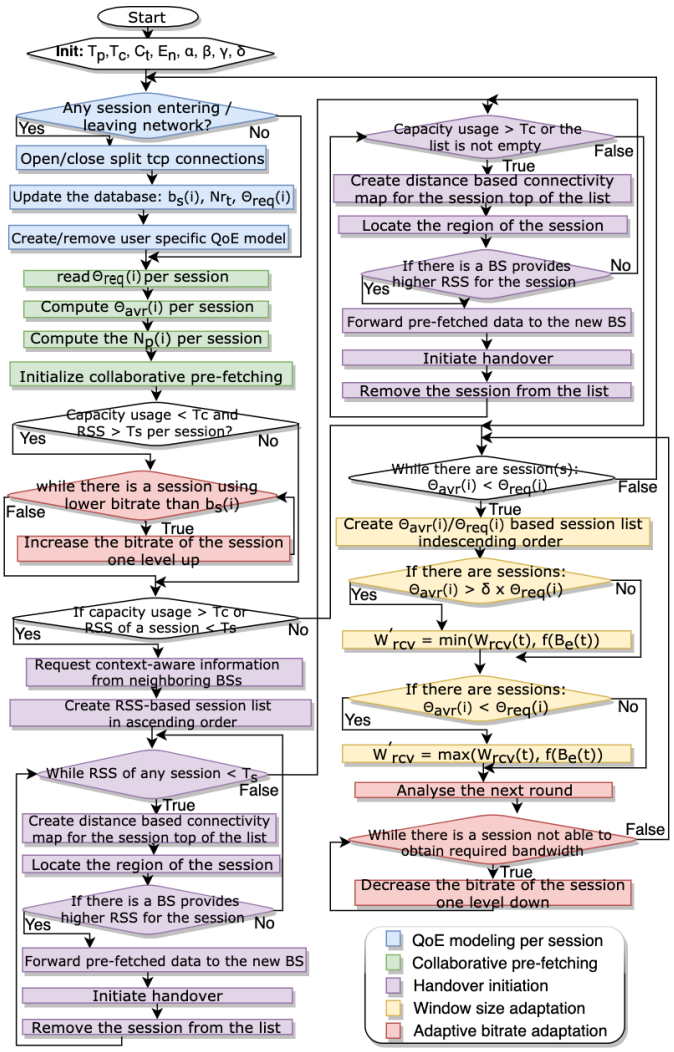


Fig. 3: Flowchart of the proposed scheme

round. This way, pre-fetched video chunks are transmitted to the UE with low-latency. In case chunks are not pre-fetched, the proposed scheme forwards the request to the cloud server, retrieves the requested chunks and transmit them to the UE.

After pre-fetching, the scheme checks whether the capacity usage of the edge server is lower than the τ_c or received signal strength (RSS) of sessions are higher than the τ_s. The condition is simply checked to increase the bitrate of session(s) that has no network-related or mobility-based issues currently, yet is using a lower bitrate than the UE-preferred bitrate.

In case network-related or mobility-based issues exist, context-aware information is requested from neighboring BSs, RSS-based session list and distance-based connectivity map is created, and handover procedure is initiated if there is another BS (with sufficient capacity) that is expected to provide higher signal strength for the session(s) having signal strength issues. The scheme also monitors the average throughput obtained in each round per session. In case it is less than the throughput required for a selected bitrate, the scheme initiates its explicit window size adaptation for session(s) having less throughput and for session(s) having more throughput than a δ coefficient, as explained in Section III(a). The scheme also monitors the network in the next round to see if the actions taken have addressed the bandwidth issue for these sessions. In case there is still a session that is having less throughput than required, the scheme decreases the bitrate of the session one level down.

Within the scope of the paper, QoE levels expected to be achieved with the proposed scheme and the traditional cloud-UE communication scheme are also studied analytically. This is to show that the QoE level obtained by the proposed scheme at any time of any session will not be lower than that of the cloud-UE communication scheme.

