Enhancing Mobile Video Capacity and Quality Using Rate Adaptation, RAN Caching and Processing

Hasti A. Pedersen and Sujit Dey, *Fellow, IEEE*

***Abstract*—Adaptive Bit Rate (ABR) streaming has become a popular video delivery technique, credited with improving Quality of Experience (QoE) of videos delivered on wireless networks. Recent independent research reveals video caching in the Radio Access Network (RAN) holds promise for increasing the network capacity and improving video QoE. In this paper, we investigate opportunities and challenges of combining the advantages of ABR and RAN caching to increase the video capacity and QoE of the wireless networks. While each ABR video is divided into multiple chunks that can be requested at different bit rates, a cache hit requires the presence of a specific chunk at a desired bit rate, making ABR-aware RAN caching challenging. To address this without having to cache all bit rate versions of a video, we propose adding limited processing capacity to each RAN cache. This enables transrating a higher rate version that may be available in the cache, to satisfy a request for a lower rate version, and joint caching and processing policies that leverage the backhaul, caching, and processing resources most effectively, thereby maximizing video capacity of the network. We also propose a novel rate adaptation algorithm that uses video characteristics to simultaneously change the video encoding and transmission rate. The results of extensive statistical simulations demonstrate the effectiveness of our approaches in achieving significant capacity gain over ABR or RAN caching alone, as well as other ways of enabling ABR-aware RAN caching, while improving video QoE.**

***Index Terms*—Adaptive bit rate (ABR) algorithm, video processing and caching, video quality of experience, wireless network capacity.**

I. Introduction

ADAPTIVE Bit Rate (ABR)[1] streaming has become a popular video delivery technique, improving the quality of delivered video on the Internet as well as wireless networks. Several ABR streaming techniques have been developed and deployed, among them are Apple HTTP Live Streaming (HLS), Microsoft Smooth Streaming, and Adobe Systems HTTP Dynamic Streaming [1], [2]. More recently, Dynamic Adaptive Streaming over HTTP (DASH) has been developed as a new standard for ABR with the aim to improve video Quality of Experience (QoE). In [3], we showed that video caching at the (e)NodeBs of the Radio Access Network (RAN) can increase capacity of the wireless network while improving video QoE; however, the proposed technique is not ABR-aware, and it cannot work effectively with ABR streams. In this paper, we propose techniques to effectively use both ABR and RAN caching to improve the end-to-end video capacity of cellular networks beyond what can be achieved by either ABR or RAN caching individually, while preserving the advantages in terms of QoE obtained by each of them.

Since, with ABR, each video is divided into multiple chunks, and each chunk can be requested at different bit rates, the caching problem becomes more challenging: a cache hit will require not only the presence of a specific video chunk, but also the availability of the desired video bit rate version of the chunk. Handling video chunks for caching is especially challenging for multiple reasons: Cache hit or miss can no longer be determined at video level, but must be evaluated at chunk/rate level. With ABR, having a video in the cache does not translate to having all the chunks of that video with the same rate in the cache. In addition, the decision to cache or evict a video from the cache is more complex and requires dealing with the video at chunk and rate version level. One way to solve this problem is to cache all rate variants of the video, but this approach may significantly increase backhaul bandwidth and cache storage requirements, or reduce the number of unique videos that can be cached leading to poor cache hit ratio. Alternatively, one can cache only the highest bit rate videos and use a processing resource to do rate down-conversion (transrating) for each request requiring a lower rate version, but this approach may require excessive processing or overutilize existing processing resource leading again to poor cache hit ratio.

In this paper, we propose a novel approach to support ABR while efficiently utilizing RAN caching, avoiding the problems described above. The objective of this research is to improve the video capacity (number of concurrent video requests served) of the cellular networks, while improving or maintaining video QoE. As shown in Fig. 1, instead of fetching the requested video version from the Internet CDN, we enhance the RAN caches with limited video processing capability. This will allow the possibility of transrating to a requested lower rate if a higher bit rate version is available in the cache, thus avoiding the need to cache all video bit rate versions. However, our approach does not cache only the highest bit rate versions which, as mentioned before, may exhaust the available processing resource quickly. Instead, our approach aims to cache rate versions that allow efficient utilization of both the given cache and transrating resources. For any video request, Fig. 1 illustrates the possible options with the new architecture: the exact bit rate version is available in the cache, or it can be obtained either by transrating from a higher bit rate version available in the cache, or by fetching from the Internet CDN using the backhaul. For the

Manuscript received November 28, 2013; revised November 01, 2014; accepted January 21, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Liu. Date of publication May 21, 2015; date of current version April 14, 2016. This work was supported in part by the Intel-Cisco Video Aware Wireless Networks (VAWN) program and by the UC Discovery Grant Program.

The authors are with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92092 USA (e-mail: hasti@ucsd.edu; dey@ucsd.edu).

This work has supplementary downloadable material available online at http://ieeexplore.ieee.org.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2410298

[1][Online]. Available: en.wikipedia.org/wiki/adaptive_bitrate_streaming/

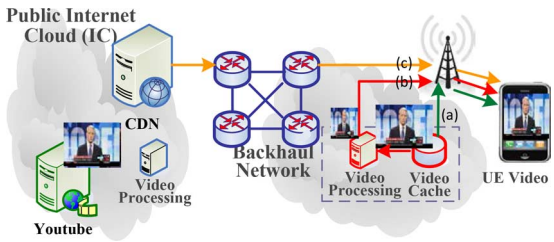

Fig. 1. Joint video caching and processing. (a) Fetch video with exact rate version from the cache. (b) Transrate from a higher rate version. (c) Fetch video from Internet CDN.

last option, either an exact bit rate version can be fetched, or a higher bit rate version can be fetched and cached which will possibly allow future requests of lower bit rate versions to be also satisfied using transrating. We develop a joint caching and processing framework which, given the available cache size, processing capability and backhaul bandwidth, enables selecting between the available options. This has two effects: it increases the number of ABR video requests that can be satisfied concurrently and improves video QoE. Based on the above framework and the popularly used Least Recently Used (LRU) caching policy [3], we develop a new ABR-aware LRU joint Caching and Processing policy (ABR-LRU-P), which we later demonstrate to be highly effective in utilizing RAN caches for ABR streaming.

In [4], we proposed a Proactive User Preference Profile based (P-UPP) caching policy, one that pre-fetches videos for caching based on the video preferences of active users in the cell, and had demonstrated its effectiveness in producing a high cache-hit ratio. To achieve high hit ratios with ABR streaming, it no longer suffices to know which videos users in a cell may like to watch. It is necessary to also have an estimate of the video bit rate that the users may request, the latter depending on the channel condition of the requesting users and the overall utilization of the network. Using rate prediction, our proposed ABR-aware P-UPP based joint caching and processing policy (ABR-P-UPP-P) aims to fetch not only the most likely requested videos by the users of a cell, but also at the most likely requested rates.

Finally, we develop a novel rate adaptation algorithm that enables the mobile devices to change the video bit rate and the required transmission rate to respond to the varying wireless channel conditions and utilization. In [3], we had used Leaky Bucket Parameters (LBPs) associated with a video to select a transmission rate that can be met by the RAN scheduler such that the video playback can start within a desired initial delay and proceed without stalling. In this paper, we extend the LBPs with the additional flexibility of multiple encoding bit rates available for the video, thus allowing the new rate adaptation algorithm to use two degrees of freedom, video transmission rate and video bit rate, to adapt to changing channel conditions. We incorporate this new ABR algorithm as part of the Video Aware Wireless Channel Scheduler (VAWS) that we proposed in [4] to improve capacity and QoE. By incorporating ABR, VAWS can now not only use different transmission rates, but also different video bit rates, providing flexibility to serve more concurrent video requests and improve users' QoE, by trading off video frame quality with risk of stalling.

We have developed a simulation framework to demonstrate the effectiveness of the proposed rate adaptation and joint caching and processing policies under different wireless channels and user distributions, as well as different cache sizes and processing capacities. The simulation results demonstrate significant increase in video capacity is possible using our approach, as opposed to using ABR or RAN caching alone, as well as compared to caching all rate versions or only highest rate version of requested video. The results also demonstrate the ability of our approaches to achieve significant capacity gain while mostly retaining the QoE advantage of ABR.

In summary, the novelty of this paper and the importance of its contributions are that it is the first to: 1) propose a rate adaptation algorithm that runs on the mobile client and uses LBP of the requested videos—thus, indirectly considering video frame structure of each video—to improve capacity and QoE; 2) address the caching challenges imposed by ABR streaming by proposing proactive and reactive ABR-capable joint caching and processing policies that leverage both transrating and cache resources available; and 3) study the impact of transrating resources at the RAN on increasing end-to-end video capacity of the cellular networks.