Note that each component of the QoE formula given in Eq.(10) is dependent on QoS metrics, such as downlink speed, (which is expected to be the same for both schemes since the sender and receiver are the same role players in both methods), end-to-end latency and packet loss rate. In this context, it is clear to say that the scheme, which will have lower latency and packet loss rate will have a higher QoE level.

Let ℓ and $\hat{\ell}$ denote end-to-end latency between the sender (cloud server) and receiver (UE) for an ordinary video run and a video run via MEC servers, respectively. The latency with the cloud-UE communication scheme is calculated as follows:

$$\ell = z\{(1 + P_{l_W})\ell_W + (1 + P_{l_R})\ell_R\} \tag{16}$$

where z is the number of video chunks, P_{l_W} is the packet loss rate between the cloud and the BS, P_{l_R} is the packet loss rate between the BS and the UE, and finally, ℓ_W and ℓ_R are the sum of constituting delays from processing, (typically a few micro-secs or less), queuing (depends on congestion), transmission (insignificant for high-speed links) and propagation (depends on distance between sender and receiver) in between the cloud and BS, and in between the BS and UE, respectively.

On the other hand, since some of video chunks (e.g. m) are expected to be pre-fetched earlier without user request, initiated by the MEC server deployed on the BS with the proposed scheme, the end-to-end latency between the cloud and UE can be computed as follows:

$$\hat{\ell} = z\{(1 + P_{l_W})\ell_W + (1 + P_{l_R})\ell_R\} - m(1 + P'_{l_W})\ell'_W \tag{17}$$

where P'_{l_W} and ℓ'_W are the packet loss rate and latency applied to the m packets pre-fetched ealier, say within the time period of $[\pi_0, \pi_m]$. The dowload speed of pre-fetched chunks to the UE is expected to be faster than the download speed of chunks through backhaul link between the cloud and UE due to the physical distances.

As explained in detail throughout this section, coopeative pre-fetching and hand-over approaches are carried out for DASH-based video sessions within the scope of the study.

Definition 1. Collaborative pre-fetching is to allow storing video chucks on multiple edge servers within a neighbourhood.

Since pre-fetching enables fully utilizing front-end throughput between edge and UE, the following theorem can be defined.

Theorem 1. *Collaborative pre-fetching reduces the end-to-end latency extending the total available buffer size per UE.*

Proof. Here, latency considered for reduction is the latency in between cloud and MEC server since no change is expected to happen in between MEC and UE. The latency over wired connection between cloud and MEC server can be splitted as $\ell_W = \ell_W(\pi_0, \pi_m) + \ell_W(\pi_m, \pi_z)$, where π_0, π_m and π_z are the times at the beginning, the time when m packets are downloaded and stored on MEC server and the time when all chunks are completely downloaded. It is assumed that $Dw(\pi_0, \pi_m) \geq Dw(\pi_0, \pi_z)$ due to growing backhaul traffic. Since buffer size of a MEC server is much bigger than buffer size of a UE, $\ell_W(\pi_0, \pi_m) \geq \ell'_W(\pi_0, \pi_m)$, where $\ell'_W(\pi_0, \pi_m)$ is the latency of the corresponding time period when MEC buffer is used. The theorem suggest that $\ell \geq \hat{\ell}$, from Eq.(16), Eq.(17) can be rewritten as $\hat{\ell} = \ell - m(1 + P'_{l_W})\ell'_W$. In the worst case, second term of righthand side of the equation, $m(1+P'_{l_W})\ell'_W$, will approximate to 0, hence $\hat{\ell} = \ell$, otherwise it will be $\hat{\ell} < \ell$, which simply proves the theorem. □

In addition to the end-to-end latency, packet loss rate also affects QoE. Let SS_i and SS_j are the signal streghts between a UE and MEC i and MEC j, where $SS_i \leq SS_j$ while UE is held by MEC i. The level of packet loss rate with SS_i and SS_j are denoted with $P_{l_R}(SS_i)$ and $P_{l_R}(SS_j)$, respectively. Due to the signal strengths, it is expeceted $P_{l_R}(SS_i) \geq P_{l_R}(SS_j)$. As discussed before, once $SS_i \leq \tau_s$ a handovered operation is conducted for UE to be held by MEC j, which is expected to have a stronger signal strength. In this respect, the following statement and theorem of packet loss rate can also be defined.