A. Problem Formulation

As explained earlier, the goal of this paper is to utilize RAN caching and processing to overcome the challenges imposed by ABR streaming, and to maximize the end-to-end video capacity of the cellular networks while improving QoE of the video requests under the existing video QoE and resource constraints. We formulate this objective as a variant of the knapsack problem: the dynamic and stochastic multiple, multidimensional, multichoice knapsack problem with probabilistic constraints (DSMMKP) [5], [6].

The video requests arrive one-by-one stochastically over time and can be either accepted or rejected. An accepted video request is assigned to two or more available resources in real-time—namely, wireless channel and backhaul, cache, or processing—and should satisfy certain QoE requirements. Each video has multiple bit rate versions, each satisfying the VQM requirements. An accepted video request can be assigned any of the available video bit rate versions. If a resource goes under-utilized, potential reward is lost (e.g., a cache that is not fully populated with relevant videos, or backhaul bandwidth that is not fully utilized due to lack of user arrivals). The total reward (number of concurrently accepted video requests) is a function of video request arrival rate and acceptance rate. Underutilized resources can be used for caching purposes to serve potential future video requests (e.g., by proactively preloading caches with videos that are expected to be requested in the future).

A cellular network can be thought of as a multidimensional knapsack; more specifically, a two-dimensional knapsack. Each accepted video request should satisfy both the backhaul bandwidth constraint expressed in terms of total bits-per-second, and the wireless channel power constraint expressed in terms of total Watts. The wireless channel is stochastic, and therefore the power required to serve a video request varies over time due to fading and changes in path loss due to user mobility. The backhaul is supplemented with cache and processing resources. Any video request can be fetched from the CDN through the backhaul as long as it satisfies the backhaul size constraint.

On the contrary, only videos that result in a cache hit with the exact rate version or higher rate version can be assigned to cache or processing (transrating from cache). Thus, caching the videos that result in higher future cache hits is an important decision criteria.

The rewards are equal across the videos, meaning there is no difference between accepted videos as long as the QoE constraints are satisfied. However, different rewards are assigned to each bit rate version of a video. Note that a video of longer duration and higher resolution requires more resources than a shorter and a lower resolution video, but here there is no differentiation in terms of the rewards.

The goal here is to maximize the number and quality of concurrent video requests in each time period and adapt to any change in the system in real time. Thus, one can view the cellular network as a series of multi-period, multidimensional knapsacks. Here, we present the formulation only for one of these knapsacks (only considering one time period). An accepted video request remains in the system until it finishes (across multiple time periods) unless the user aborts the video or the video request's QoE is violated. However, the optimization has the flexibility of changing the video bit rate of an accepted video request across multiple periods. The lack of complete information about future video request arrival times, size and bit rate of the requested videos, and the required resources, as well as the stochastic nature of the wireless channel, are critical factors in decision making.

Given the available resources and QoE constraints, the objective is to determine an allocation (for caching the videos) and scheduling (assigning videos to processing, RAN or CN backhaul, and wireless resources). The problem formulation is as follows:

$$\textbf{Maximize}: \quad \sum_{r=1}^{R}\sum_{j=1}^{M}\sum_{i=1}^{\psi} c_r x_{ijr}$$

$$\textbf{Subject to}:$$

$$End\ users'\ QoE\ constraints:$$

$$D_i(v_k) \le T_D \quad Q_i(v_k) \ge T_Q \quad P_{stall_i}(v_k) \le T_{stall}$$

$$Available\ Resources\ and\ their\ Constraints:$$

$$\sum_{r=1}^{R}\sum_{j=1}^{M}\sum_{i=1}^{\psi} p_{ir} x_{ijr} \le P$$

$$\sum_{r=1}^{R}\sum_{i=1}^{\psi} w_{ijr} x_{ijr} \le BW_j \quad \forall j = 1 \ldots M, M = 3$$

$$\sum_{r=1}^{R}\sum_{j=1}^{M} x_{ijr} \le 1 \quad \forall i$$

$$x_{ijr} \in \{0,1\} \quad \forall i,j,r. \tag{1}$$

In this formulation, the solution comprises the binary decision variable x_{ijr}, which is "1" if video request i with bit rate version r, is accepted and assigned to the jth resource and "0" if the video request is blocked (fourth resource constraint).

An accepted video request can be assigned only one of the available video bit-rate versions and resource types (third resource constraint). c_r is the reward associated with selecting each bit rate version of the requested video. c_r can be defined to satisfy different system requirements. For instance, to improve capacity, one must assign higher rewards to multiple video requests at lower bit rates sharing the available resources than to fewer video requests at higher bit rates. The objective function maximizes the total reward, which for the special case c_r=1 (all the video bit rates are treated equally) maximizes the number of concurrent video requests served. Selecting $c_r = 1$ can lead to undesired behaviors; e.g., (a) video bit rates of all the video requests may drop to the lowest bit rate version just to admit a video request with a very poor channel condition, (b) there is no guarantee once enough resources are available, the higher video bit rate version is preferred to the lower version. Alternatively, one can use an objective function based on Mean Opinion Score (MOS), which will strike a good balance between capacity and user experience. Typically MOS is sublinear in the bit rate due to diminishing returns of higher bit rates, and will therefore result in the desired improvement of capacity. Thus, one can select $c_r = f(b_r)$; where $f(b_r)$ reflects MOS associated with video bit rate version b_r.

ψ is the number of concurrent video requests including the latest arriving requests. Given that requests arrive stochastically over time, ψ can be thought of as the expected number of video requests in a time period. M is the multiple knapsacks available for the backhaul dimension (backhaul, cache, and processing). Each accepted video request should satisfy backhaul, cache, or processing resource constraints (second resource constraint), as well as the power constraint of the wireless channel (first resource constraint). w_{ijr} is the required bandwidth of the ith video request with rate version r for resource j. w_{ij} is infinite if $j = 3$ (cache) and the video is not cached, or if $j = 2$ (transrating) and there are no available transrating opportunities (higher rate version is not in the cache). P is the total transmit power available at the (e)NodeB. Total bandwidth of the videos that are assigned to backhaul, cache, or processing should be less than or equal to the backhaul capacity BW_1, transrating capacity BW_2, and cache capacity BW_3 respectively (second resource constraint). Here, we define BW_2 in terms of the number of encoded bits that can be processed per second. For instance, a RAN processing capacity of 20 Mbps allows transrating of ten concurrent video requests with video bit rates of 2 Mbps each. $D_i(v_k)$, $Q_i(v_k)$ and $P_{stall_i}(v_k)$ are the initial buffering delay, video quality, and stalling probability respectively experienced by the ith video request for video v_k. $D_i(v_k)$ should be smaller than T_D seconds, the maximum tolerable initial delay before start of playback; $Q_i(v_k)$ should be greater than T_Q, defined in terms of complement of VQM (VQM spans from "1" (best) to "0" (worst) quality), and $P_{\mathrm{stall}_i}(v_k)$ should be smaller than acceptable threshold of T_{stall}.

Since solving the above capacity maximization problem is NP-hard in the strong sense (refer to Appendix C in the supplementary material available online for proof), we propose an approach that consists of solving two subproblems as described before: 1) client-side adaptive video rate selection in coordination with VAWS [4] to maximize the number of concurrent video requests that can be served by a RAN scheduler through the wireless channel, in addition to meeting/improving video QoE, which is the target of conventional ABR techniques and 2) ABR-aware reactive and proactive joint caching and processing policies to maximum the video capacity given the available resources and their constraints. Next, we describe the related work on each of the two subproblems that we explain above.

B. Related Work and Paper Outline

There has been much research, development, and commercialization of ABR streaming. Some of the adopted standards include Apple HTTP Live Streaming, Microsoft Smooth Streaming, Adobe OSMF, and DASH. The rate adaptation algorithm—where the desired video rate is selected depending on the available bandwidth—is a critical part of any ABR streaming implementation including the ones listed above. Although ABR rate selection algorithm is not being standardized, there has been much research done on this topic. [7] proposed a machine learning technique for predicting TCP throughput for video rate adaptation in addition to using feedback from the client video playback buffer. [8] presented a rate adaptation algorithm based on their proposed smoothed HTTP/TCP throughput, instead of using instantaneous TCP throughput. [9] studied the bandwidth consumption of the Netflix video service as an example implementation of DASH protocol, and discussed the impact of TCP rate control algorithm and its interaction with application layer rate adaptation and control.

The above rate selection techniques, however, focus on minimizing video stalls, and do not consider maximizing video capacity which, in addition to minimizing stall, is our target. Moreover, none of the current techniques consider video characteristic like LBPs in making rate change decisions, while our proposed rate adaptation algorithm uses individual video characteristics by using extended E-LBP tables to make the rate change decisions. Finally, the current policies only change the video bit rate; however, using E-LBP table that we explain later in this paper, we not only change the video bit rate, but also the required transmission rate. Therefore, our ABR algorithm enables RAN schedulers like VAWS [4] to satisfy more requests, hence increasing capacity, and improving video QoE by eliminating or decreasing the probability of stalling. Next, we explain the limited research currently done on ABR-aware caching.

The authors of [10] proposed a cloud-assisted ABR, and a technique that prefetches videos based on social associations of the users, caching those videos at the mobile device to provide a better video QoE for the video users. However, their proposed framework was mostly focused around Scalable Video Coded (SVC) contents and does not address the challenge of caching multiple rate versions of the same video. Furthermore, they consider caching at the mobile device that proclaims different requirements than the shared caches at the (e)NodeBs that we target in this paper. The work in [11] and [12] studied the problems associated with ABR caching, but for CDNs which have very large caches, as opposed to the relatively small RAN caches that we target. We had earlier demonstrated the CDN caching policies are not effective for RAN caching [3]. Moreover, they did not propose any algorithms to address the challenges of ABR capable caching.

The work in [13] addresses the problem of deciding which bit rate versions to cache, assuming the probability of rates requested by mobile users (user request model) is known. However, in mobile networks it is impossible to know ABR rate probabilities *a priori* as they vary dynamically depending on many channel and usage factors as shown in Section IV. Video selection is a key determinant for performance of small sized RAN caches; however, [13] does not propose a solution for selecting which videos to cache, but focuses on choosing the rate versions to be cached. Furthermore, [13] only considers constrained storage, while we consider constraints of all the resources involved in end-to-end mobile video delivery, namely backhaul network, cache, processing and wireless channel.

We believe none of the previous publications address the implications of ABR caching for the relatively small caches at the RAN (e)NodeBs. In addition, we are not aware of any previous publication that considers limited video processing resources to aid caching in the (e)NodeBs, and joint caching and processing techniques to maximize capacity given the backhaul, cache, and processing capacity constraints.

The remainder of the paper is organized as follows: In Section II, we describe our LBP-based video rate adaptation algorithm that increases wireless channel capacity while resulting in no or limited stalling during video playback. Subsequently, in Section III, we propose ABR-aware reactive and proactive joint video caching and processing policies that aim to increase video capacity. Finally, in Section IV, we outline our simulation framework, and demonstrate the effectiveness of our approaches in achieving significant capacity gain over alternative methods discussed earlier. We conclude the paper in Section V. In Appendix A,[2] we study the time complexity of the proposed joint caching and processing algorithms. The probabilistic constraint of the ABR-P-UPP-P caching policy is detailed in Appendix B. Finally, we present the proof of NP-hardness of the problem formulation in Appendix C.

II. ABR Rate Selection Algorithm

Conventional ABR algorithms enable mobile devices to request an appropriate bit rate version of the video to avoid buffer underflow, thus avoiding stalling in varying wireless network conditions. In this subsection, we introduce a new client-centric ABR algorithm, which uses video frame characteristic through E-LBP table for rate selection. Using this ABR algorithm, the mobile device is able to change the video bit rate and the required transmission rate not only to avoid stalls, but also to increase video capacity of the network. We call our joint adaptive bit rate and transmission algorithm Bit Rate and Transmission Rate Selection algorithm (BiTRaS).

In [3], we had introduced the use of video LBPs [14] for the UE to request a transmission rate that results in an initial buffering delay just below a desired threshold, T_D, set according to the QoE requirements of the user or video service. LBPs consist of L 3-tuples (R, B, F) corresponding to L sets of transmission rates and buffer size parameters for a given bit stream [14]. An LBP tuple guarantees that as long as the minimum transmission rate $(R_{\min})$ is maintained at R bps, the client has a buffer size of B bits, and the buffer is initially filled with F bits before video playback starts, the video session will proceed without any stalling. Since it may not always be possible to sustain a required minimum transmission rate in the wireless channel, the video session may still end up experiencing stalling, as reported in [4]. We had proposed a backhaul scheduler and Video Aware Wireless Channel Scheduler (VAWS) [4] that allocate backhaul

[2]The appendices of this paper are available as supplementary downloadable material (online appendix) at http://ieeexplore.ieee.org, provided by the authors. Any reference to appendices in this paper refers to the online appendix.

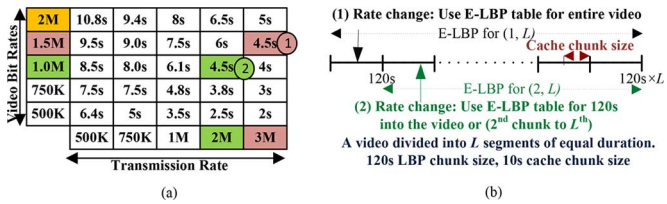

Fig. 2. (a) E-LBP table: transmission rate, video rate, and delay. (b) Example illustrating usage of E-LBP segments by BiTRaS.

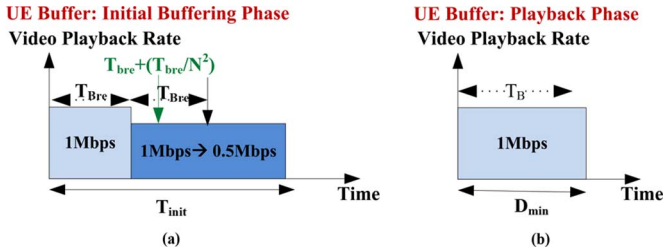

Fig. 3. Video bit-rate timelines.

bandwidth and wireless channel resources (e.g., power and subcarriers in LTE) to video requests to satisfy their minimum required transmission rate (selected from video LBP table). In this paper, we leverage the additional flexibility of adapting video bit rates, besides adapting transmission rates done in [3] and [4], to further enhance video capacity while satisfying initial delay and further reducing stalling. We do so by: 1) enhancing the LBP table to an Extended LBP (E-LBP) table, with an added dimension of video bit rates besides transmission rates and 2) developing a new adaptive rate algorithm (BiTRaS) which utilizes the E-LBP table to select the optimal video bit rate and required transmission rate. Note that since BiTRaS runs on the mobile device and integrated with the video client, we use the terms BiTRaS and video client interchangeably. In this paper, we use backhaul and VAWS schedulers that we proposed in [4] to allocate the transmission rate requested by the video client.

Fig. 2(a) shows an example E-LBP table where columns represent different transmission rates, rows represent the available video bit rates, and the values in the cells represent the initial buffering period, F/R, to guarantee playback without stalling, provided the transmission rate is delivered consistently for the corresponding video bit rate. As BiTRaS can initiate a rate change anytime during the video playback, we divide each video into several segments (each segment consisting of multiple chunks) and generate L E-LBP tables; each covering the l-to-Lth segment, where $l = 1, \ldots, L$. Fig. 2(b) shows an example, where each segment is 120 s. If a video client requests a rate change [marked by (1)] during segment 1 (anytime from beginning to 120 s into the video), BiTRaS will use the E-LBP table for the entire video. On the other hand, if video rate change occurs [marked by (2)] during segment 2 (from 120 s to 240 s), the video client will use the E-LBP table that is generated from 120 s to the end of the video or from the second segment to the Lth segment.

From the example E-LBP table shown in Fig. 2(a), we can see how BiTRaS will be able to leverage the flexibility of adapting video bit rates (R_v) and transmission rates (R_t) simultaneously either to improve QoE or assist VAWS to improve capacity. BiTRaS uses the E-LBP table to identify whether the video buffer is in danger of under-flowing and hence stalling. If the achieved rate (R_a) to a mobile device falls below the requested rate (R_t) selected by BiTRaS earlier, BiTRaS may switch to a lower video bit rate to avoid stalling. As frequent video bit rate changes degrade video QoE (according to subjective study performed by [15]), BiTRaS uses different thresholds to avoid constant rate changes, while still having enough time to play back the new video bit rate without stalling. We explain these thresholds next and present the algorithm.

Fig. 3 shows an example timeline and thresholds we use to change the video bit rate to a lower value during initial buffering and playback phase. In the initial buffering phase [Fig. 3(a)], the UE monitors its buffer every T_{Bre} seconds, starting at time T_{Bre}; if actual number of bits received (B_r) by time T_{Bre} is below the expected number of bits (B_e), the UE switches to a lower video bit rate. Selecting an appropriate value for T_{Bre} is important because if T_{Bre} is selected too early in the T_{init} sequence, it may lead to aggressive and unnecessary rate changes due to fluctuating channel conditions. However, selecting T_{Bre} too close to T_{init}, may leave BiTRaS with too little time to switch to a new video bit rate; possibly resulting in multilevel video bit rate changes which is not desirable or/and unnecessary stalling. Thus, the video client monitors the buffer status starting from $T_{Bre} = \frac{T_{\text{init}}}{N}$, where N is a design parameter above 1 that determines the aggressiveness of the algorithm. For instance, for N $= 3$ and $T_{init} = 4.5$ s, T_{Bre} is equal to 1.5 s. From the E-LBP in Fig. 2(a), assuming that the video client has selected $R_v = 1.5$ Mbps and $R_t = 3$ Mbps [marked by (1)], the client expects receiving $B_e = T_{Bre} \times R_t = 4.5$ Mbit of video bits at T_{Bre}. If the video client has received more than 4.5 Mbit, it continues to monitor the buffer at 1.5-s intervals. If $B_e < 4.5$ Mbit, the client buffer may not reach the desired level for starting the playback on time. The client uses B_r to estimate the actual achieved transmission rate (R_a) and subsequently divides the next 1.5-s interval into subintervals of $\frac{T_{\text{init}}}{N^2}$ at which the buffer level is monitored. If the buffer level remains below B_e, BiTRaS triggers a rate change. It uses the E-LBP table to select a new transmission rate R'_t equal to or lower than R_a, and a new video bit rate that can satisfy the new initial delay, which is the time left until playing back the new rate $(T_{B,\text{init}} = T_{\text{init}} - T_{Bre} + \frac{B_c}{R_v})$, where B_c is the current buffer fullness.