Definition 2. Handover among BSs is an essential factor to avoid/limit RAN-based congestion and packet errors.

Theorem 2. *Collaborative handover approach reduces the overall packet loss rate increasing the average RSS of UEs.*

Proof. Let ℓ_i and ℓ_j be latencies to be calculated when a UE is held by MEC i, and by MEC j, respectively. The theorem suggests that $\ell_i \geq \ell_j$, given that $SS_i \leq SS_j$. As can be seen from Eq.(16) and (17), P_{l_R} has increasing effect upon the righthand side of both equations. In case single MEC server is used in the system or $SS_i \geq \tau_s$, P_{l_R} remains the same as in the cloud-UE communication scheme since the sender and receiver are the same entities in both methods. However, once UE is handovered from MEC i to MEC j given that $SS_i \leq SS_j$, the packet loss rate will decrease to $P_{l_R}(SS_j)$, due to $P_{l_R}(SS_i) \geq P_{l_R}(SS_j)$, hence $\ell_i \geq \ell_j$ is held, which proves the theorem. □

Consequently, collaborative pre-fetching and handover help extending the total buffer size, reduce both the latency and the packet loss rate, and hence provide higher QoE levels per UE.

Definition 3. QoE is a function of latency, packet loss rate and bitrate, $QoE = f(\ell, P_l, W)$.

Theorem 3. *QoE per user improves with reduced latency and packet loss rate.*

Proof. As demonstarted with Theorem (1) and (2), both latency and packet loss rate can be improved. From Eq.(10) and Definition (3), QoE will be affected of any change in ℓ, P_l and W. The proposed algorithm devises a procedure to reduce both ℓ and P_l as demonstrated with Theorem (1) and (2). From Eq.(10), any decrease achieved in both of ℓ and P_l will result in improved QoE. □

It should also be noted that the proposed scheme is basically a distributed network system among cloud(s), edge(s), and clients. The system also includes various parties, such as Mobile Operators, Content Providers and clients. As a result, the large and distributed structure of the proposed system makes it also suitable for the blockchain integration.

Blockchain, has become one of the promising distributed secure data management architecture. Even though it has been broadly adopted in numerous fields from finance to healthcare in the literature, its application in mobile networks is yet limited due to the fact that blockchain clients require solving preset proof-of-work puzzles to add new data to the blockchain, which in fact consumes significant resources such as CPU and energy that is not applicable for resource-limited mobile devices [25]. Yet, mobile edge computing paradigm can be a promising factor to solve the issue of significant resource requirements for resource-limited mobile devices.

Making use of MEC, blockchain-related computing tasks can be easily offloaded from UEs to edge servers and hence, trust among providers and data integrity can be achieved throughout the entire communication from the cloud(s) to the client(s). In addition, it should also be noted that withouth blockchain integration, MEC itself might have difficulties to ensure optimal system performance due to uneven distributed resource demands of edge servers and UEs, which may result in higher transmission delay and data loss for DASH-based video sessions. Consequently, although additional costs are required for edge-servers (CPU, energy, etc.) for blockchain operations, client-based cost reductions may also occur on UEs as a result of MEC and blockchain integration due to possible performance increase (e.g. latency and loss).

V. Performance Evaluation

Within the scope of the study, a simulation scenario that consists of a resource-constrained video server, LTE-A cells with connected MEC servers and varying number of wireless UEs streaming DASH-based video from the cloud is simulated in OMNET++ through SimuLTE [26] which is an open-source platform built on top of OMNeT++ and INET Framework.