During the playback phase [Fig. 3(b)], if the UE buffer falls below a certain playback time, $D_{\min}$, BiTRaS switches to a lower video rate (we explain $D_{\min}$ later in this section). The UE uses the remaining playback buffer duration, T_B, for the old video bit rate as the target initial delay when looking up the required transmission rate for the new video bit rate in the E-LBP table. Following the same approach, if the UE buffer exceeds $B_{\max}$ (maximum buffer size from LBP table), the UE switches to a higher video bit rate. To come up with $D_{\min}$, we make a slight modification to our LBP generation process explained in [4]. In generating LBP, here the goal is to find the initial fullness level F for a D/G/1 queue, with input rate R_t and output rate R_v, so that the playback buffer level never falls below a minimum level, corresponding to having $D_{\min}$ seconds play time left in the buffer.

When selecting the new video bit rate from the E-LBP table, either switching to a higher or lower rate, if possible BiTRaS selects the next video bit rate that is one step above or below the current video bit rate which agrees with the subjective QoE

results [15] that states that gradual changes are preferred by users over more abrupt changes. From our earlier example, if R_a drops to 2 Mbps from 3 Mbps and the remaining time for the rate change is 4.5 s, BiTRaS selects $R_v = 1$ Mbps, which is one rate below R_v of 2 Mbps [marked by (2) in Fig. 2(a)]. However, if R_a would drop to 1 Mbps or less, one step transition is no longer possible, as bit rate of 500 kbps will have to be selected to satisfy the new initial delay of 4.5 s.

Fig. 4 details the BiTRaS algorithm. For each video request, BiTRaS starts by selecting the default video bit rate (here, highest video bit rate available) and requests the lowest transmission rate $R_{\min}$ which results in an initial delay T_{init} that satisfies the user's maximum acceptable initial delay requirement, T_D, from the E-LBP table (line 2). Next, BiTRaS resets the timers T_{Bre} and T_{Adapt} (time of last rate adaptation) (line 3). If the video client is in initial buffering phase, BiTRaS monitors the buffer levels at T_{Bre} (lines 5–7). If the buffer is not in danger of underflow, it continues to monitor the buffer level at regular intervals as explained earlier (line 8). If the UE's buffer is in danger of underflow, BiTRaS shortens the monitoring interval to $\frac{T_{\text{init}}}{N^2}$ (line 19), and if for another interval, the UE buffer is still in danger of underflow, BiTRaS calculates R_a and $T_{B,\text{init}}$. From the E-LBP table, BiTRaS selects the highest video bit rate, R_v, for which $R_t \leq R_a$ and $T_{\text{init}} \leq T_{B,\text{init}}$. The goal is to achieve a one-step video bit rate transition if possible (line 13). If no video bit rate is found that satisfies R_a and $T_{B,\text{init}}$, BiTRaS cancels the download (line 17). If the video client is in playback phase, BiTRaS monitors the buffer level (line 23) and if the buffer is in danger of underflow or overflow, and sufficient time has passed since the last rate change to allow the buffer to return to its desired level, BiTRaS selects the video bit rate from the E-LBP table with initial delay and new transmission rate of at most T_B and R_a respectively (lines 22–24). If there exists such a video bit rate, it effectuates the rate change and calculates T_{Adapt} (line 26). Otherwise, if the UE buffer is stalling more than T_{stall} seconds and no video bit rate is available from E-LBP table to allow seamless rate change, BiTRaS cancels the video (line 28). This is in line with subjective testing results reported in [16] which suggest that users terminate the videos due to many or long stalls.

E-LBP based Adaptive Rate Algorithm

1. **For** each new video request v_i
2. Using E-LBP, select $R_t = minR$ such that $T_{init} \leq T_D$
3. Reset timers $t = 0$, $T_{Bre} = \frac{T_{init}}{N}$, $T_{Adapt} = 0$, $k = 0$
4. **While** download is ongoing
5. **If** video client in initial buffering phase $\cap$ $t > T_{Bre}$
6. Calculate B_e and B_r at time t
7. **If** $B_r \geq B_e$ (no indication of underflow)
8. $T_{Bre} = T_{Bre} + \frac{T_{init}}{N}$
9. **Else**
10. Video client may not be able to start the playback on time
11. **If** $k \geq 1$
12. Set: $R_a = B_r/t$, $T_{B,init} = T_{init} - T_{Bre} + B_c/R_v$
13. Find max R'_v, s.t. $R'_t \leq R_a \cap T'_{init} \leq T_{B,init}$
14. **If** $R'_v \neq \emptyset$
15. Change the video bit Rate: set $R_v = R'_v$, $R_t = R'_t$, $T_{init} = T'_{init}$, $T_{Adapt} = t$
16. **Else**
17. Cancel download, block request
18. **End If; End If;**
19. $T_{Bre} = T_{Bre} + \frac{T_{init}}{N^2}$, $k = k + 1$
20. **End If;**
21. **Else If** video client in playback phase
22. $T_B = B_c/R_v$ (B_c: current Buffer Fullness)
23. **If** $(T_B < D_{min} || T_B > B_{max}) \cap (t \geq T_{Adapt})$
24. Given R_a and T_B find R'_v from E-LBP table such that $R'_t \leq R_a$ and $T'_{init} \leq T_B$
25. **If** $R'_v \neq \emptyset$
26. Request new $R_v = R'_v$, $R_t = R'_t$, $T_{init} = T'_{init}$, $T_{Adapt} = t + \frac{(D_{min} - T_B)R_v}{R_t - R_v}$
27. **Else If** $R'_v = \emptyset \cap$ video in stall for more than T_{stall}
28. Block the video session
29. **End If; End If; End If**
30. **End While, End For**

Fig. 4. E-LBP-based adaptive rate algorithm.

III. Joint Video Caching and Processing

Here, we outline our reactive and proactive joint caching and video processing framework.

ABR streaming imposes an extra challenge on caching as it no longer suffices for a video to be in the cache but the video should be in cache at the requested bit rate. This is further complicated by the fact that each video is divided into multiple chunks, each of which can be requested at different bit rates. Even if the entire video is cached at the rate originally requested, the desired bit rate may change over time so that after a rate change, the desired rate is no longer available in the cache. To help alleviate these challenges, we propose adding processing/transrating resources to each RAN cache which can be used to transrate a higher bit rate version of the chunk that maybe available in the cache to a required lower rate version, thereby relieving the need to fetch and possibly cache the lower rate version. Like the backhaul resource, transrating is quantified in terms of transmission rate, and needs to be $R_{\min}$ or higher in order to meet the requirements determined by the LBP parameters. Furthermore, when making caching decisions, we cache the videos so that not only the cache hit ratio is increased (BW_3 in Section I-A), but also the number of video rate versions that can be served by transrating from the cached rate versions is increased. However, it may not always be optimal to use transrating when a higher bit rate version of a video is in the cache. Additionally, it might be better to use the backhaul to fetch the video with the exact rate version from the CDN. As an example, consider a scenario with available transrating capacity of 2 Mbps, backhaul bandwidth of 2.3 Mbps and sequential video request arrivals, that can be served either using transrating or through the backhaul with required rate of 800 kbps, 2 Mbps, and 1.5 Mbps. If we greedily assign the first request (800 kbps) to transrating, then we need to fetch the second video (2 Mbps) using the backhaul. In this case, the third video request (1.5 Mbps) cannot be served, due to lack of backhaul and transrating resources. However, by assigning 800 kbps request to the backhaul, and 2 Mbps to transrating, we can admit the 1.5 Mbps by assigning it to the backhaul.

The above example illustrates the need for a resource allocation scheme for incoming video requests whose exact rate versions cannot be found in the cache but a higher rate version can. Next, we formulate the above problem as an optimization that aims to maximize the number of requests that can be served concurrently. Later, we explain our proposed ABR-LRU-P and ABR-P-UPP-P joint caching and processing policies.

A. Resource Allocation for Joint Caching and Video Processing Policies

As shown in Fig. 1, with the proposed joint caching and processing architecture, when BiTRaS or any other ABR algorithm changes video bit rate, either of the following scenarios may

apply: 1) the video chunks are available in the cache with the exact requested rate; 2) the video chunks are in the cache but with a higher rate version; or 3) there are video chunks with higher rate version in the UE's (e)NodeB buffer pending transmission to the UE. If the video is not available at the desired or higher bit rates (none of the above), it has to be fetched from CDN using the backhaul. For scenarios 2) or 3), the video request can be satisfied either by transrating it to the desired video bitrate using processing resource, or fetching it at the desired video bitrate using the backhaul.

For this subset of requests that we call "T", we propose a multiple knapsack formulation that allocates resources (backhaul and processing) to increase the number of videos served by maximizing objective function $f(T)$. $f(T)$ is maximized under the constraint that w_{i1r} or w_{i2r} (problem formulation) is set to $R_{\min}^{I_{ir}(v_k)}$ (the minimum required transmission rate from E-LBP), and the sum of the assigned transrating or backhaul bandwidth of all scheduled video requests assigned to each resource is less than or equal to the capacity of resource BW_j, $j = 1$ (backhaul), or 2 (transrating):

$$\begin{aligned} \textbf{Maximize}: \quad & \sum_{j=1}^{2}\sum_{i=1}^{T}\alpha_j x_{ij} \\ \textbf{Subject to}: \quad & \sum_{i=1}^{T} R_{\min}^{I_{ir}(v_k)} x_{ij} \le BW_j, \forall j = 1,2 \\ & x_{i1} + x_{i2} \le 1, \forall i \\ & x_{ij} \in \{0,1\}. \end{aligned} \tag{2}$$

x_{ij}s are the decision variables of the maximization problem. By definition, x_{i1} and x_{i2} are mutually exclusive - i.e., if $x_{i1} = 1$ then $x_{i2} = 0$ and vice versa. $x_{i1} = 1$ indicates video request i is assigned to backhaul resource; $x_{i2} = 1$ indicates i is assigned to transrating resource. If neither backhaul nor transrating is chosen, both x_{i1} and x_{i2} are zero.

In the above formulation, α_1 and α_2 are set to a weight factor multiplied by the complement of the backhaul utilization, $w_{bk}(1 - U_{bk})$, and transrating utilization, $w_{tr}\ (1 - U_{tr})$ respectively. w_{bk} and w_{tr} are weights used to bias towards using transrating resource if utilization of the backhaul and transrating is the same. The goal is to assign more requests to the transrating resource, as only a portion of the video requests can be served by transrating (videos with higher rate version in the cache). Meanwhile, backhaul can be used by any request and hence should be preserved for future requests that are not candidates for transrating. Furthermore, the formulation ensures that the less utilized a resource is, the more likely it is assigned to serve a new video request.

Given the selected objective function and the constraints, we can see that our formulation has the canonical form of a binary integer program, for which the solution has been shown to be NP-complete [17]. Hence, we solve the linear relaxation of this binary integer program, relaxing the constraints to $0 \le x_{ij} \le 1$. Later, we round the linear solutions to integer values, following a rounding method similar to the one described in Section III-C2.

Fig. 5 shows the resource allocation for videos in T. First, we calculate the backhaul and transrating utilization given $R_{\min}$ of all the ongoing video requests allocated to each resource (line 1).

Resource Allocation Algorithm

F_i: set of scheduled video requests assigned to backhaul or processing;
F_{i1}: video requests assigned to backhaul;
F_{i2}: video requests assigned to transrating;
F_k: set of all newly arrived video requests.
1. Calculate U_{bk} and U_{tr} based on R_{min} of currently assigned requests to each resource; $0 \le U_{bk} \le 1; 0 \le U_{tr} \le 1$;
2. Calculate $\alpha_1 = w_{bk}(1 - U_{bk})$, $\alpha_2 = w_{tr}(1 - U_{tr})$; $w_{tr} \ge w_{bk}$
3. **For** $\forall i \in F_k; |F_k| = K$
4. Solve maximize$(\sum_{j=1}^{2}\sum_{i=1}^{k}\alpha_j x_{ij})$ for x_{ij}; while $\sum_i R_{min}^{I_{ir}(v_k)} x_{i1} \le BW_1 - \sum_{i\in F_{i_1}} R_{min}^{I_{ir}(v_k)}$ and $\sum_i R_{min}^{I_{ir}(v_k)} x_{i2} \le BW_2 - \sum_{i\in F_{i_2}} R_{min}^{I_{ir}(v_k)}$
5. **If** $\exists i$ s.t. $x_{i1} = 0 \cap x_{i2} = 0$ (video request is not admitted)
6. Solve the optimization across all the video requests: $F_i \cup F_k$
7. **If** better solution is achieved by solving for all video requests;
8. Select the new solution and reassign resources accordingly
9. **End If; End If; End For**

Fig. 5. Resource allocation algorithm.

Utilization of a resource is simply the ratio of the capacity of resource currently allocated divided by the total available capacity of the resource. Subsequently, we calculate α_j for $j = 1, 2$ and introduce two multiplicative factors w_{bk} and w_{tr}, where $w_{tr} \ge w_{bk}$, to allocate more resources to transrating than to backhaul given the same utilization level (line 2). For all of the new video requests, we maximize the total number of videos going through the backhaul or being transrated using the formulation in (2) (line 4). As changing the allocation of the admitted requests is not desirable unless a better allocation can be achieved, we solve (2) only for the newly arrived video requests. If there is a video request that cannot be admitted (line 5), we solve the optimization problem (2) across all video requests (currently served and new) that can be either scheduled through the backhaul or using transrating. If more requests can be served (more instances where $x_{i1} \ne 0$ or $x_{i2} \ne 0$) by redistributing video requests across different resources, the new resource allocation is used (lines 5–9).

B. ABR-LRU-P Joint Caching and Video Processing Policy

In this section, we enhance the Least Recently Used (LRU) [3] caching policy to efficiently serve ABR video requests from (e)NodeB caches, utilizing the available (e)NodeB processing and backhaul resources optimally to increase video capacity of the wireless networks.

LRU [3] is a reactive caching policy that fetches the video from the Internet CDN upon user request if it is not already found in the cache. It then subsequently caches the content while simultaneously evicting entries in the cache that have been least recently used to free up space for the requested content. Our proposed ABR-LRU-P is a variant of the LRU that allows a video with different bit rate representations to be cached. Additionally, it allows for different chunk sizes (e.g., dividing the video into chunks of 1 s, 10 s, or multiples of the GoP size), to be cached at the currently requested bitrate. On eviction of a video from the cache, ABR-LRU-P removes the video that has been least recently used. If multiple rates of the same video exist, it selects the rate that has been least recently used for eviction. ABR-LRU-P starts from the last chunk of the video and evicts until there is enough cache space available for the chunk or chunks of the video that is to be cached.

One of the challenges for ABR-LRU-P is that a cache hit does not necessarily translate to finding all the chunks of a video with the desired bit rate in the cache. In other words, a cache hit for a chunk of a video may not necessarily translate to a cache hit for

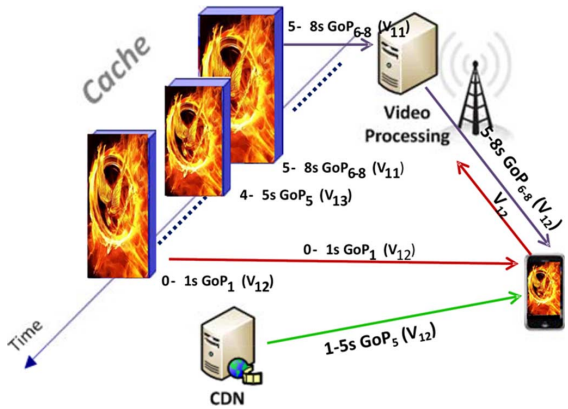

Fig. 6. Example scenario showing how video requests are satisfied in the proposed ABR capable RAN caching and processing approach.

the next chunk, as different chunks of a video can be cached at different bit rates. For instance, as shown in Fig. 6, a UE requests video 1 with the 2nd highest bit rate, v_{12}; there is an instance of the video in the cache with the desired bit rate for the first second; however, video chunks that correspond to 1–4 s of video playback are not in the cache and chunks corresponding to 4–5s of playback are cached in 3rd available rate of the video (lower bit rate than v_{12}). Therefore, the video chunks from 1 to 5 s need to be brought in from the backhaul. The remainder of the video chunks exist in the cache with the 1st (highest) available bit rate (5–8 s of v_{11}), so using the resource allocation algorithm discussed in Section III-A, we can either transrate the video bit rate to the desired rate or bring the video using backhaul. If the video cannot be admitted, we use the video available in the cache and serve the mobile device with a higher bit rate than it can support, risking stalling during playback.