A. Experiment Setup

Simulation topology that consists of one cloud server, one central BS with two neighboring BSs, and total of 90 randomly distributed UEs connected to the central BS is shown in Fig. 4. An HTTP video server that has 2Gbps link capacity is placed away from the edge. All UEs that have a maximum bandwidth of 112Mbps are initially associated with the central BS that has the central edge server running the proposed scheme. Parameters used in simulation are shown in Table I. As in [27], simulation interval is set as 300s where each UE has unique video streams that lasts for 270s. Start-up times of video sessions are randomly selected by UEs from the uniform interval [1; 30s]. 4 bitrates are available per video at the source

TABLE I: Simulation Parameters

Application parameters	
Application type	Adaptive HTTP streaming
Number of representation	4 [4; 8; 16; 32 Mbps]
Round/segment size	5s
Chunk Size	1s
Video duration	270s
Network parameters	
Edge server capacity τ_c	80%
UE signal threshold τ_s	-62 dBm
Backhaul link capacity	2 Gbps
UE max. RAN bandwidth	112 Mbps
Backhaul latency	250 - 300 ms
RAN latency	10 - 20 ms
Simulation parameters	
Number of UE sessions	90
Edge server buffer size	10 Gbps
UE buffer size	250 Mbps
Bitrate video quality $Q_{b_k}(i)$	4 [3.9], 8 [4.2], 16 [4.5], 32 [4.9]
Weight parameters	α [3], β [0.1], γ [0.02], δ [1.5]
Initial play-out delay	Fixed [2s]
Simulation time	300s
Simulation domain	500x500m
Velocity	Stationary or mobile [10 m/s]

[4; 8; 16; 32 Mbps]. At the beginning of each simulation, 32Mbps bitrate is selected for 70% of sessions, 16Mbps bitrate is selected for 20% of sessions and 8Mbps bitrate ise selected for 10% of sessions. While buffer size of 250Mbps is set per UE, 10Gbps buffer is considered per edge server.

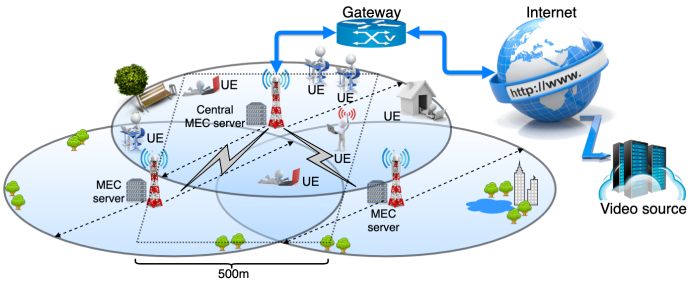


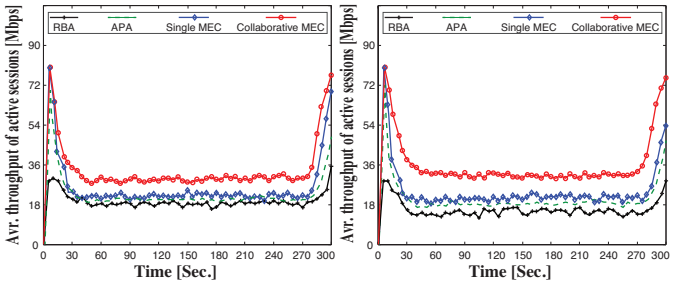
Fig. 4: Simulation topology

Throughout simulations, performance of the following four video streaming methods are evaluated: (i) conventional rate-based adaptation (RBA)[5] strategy between the cloud and UEs, (ii) adaptive pre-fetching approach (APA)[6] proposed in [16], (iii) central edge server performing the proposed scheme on its own (Single MEC), and (iv) central edge server performing the proposed scheme collaborating with other edge servers (Collaborative MEC). Simulations are tested on two different scenarios where UEs are either stationary or mobile during the whole process. Throughout simulations, comparisons are made by computing the average value of 30 simulation runs with confidence interval of 95%. In addition, 2-Ray Ground model and Random Waypoint Mobility model are used for the propagation modelling and movement path generation of UEs.