Fig. 7 shows the ABR-LRU-P policy. In the event of a new video request for v_{ir} (line 1), we start from the first chunk that is to be transferred to the UE (lines 2–3); if the chunk with the requested rate is in the cache, we bring the chunk from the cache and update the access time of the chunk (lines 5–7). If the chunk is cached at a higher rate version than the one requested, use resource allocation algorithm (Fig. 5) to optimally allocate the video to either transrating or backhaul. If we bring the chunk through the backhaul, we cache the chunk. Otherwise, if we use transrating resource, we update the original chunk access time (lines 8–14). If the video is not a cache hit with a higher rate version, fetch the chunk through the backhaul and cache the chunk according to the LRU caching policy (15–18). If neither backhaul bandwidth nor a higher rate chunk is available in the cache or (e)NodeB buffer, UE buffer may be in danger of underflow and BiTRaS as we explained in Section II, may trigger a rate change.

C. Proactive User Preference Profile Caching Policy

In [3], we proposed a caching policy, termed P-UPP, that proactively caches videos according to the User Preference Profiles (UPP) of active video users in the (e)NodeB, demonstratively increasing the cache hit ratio of the (e)NodeB caches and the end-to-end video capacity. However, with ABR streaming, a video request cannot be served from the cache unless the right bit rate version is available. Hence, to be successful, P-UPP

ABR-LRU-P caching policy

1. **For** each video request v_{ir} (either a new request or rate change)
2. $c_k = 0$ (current chunk of v_{ir})
3. S: UE buffer at (e)NodeB that contains video chunks that are pending transfer to UE
4. **While** video download is ongoing
5. **If** chunk c_k is cached or in S
6. Transfer chunk c_k from S or cache to the UE
7. update the access time of the chunk (according to LRU policy)
8. **Else If** chunks of higher video rate in the cache or S
9. Use resource allocation algorithm (Fig. 5) to decide whether to transrate or use backhaul
10. **If** transrating
11. update the access time of the video
12. **Else If** using backhaul
13. follow LRU eviction policy and cache c_k
14. **End If**
15. **Else If** enough backhaul bandwidth available
16. Bring chunk c_k from the backhaul with the requested rate
17. Follow LRU caching policy for eviction
18. Cache chunk c_k
19. **Else**
20. Client's buffer is in danger of underflow. BiTRaS algorithm will possibly trigger a rate change.
21. **End If**
22. $c_k = c_k + 1$
23. **End While, End For**

Fig. 7. ABR-LRU-P caching policy.

has to be able to estimate not only which videos will be requested (which it successfully does using the UPPs of the active users), but also what bit rate versions will be requested. In this section, we first explain our approach to predict what video bit rates will be requested by the mobile ABR clients. Next, we propose an ABR-aware P-UPP joint caching and processing (ABR-P-UPP-P) policy, which uses the rate prediction technique and user UPPs to proactively cache videos with the appropriate rate versions, while also maximally utilizing the processing resource at the (e)NodeB to transrate as needed to achieve high cache hit ratio for ABR video requests.

1) Video Bit Rate Prediction Algorithm: In this subsection, we discuss how to predict what video bit rate may be used by the BiTRaS clients (mobile devices) for the subsequent requests. The per-user achieved rate within an (e)NodeB depends on the utilization of the (e)NodeB and the user channel condition. Note that user's achieved rate, R_a, is the actual rate given to the user by the video aware scheduler and it might be different from the requested rate as in some circumstances network cannot sustain the requested rate. Different techniques can be potentially used to predict the achieved rate of a user, like SINR measured by mobile devices and reported back to (e)NodeBs, or by monitoring status of the UE's buffer. However, even if these estimation methods are accurate, the achieved throughput of a user may include other data traffic, like that from other applications running in the background, and it may be difficult to predict the throughput due to the video request itself.

Hence, in this paper we develop a technique to estimate the request probability of a video bit rate by looking at the weighted bit rate versions of the ongoing chunk downloads and previous estimates. To predict the probability that video bit rate r is being requested at time t, $R(t,r)$, we use the exponential moving average (an IIR filter) as follows:

$$R(t,r) = (1-\beta)R(t-1,r) + \beta R_{inst}(r)$$
$$0 < \beta < 1 \quad 1 \le r \le |R| \tag{3}$$

where β is the smoothing factor, $R(t,r)$ is the request probability of the rth video bit rate at time t. The smoothed video

rate probability $R(t,r)$ is a simple weighted average of the current observation R_{inst} and the previous smoothed statistic $R(t-1,r)$. R_{inst} is the distribution of selected video bit rate, r. Larger values of β reduces the level of smoothing, and in the limiting case with $\beta = 1$ the output is the same as the instantaneous rate (with lag of one time unit). Note that other sophisticated estimation methods can be used instead of the exponential moving average method to infer the requested video bit rate [18].[3]

The rate prediction is performed along with the caching algorithms; it keeps track of the number of video requests with a given source video bit rate in each time interval, and using (3) calculates distribution of the video rates for the next time interval. Later in the simulation result section, we validate our prediction method using null hypothesis testing.[4] Next, we explain our ABR-P-UPP-P joint caching and video processing policy that utilizes the predicted video bit rate as well as cell site UPP to make caching decisions.

2) ABR-P-UPP-P Joint Caching and Video Processing Policy: In this subsection, we explain our ABR-P-UPP-P joint caching and processing policy that uses UPPs of active users in the cell, and their video request rate prediction, to cache videos with the rates that are most likely to be requested at each (e)NodeB. Like in [3], we assume that we know the UPP of the users in the cell and we can infer the requested video bit rate through a set of measured bit rates as explained in Section III-C1.

In [3], we calculated the probability that a video will be requested by the active user set (AUS) of an (e)NodeB as follows:

$$P_R(v_i) = \sum_{k=1}^{|VC|} p(v_{i,k}) p_{\mathrm{AUS}}(vc_k). \tag{4}$$

$P_R(v_i)$ is the probability that video i is requested, $p_{AUS}(vc_k)$ is the probability that AUS requests video category vc_k, $|VC|$ is the total number of video categories and $p(v_{i,k})$ is the request probability of video i with video category k. $p_{\mathrm{AUS}}(vc_k)$, is the weighted sum of probabilities that vc_k is being selected by each user in the AUS, and is given by

$$p_{\mathrm{AUS}}(vc_k) = \sum_{j=1}^{|U|} p(u_j) p(vc_k|u_j). \tag{5}$$

In the above equation, $|U|$ is the cardinality of AUS (number of active users), and $p(u_j)$ is the probability that user, u_j, requests a video. For detailed description of how the above terms are calculated, please refer to [3]. To calculate the probability $P_R(v_{ir})$ that a specific rate r of a video i is requested, we use the following formulation:

$$P_R(v_{ir}) = R(r) \times P_R(v_i) \tag{6}$$

where $R(r)$ is the probability that rate r is requested and calculated using (3) (time t dropped for simplicity of notation). To maximize the cache hit ratio of future requests (increase BW_3 in the problem formulation), our ABR-aware P-UPP policy can proactively cache videos with highest $P_R(v_{ir})$. However, another objective should be to maximally utilize the transrating resource available to satisfy future requests for different bit rate versions than the ones available in the cache. Thus, our proposed ABR-P-UPP-P policy aims to not only identify the set of videos with rates that result in higher cache hit ratio, but also to cache videos and rate versions that can be used later by the transrating resource available at the (e)NodeB to satisfy requests of lower bit rate versions. With the above dual objectives in mind, we formulate and solve the proactive caching problem as an Integer Linear Program as follows:

$$\begin{aligned}
\textbf{Maximize}: \quad & \sum_{i=1}^{|V|}\sum_{r=1}^{R} \alpha_{ir} P_R(v_{ir}) + \sum_{i=1}^{|V|}\sum_{r=2}^{R} \beta_{ir} P_R(v_{ir}) \\
\textbf{Subject to}: \quad & \alpha_{ir}, \beta_{ir} \in \{0,1\} \\
& \sum_{i=1}^{|V|}\sum_{r=1}^{R} \alpha_{ir} S(v_{ir}) \leq \mathrm{Cache}_{\mathrm{Max}} \\
& P\left(\sum_{i=1}^{|V|}\sum_{r=1}^{R} \beta_{ir} P(v_{ir}) > BW_{tr}\right) \leq \gamma \\
& \alpha_{ir} + \beta_{ir} \leq 1 \\
& \{\forall \beta_{ir} = 1, \exists \alpha_{ik} \neq 0, \text{s.t. } k < r\}
\end{aligned} \tag{7}$$

where α_{ir} and β_{ir} are the solutions to the optimization problem, having value either 0 or 1. If $\alpha_{ir} = 1$, the video v_{ir} is proactively cached. If $\beta_{ir} = 1$, then video v_{ir} is not cached but rather considered candidate for transrating. In this case, we need to ensure there exists at least one α_{ik} with $k < r$ such that a higher rate version of video v_i is cached; the above is achieved by introducing the constraint $\{\forall \beta_{ir} = 1, \exists \alpha_{ik} \neq 0, \text{s.t. } k < r\}$. For the same reason, the subscript r of the 2nd term of the maximization starts from 2, to ensure that a higher rate version is always in the cache for videos that are envisioned for transrating.

Equation (7) has two size constraints, one is memory, $S(v_{ir})$, which is deterministic, and the other is processing, $P(v_{ir})$, which is probabilistic. Unlike cache memory which is a static resource for which the consumption is known at time of solving the optimization problem, $P(v_{ir})$ is a dynamic resource and its usage is only known after video request arrival. We discuss the probabilistic constraint in Appendix B.

Since solving binary integer program is NP-complete [19], we first solve the linear relaxation of the problem, where α_{ir} and β_{ir} obtain values between 0 and 1, and subsequently round them to either 0 or 1. The rounding of linear values to integer ones can be done using a technique known as randomized rounding [20]. Randomized rounding will round α_{ir} and β_{ir} to 1 with probability of α_{ir} and β_{ir}; this may result in both α_{ir} and β_{ir} being rounded to 1, which is not desirable as we do not want to cache a video that is selected for transrating. Hence, we propose a heuristic approach described below.

Fig. 8 explains the algorithm for rounding α_{ir} and β_{ir} to integer values. First, we sort α_{ir} and β_{ir} in descending order and create sorted lists L_α and L_β (line 1). If we simply round α_{ir} to 1 in the sorted order, we may miss rounding a lower valued α_{ir} to 1 that may result in a higher cache hit ratio due to its transrating potential for other videos. Thus, we consider both L_α and L_β when making rounding decisions as follows. For each α_{ik} in list L_a in descending order and as long as it is not the end of the list or until the cache is full (lines 2 and 3), we sum

[3][Online]. Available: http://en.wikipedia.org/wiki/exponential_smoothing

[4][Online]. Available: http://en.wikipedia.org/wiki/chi-squared_distribution

Rounding ILP solutions to Integer Values

1. Sort $\alpha_{ir} \neq 0$ and $\beta_{ir} \neq 0$ in L_α and L_β in descending order;
2. Start from the first item is the sorted list $L_\alpha : \alpha_{ik}$
3. **While** $\exists \alpha_{ik} \in L_\alpha \cap$ cache $\neq$ full
4. **If** $\exists \beta_{ir} \in L_\beta$ $(r > k)$
5. $w_{ik} = \sum \beta_{ir}$
6. **End If**
7. **If** $w_{ik} > 0$
8. Round α_{ik} to 1
9. **Else**
10. Add α_{ik} to CList (candidate List)
11. **For** $\alpha_{il} \in$ CList
12. **If** $|\alpha_{il} - \alpha_{ik}| > T_{e1}$
13. Round α_{il} to 1
14. **End If, End For, End If, End While**

Fig. 8. Rounding ILP solutions to Integer Values.

up all the β_{ir} with $k < r$ in list L_β and assign the sum to w_{ik} (lines 4 and 5). If $w_{ik} > 0$, meaning there exist videos with transrating potential from α_{ik}, round α_{ik} to 1 (lines 7 to 9); otherwise, add α_{ik} to the candidate list and identify potential α_{il} in the candidate list that can be rounded to 1 if the difference between any α_{il} in the list and the newly added α_{ik} is greater than a threshold T_{e1}. The use of a candidate list is to avoid ignoring the videos that can result in high requests but are not candidate for transrating. Further, as we round α_{ir}s to one, we add them to an *MLR* (Most Likely Requested) set; such that the first element is most likely to be requested. Following the same argument, *LLR* (Least Likely Requested) set is *MLR* set but sorted in ascending order for cached videos.

Fig. 9 details the ABR-P-UPP-P caching policy. At each time interval t, or when a new video request arrives, whichever is earlier, ABR-P-UPP-P calculates the distribution of video rates currently served and estimates $R(r)$ as explained in Section III-C1 (line 1). If either AUS or the video bit rate distribution changes more than a threshold, ABR-P-UPP-P calculates the request probability of each video i based on the cell site UPP (lines 2–6). Regardless of a change in AUS, ABR-P-UPP-P updates the request probability of each video given the bit rate (line 7). Next, the ILP formulation (7) is solved, and the *MLR* and *LLR* sets are constructed as explained in our rounding algorithm (lines 8–9). We update the cache by the *MLR* videos that are not in the cache. More specifically, for each video v_{ir} from the *MLR* set to be added to the cache, we calculate the difference between its α_{ir} and α_{kr} of the subset of *LLR* videos from the cache with least P_R values that need to be evicted to free up space for the new video. Only if the difference is greater than a threshold, T_e, we effectuate the cache update (lines 9–13). T_e is used to avoid unnecessary cache updates. Note that the algorithm can either make the caching decisions in granularity of a video chunk, multiples of a chunk, or a whole video.

In the event of a video request, if the video is a cache hit, we download the video from the cache (lines 2–5). While delivering the video v_{ir}, we lock the current and later chunks of the video so that they are available in the cache and not evicted by an invocation of the proactive cache policy during the download. If a higher rate version and transrating resource are both available, we use the resource allocation algorithm (Section III-A) to decide whether to fetch the video through the backhaul or transrate the video (line 7). If a higher rate version of the video is not in the cache, we fetch the video through the backhaul (line 9).

ABR-PUPP-P caching policy

1. At each time interval t, calculate $R(t, r)$ (eq. 3) by executing the Video bitrate Prediction algorithm
2. **If** AUS || rate distribution change $>$ Threshold
3. **If** there is change in AUS
4. Find cell site UPP based on AUS
5. Calculate request probability $P_R(v_i)$ based on new cell site UPP
6. **End If**
7. Calculate $P_R(v_{ir}) = R(r) \times P_R(v_i)$
8. Construct and Solve LP problem formulation (eq. 7)
9. Calculate MLR and LLR based on the result of LP problem
10. **For** each video i in sorted list of MLR set, MLR_i
11. LLR_k subset of LLR videos with least α_{ij} that has to be evicted from cache to fit MLR_i
12. **If** $\alpha_{ir}(MLR_i) - \sum \alpha_{ik}(LLR_k) > T_e$
13. Update the cache with MLR_i and evict LLR_k
14. **End If, End For, End If**

User Request Handling

1. Request for video v_{ir}
2. **If** $v_{ir} \in Cache$
3. Lock the access to video v_{ir} for all the chunks associated with the current download and onward
4. Move the lock forward as chunks being downloaded
5. Download v_{ir} from Cache
6. **Else If** $\exists v_{ik} \in Cache, k < r \cap BW_{tr}(v_{ik}) \leq BW_{tr-remaining}$
7. Use Resource Allocation Algorithm (Fig. 5) to decide whether to satisfy request by either transrating or fetching it through the backhaul
8. **Else**
9. Bring the video through the backhaul
10. **End If**

Fig. 9. ABR-PUPP-P caching policy.

Next, we explain our simulation parameters and evaluate the performance of our ABR and joint caching and processing policies, in contrast with other alternatives discussed before.

IV. Simulation Results

In this section, we evaluate the impact of our proposed techniques on network capacity and user QoE under various cache size, processing capacity, and wireless channel conditions. We start by describing our simulation framework and parameters used. Next, we study the impact of different wireless channel conditions on requested video bit rate distribution of our ABR algorithm, BiTRaS, as well as the accuracy of our video bit rate prediction algorithm (Section III-C1). In Section IV-C, for a certain cache size, transrating capacity, and wireless channel condition (baseline case—defined later), we evaluate the impact of BiTRaS and ABR-aware joint caching and processing policies, ABR-P-UPP-P and ABR-LRU-P, on capacity, probability of stalling and VQM experienced by users. To show the effectiveness of our proposed policies, we compare them with two alternative methods of supporting ABR for caches: 1) cache only the highest rate version of a video and transrate it to the requested video bit rate, which can be expensive in terms of transrating resources, or 2) static LRU caching policy which brings all the available rates for a missed video/chunk from the Internet CDN instead of just the requested bit rate of the video, and hence can be expensive in terms of cache size and required backhaul bandwidth. In Section IV-D, we study the effect of different cache sizes and transrating capacities on the performance of our proposed policies. Finally, in Section IV-E, we show the effectiveness of our proposed techniques under different variations of baseline case, including how users are distributed in the cell, and different wireless channel conditions.

A. Simulation Framework and Parameters Used

We extended the MATLAB Monte Carlo simulation framework we had developed earlier [3], [4] to incorporate ABR

TABLE I
WIRELESS CHANNEL SIMULATION PARAMETERS

Parameter	Distribution/Parameters Value
Total (e)NodeB Power	43dBm (in each of 3 sectors)
Channel BW	20MHz (in each of 3 sectors)
Thermal Noise at UE	-174+10log(20MHz) dBm
Noise figure, Interference Margin	7 dB, 5.5 dB
Cell Radius	1.2 km

streaming, including the new ABR rate selection algorithm and the joint caching and processing algorithms proposed in this paper. Like in [3], we assume a database of 20 000 videos following a Zipf popularity distribution with exponent value of -0.8. The video duration is exponentially distributed with mean of 8 min and truncated to a maximum of 30 min and a minimum of 2 min. We assume the videos are uniformly distributed between 200 kbps (QVGA quality) and 2 Mbps (HD quality), and that each video has four transrated variants that have relative bit rates of 0.82, 0.67, 0.55, and 0.45 of the original video bit rate. The simulation assumes a pool of 5000 potential users and uses a Poisson model for arrival to and departure from a cell, with average user active time of 45 min and inter-arrival time depending on the specific simulation. Video requests are generated independently per active user and follow a Poisson process with mean of 8 min between requests. In terms of video QoE, we set up a requirement of maximum acceptable initial delay of 10s for each user and VQM value of at least 0.65. For the proposed ABR algorithm, we use $D_{\min} =$ 10s.