[5]The highest bitrate among available ones is selected per UE based on the throughput obtained from the previous x chunks in the RBA strategy.

[6]Adaptive pre-fetching is performed on a per-session basis, i.e., the scheme pre-downloads video segments from the source and maintains a progress gap ahead of the users actual request progress. Such a gap is adaptive and is optimized based on its real-time knowledge on network and user context.

B. Impact of the Proposed Scheme on Throughput and Buffer

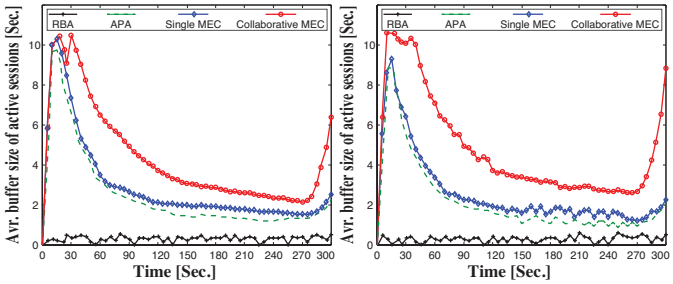


(a) Stationary UEs (b) Mobile UEs

Fig. 5: Average throughput of active sessions

Based on the simulation scenario, it is expected that the network gradually intensify for 30 seconds and 2 Gbps link capacity to be exceeded after a point. Some of sessions are also expected to reduce their UE-preferred bitrates to make the required link capacity lower than the capacity the network can provide. Average throughput values of four video streaming methods examined are shown in Fig. 5. While all sessions are randomly positioned within coverage of the BS where the central edge server is located in the stationary scenario in Fig. 5a, all sessions are randomly nested within a 500x500m domain and random movement in different directions for sessions are defined in the mobile scenario in Fig. 5b.

As shown in Fig. 5a and 5b, average throughput value per session decreases as the number of sessions on the medium increases. With the number of sessions fixed after a certain point, the decrease turns into a horizontal movement. As seen from the figures, our schemes work more efficient than the RBA and APA due to split TCP sessions, session-based window size adaptations and chunk pre-fetchings. In addition, As seen in Fig. 6b, the RBA, APA and single MEC adaptations experience throughput loss in mobile scenario due to sessions having weak signal strength moving away from the cell. Yet, collaborative MEC adaptation provides a throughput rise balancing the load among edge servers via handing some sessions having weak signal strength over to other edge servers.



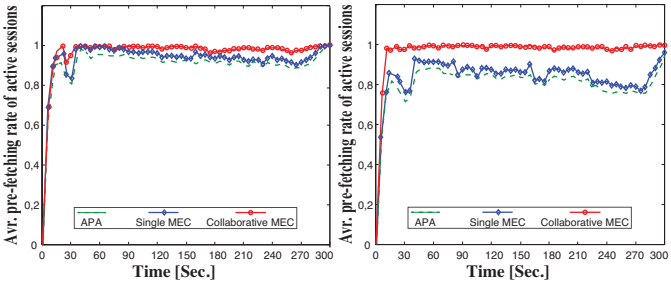
(a) Stationary UEs (b) Mobile UEs

Fig. 6: Average buffer size of active sessions

Average buffer size of sessions is shown in Fig. 6. The APA and the proposed schemes compute the number of chunks to be pre-fetched per session in the next round and saves chunks to the edge server's buffer before they are requested

by UEs. Bandwidth of the link is therefore effectively utilized and instead of UEs' bitrate reduction, these chunks are served when the medium is densely loaded. As also seen in Fig. 6, average buffer size is bigger in the collaborative MEC scheme since total number of sessions decrease due to handovers.

Fig. 7 shows the average pre-fetched response rate of active sessions . As seen from the figure, pre-fetching rate of the first 30 seconds is mostly less than pre-fetching rate of the following time periods since some of chunk transmissions are handled via backhaul traffic for new stations entering the medium in the first 30 seconds. Moreover, collaborative MEC scheme allocates more buffers to fewer number of sessions, hence achieves higher pre-fetching rate. As seen in Fig. 7b, fluctuations at pre-fetching rate are also observed with the single MEC adaptation in mobile scenario due to increased packet loss rate and adaptive bitrate adaptations.



(a) Stationary UEs (b) Mobile UEs

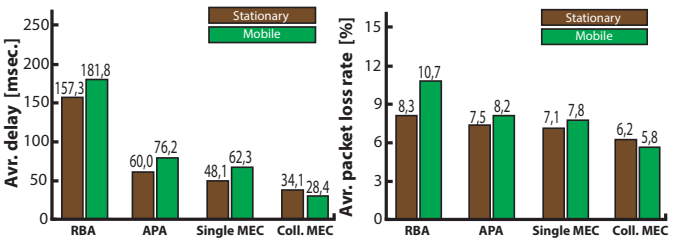
Fig. 7: Average pre-fetching rate of active sessions

C. Impact of the Proposed Scheme on QoS and QoE

Average delay and packet loss rates of sessions obtained by the four methods are shown in Fig. 8. RBA adaptation has high latency as it mostly delivers backhaul traffic without pre-fetching. In addition, since there is no TCP splitting or window size adaptation, higher packet loss rates are observed with the RBA and APA, compared to the proposed methods.

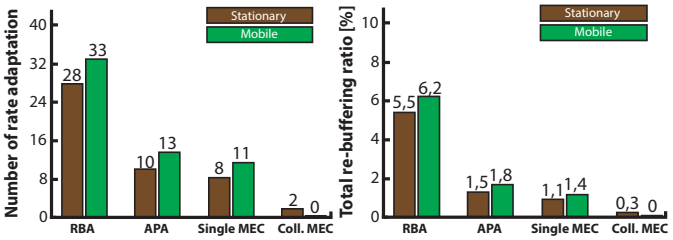
The number of rate adaptations and average re-buffering ratios of the four methods are shown in Fig. 9. As seen in Fig. 9a, APA and the proposed methods perform a much smaller number of rate adaptations compared to the RBA due to pre-fetched chunks kept in the edge server. With the collaborative MEC method, session-based average buffer size has been increased through load balancing with handover operation. In this way, bitrate adaptation has been made only two times in the stationary scenario and no bitrate adaptation has been made in the mobile scenario. Therefore, the proposed collaborative MEC scheme preserves the UE-preferred QoE level during the whole video session per user. In addition, due to the high buffer size and low bitrate adaptation, re-buffering problem experienced during the video playback is also reduced/avoided with the proposed scheme as shown in Fig. 9b.

Finally, the QoE levels obtained by the examined four video streaming methods are shown in Fig. 10. As can be seen from the figures, QoE levels obtained are lower than the video quality values determined for that bitrate due to negative impact of initial play-out gap, rate adaptation and re-buffering.



(a) Average delay (b) Average packet loss rate

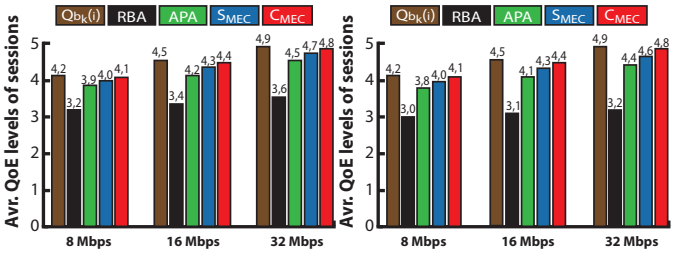
Fig. 8: Average delay and packet loss rate



(a) Number of rate adaptation (b) Re-buffering ratio

Fig. 9: Number of rate adaptation and re-buffering ratio

Yet, the proposed collaborative MEC scheme provides users with QoE levels that are very close to the video quality levels, minimizing/avoiding rate adaptation and re-buffering.



(a) Stationary UEs (b) Mobile UEs

Fig. 10: Achieved QoE values of sessions

Apart from assigning a single value for α, β, and γ; low, medium and high values were also given to these parameters utilizing the Taguchi method[7] and the effects of these values on QoE in different/variable condition such as channel status, bitrate quality and buffer level were examined. Parameter values and QoE levels of 9 scenarios examined for the stationary UEs that have 32 Mbps bitrate are shown in Table II. As seen from Table II, effect of α, β, and γ for the APA, S_{MEC} and C_{MEC} approaches is very limited because pre-fetching and handover procedures reduce end-to-end latency and prevent/limit possible channel congestion and packet losses. In contrast, significant effects of these parameters on the RBA approach (high fluctuations at the QoE level) have been observed due to high amount of channel congestion, latency and packet losses

[7]It is a process optimization approach that conducts/evaluates results of matrix experiments to determine the best levels of control factors. In this context, the parameters α, β, and γ are reassigned to be two times the size of the initial value or half the size; with low values as 1.5, 0.05, 0.01, medium values as 3, 0.1, 0.02, and high values as 6, 0.2, 0.04, respectively.

TABLE II: Parameters and QoE levels of 9 scenarios

Run No	Level P1	Level P2	Level P3	QoE RBA	QoE APA	QoE Smec	QoE Cmec
1	low	low	low	3.91	4.58	4.76	4.84
2	low	medium	medium	3.74	4.52	4.73	4.82
3	low	high	high	3.63	4.49	4.71	4.79
4	medium	low	medium	3.67	4.51	4.72	4.83
5	medium	medium	high	3.57	4.49	4.70	4.79
6	medium	high	low	3.53	4.47	4.69	4.81
7	high	low	high	3.50	4.47	4.68	4.79
8	high	medium	low	3.44	4.45	4.66	4.80
9	high	high	medium	3.32	4.42	4.63	4.77

resulting from the RBA approach. Since the experienced level of service quality is subjective per user, and users can react differently to various network issues, QoE values fluctuate when network problems are intense. Therefore, it is proper to state that instead of assigning a single value for parameters α, β, and γ, it may be more appropriate to assign user-variable ranges of values to these parameters as in Table II. Nevertheless, the main focus should be to keep the QoE value as high as possible by minimizing the volatility in these parameters, avoiding/reducing possible network issues, as in the proposed S_{MEC} and C_{MEC} approaches.

VI. Conclusion

Within the scope of the study, a collaborative QoE-based mobility-aware video streaming scheme is presented. The proposed scheme first creates a session-based QoE level, gathering/computing information from the client and medium-side, and then performs three key manipulations to preserve UE-preferred QoE level per user; (i) TCP window size adaptation, (ii) pre-fetching chunks to the edge server(s), and (iii) load balancing through handover operation. In case the QoE cannot be preserved for a session despite the aforementioned manipulations, then the proposed scheme simply adjusts the best suited video quality for that session through Adaptive Bit Rate operation based on the channel condition without harming QoE levels of other video sessions.

Throughout the paper, the proposed scheme is compared both with the conventional rate-based adaptation and the VNF-enabled pre-fetching strategy. Comparisons are made in terms of throughput, buffer size, QoS and QoE-related metrics. Achieved results validates the efficiency of the proposed scheme and illustrate that high efficiency can be achieved through the collaboration among MEC servers, utilizing explicit window size adaptation, collaborative pre-fetching and handover among edge servers. Finally, rather than a session-based optimization, we also aim to work on a network-based optimization, and propose a novel video streaming application that provide users with the highest achievable QoE for video streams over wireless/cellular networks as a future work.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, 2017.

[2] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *arXiv preprint arXiv:1612.03184*, 2016.

[3] C. V. N. Index, "Global mobile data traffic forecast update, 2015–2020," *Cisco white paper*, 2016.

[4] T. Stockhammer, "Dynamic adaptive streaming over http: standards and design principles," in *Multimedia systems*. ACM, 2011, pp. 133–144.

[5] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[6] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[7] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," in *Multimedia systems*. ACM, 2011, pp. 169–174.

[8] C. Ge and N. Wang, "Real-time QoE estimation of dash-based mobile video applications through edge computing," in *IEEE INFOCOM*. IEEE, 2018, pp. 766–771.

[9] L. Zhang, S. Wang, and R. N. Chang, "QCSS: A QoE-aware control plane for adaptive streaming service over mobile edge computing infrastructures," in *ICWS*. IEEE, 2018, pp. 139–146.

[10] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Edge computing assisted adaptive mobile video streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 787–800, 2019.

[11] J. Nightingale, P. Salva-Garcia, J. M. A. Calero, and Q. Wang, "5G-QoE: QoE modelling for ultra-hd video streaming in 5G networks," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 621–634, 2018.

[12] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Quality-adaptive prefetching for interactive branched video using http-based adaptive streaming," in *Multimedia*, ser. MM '14. ACM, 2014, pp. 317–326.

[13] S. Wilk, D. Schreiber, D. Stohr, and W. Effelsberg, "On the effectiveness of video prefetching relying on recommender systems for mobile devices," in *CCNC*, Jan 2016, pp. 429–434.

[14] K. Dong, J. He, and W. Song, "QoE-aware adaptive bitrate video streaming over mobile networks with caching proxy," in *ICNC*, Feb 2015, pp. 737–741.

[15] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Helping hand or hidden hurdle: Proxy-assisted http-based adaptive streaming performance," in *MASCOTS*, Aug 2013, pp. 182–191.

[16] C. Ge, N. Wang, G. Foster, and M. Wilson, "Towards QoE-assured 4K video-on-demand delivery through mobile edge virtualization with adaptive prefetching," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2222–2237, Oct 2017.

[17] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A mobile edge computing-assisted video delivery architecture for wireless heterogeneous networks," in *ISCC*. IEEE, 2017, pp. 534–539.

[18] R. Trestian, I.-S. Comsa, and M. F. Tuysuz, "Seamless multimedia delivery within a heterogeneous wireless networks environment: Are we there yet?" *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 945–977, 2018.

[19] M. Ivanovich, P. W. Bickerdike, and J. C. Li, "On TCP performance enhancing proxies in a wireless environment," *IEEE Communications Magazine*, vol. 46, no. 9, pp. 76–83, 2008.

[20] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *wireless networks*, vol. 1, no. 4, pp. 469–481, 1995.

[21] O. Oyman and S. Singh, "Quality of experience for http adaptive streaming services," *IEEE Communications Magazine*, vol. 50, no. 4, 2015.

[22] L. Kalampoukas, A. Varma, and K. Ramakrishnan, "Explicit window adaptation: a method to enhance TCP performance," *IEEE/ACM Transactions on networking*, vol. 10, no. 3, pp. 338–350, 2002.

[23] Y.-T. Yu, F. Bronzino, R. Fan, C. Westphal, and M. Gerla, "Congestion-aware edge caching for adaptive video streaming in information-centric networks," in *CCNC*. IEEE, 2015, pp. 588–596.

[24] M. F. Tuysuz, "An energy-efficient QoS-based network selection scheme over heterogeneous WLAN–3G networks," *Computer Networks*, vol. 75, pp. 113–133, 2014.

[25] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018.

[26] A. Virdis, G. Stea, and G. Nardini, "SimuLTE-A modular system-level simulator for LTE/LTE-A networks based on OMNeT++," in *SIMULTECH*. IEEE, 2014, pp. 59–70.

[27] A. Mehrabi, M. Siekkinen, and A. Yl-Jski, "QoE-traffic optimization through collaborative edge caching in adaptive mobile video streaming," *IEEE Access*, vol. 6, pp. 52 261–52 276, 2018.