In terms of resources, we use RAN cache size of up to 350 Gbit, and RAN video processing (transrating) capacity of up to 100 Mbps (e.g., 50 concurrent video requests with required rate of 2 Mbps). Furthermore, we assume 100 Mbps for backhaul bandwidth, and the wireless channel is modeled with parameters listed in Table I and will be explained further in Section IV-B.

B. Effect of Wireless Channel Variations on Rate Distribution and Accuracy of Video Bit Rate Prediction

Because variations in channel conditions lead to the need for ABR, we study their impact on video bit rate distribution produced by BiTRaS, and the accuracy of our video bit rate prediction method which will be used by ABR-P-UPP-P.

The wireless channel is characterized by variations of the channel signal strength across time and frequency. These variations are divided into large-scale and small-scale fading. In this paper, like in [3], we model small-scale fading using Rayleigh fading model and large-scale fading according to 3GPP TR 36.814 V0.4.1 [17] Urban Macro (UMa) model. The final channel model is a superposition of small-scale and large-scale fading. To model small-scale fading with different temporal variations in users' wireless channel, we use two Rayleigh fading channels with Doppler frequencies of $f_d = 3$ and 92 Hz. $f_d = 3$ Hz models small-scale fading for a pedestrian with speed of approximately 3 km/h, while $f_d = 92$ Hz corresponds to a vehicle with speed of approximately 100 km/h assuming a carrier frequency of 1 GHz. We study the impact of wireless channel on BiTRaS video bitrate changes in this subsection and on the joint caching and processing policies performance in Section IV-E using three different channel profiles: 1) baseline: $f_d = 3$ Hz with users uniformly distributed within a cell;

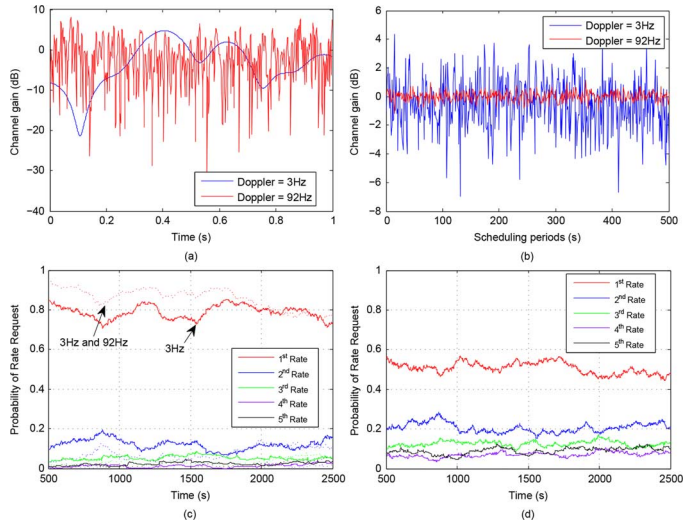

Fig. 10. (a) Instantaneous channel gain. (b) Average channel gain in scheduling interval. (c) Video rate distribution in baseline and mixed channel. (d) Video rate distribution in biased channel.

2) biased: baseline channel with users uniformly distributed 0.7–1.2 km from cell center to achieve different large-scale fading profiles relative to the baseline channel; and 3) mixed: baseline channel with a mix of users experiencing Doppler frequencies of 3 and 92 Hz to achieve different small-scale profile relative to the baseline configuration.

Fig. 10(a) shows an example of the channel gain experienced by one user during a period of one second (not including path loss). We see that the rate at which the channels vary are different, but the magnitude of the variations is comparable. However, the VAWS that we proposed in [4] or any well designed scheduler typically operates at a time scale significantly larger than that of variations shown in Fig. 10(a) and averages the conditions over a scheduling period before making the scheduling decision. Fig. 10(b) shows the average channel gain in each scheduling interval of 1 s. As we can see from the figure, the temporal variations in the magnitude of the channel, when considering averaging over each scheduling period, are much more pronounced with the pedestrian channel with $f_d = 3$ Hz than with the channel with $f_d = 92$ Hz. This is because with a slowly fading channel, a user may go into a fade and stay there during a scheduling interval and into another fade in another scheduling interval while a fast fading channel will go through many cycles within each scheduling period, so variations tend to average out more.

Fig. 10(c) and (d) shows the effect of wireless channels, described above, on the video rates selected by BiTRaS. Fig. 10(c) shows the distribution of the five different video bit rates used by BiTRaS for the baseline channel (solid lines) and for the mixed channel (dashed lines). Similar to Fig. 10(c), Fig. 10(d) shows video bit-rate distribution for the biased channel. From Fig. 10(c) and from Q-Q plot[5] not presented here due to space constraints, we observe that rate distribution is almost the same for the baseline and mixed channel, showing that BiTRaS along with VAWS does not change video rates due to short term channel fluctuations. However, from Fig. 10(d), we infer that BiTRaS resorts to more bit-rate adaptation for the biased channel to compensate for the path loss observed by the

[5][Online]. Available: http://en.wikipedia.org/wiki/qq_plot/

TABLE II
CAPACITY, STALL PROBABILITY, AND VQM

ABR/Caching Policy	Capacity	P(stall)	VQM
1. No ABR, No Cache	99	0.010	1
2. No ABR, RAN Cache [LRU]	159	0.060	1
3. ABR, No RAN Cache	145	0.00025	0.78
4. ABR, RAN Cache [ABR-LRU-P]	223	0.0020	0.84
5. ABR, RAN Cache [Static LRU]	101	0.0114	0.77
6. ABR, RAN Cache [ABR-P-UPP-P]	262	0.0018	0.90
7. ABR, RAN Cache [Highest Rate LRU]	187	0.0030	0.85

users farther away from the cell center—for instance, at about 2000 s into the simulation, the probability that BiTRaS uses the highest bit rate is 0.46 for the biased channel and 0.8 for baseline or mixed channel whereas the probability of using the 2nd highest rate is 0.18 for the biased and 0.12 for the baseline or mixed channel. Overall, for the settings considered in this section, the aggregate effect of each user's Doppler frequency on the rate distribution across the cell is minor and the main contributor to the variations in the rate distribution over time is the network load as well as significant change in users' large-scale fading.

To validate how well our video bit rate prediction algorithm (Section III-C1) predicts the distribution of rates requested by BiTRaS, we use statistical hypothesis testing specifically using the chi-square test . Based on chi-square testing, as we have five different bit rates to predict, the degree of freedom is four and, to get a 95% confidence interval from the chi distribution , we get a value of 8.0 for null hypothesis rejection threshold. Thus, if the difference between observed and estimated rate is below 8.0, then we cannot reject the null hypothesis that the predicted distribution is identical to the actual distribution of bit rates. We measure the percentage of occurrences that the chi-squared statistics is above the null hypothesis rejection threshold. Using these measurements, we get very low values of 2.8% and 2.6% for predictions made for the baseline and mixed configurations respectively, showing that most of the time the predicted rate distributions are accurate. The use of chi-square test is appropriate due to the weakness of dependency [21] of the requested video rates. We compared throughput of the 25% of users with highest and lowest achieved throughput for every time tick and realized a low correlation coefficient for the two time series. The low correlation coefficient points to the two groups being sufficiently "separated" and therefore weakly dependent.

Next, we study the impact of BiTRaS and the joint caching and processing policies on the network capacity and video QoE.

C. Performance of ABR-Aware Joint Caching and Processing Policies: Baseline Case

Using the simulation framework described in Section IV-A, we next quantify the advantages of our proposed techniques—ABR streaming and joint caching and processing in the RAN—in terms of network capacity (number of concurrent video streams) and QoE (probability of stalling and VQM score). We use a baseline configuration of parameters (baseline case) that includes all the parameters described in Section IV-A, with 250 Gbit cache size, 12 Mbps transrating capacity, Doppler frequency of 3 Hz, and users uniformly distributed across a cell. Table II shows the capacity (number of concurrent video streams served), probability of stalling, and VQM, for different combinations of ABR, RAN cache, and cache policy usage. The capacity numbers reported are where the blocking probability is exactly 0.01 [3], which is achieved by changing the user inter-arrival rate such that the steady state target blocking rate is achieved and noting the number of concurrent video requests generated at that specific user inter-arrival rate. Furthermore, using RAN cache by itself (no ABR) (line 2) or using BiTRaS by itself (no RAN cache) (line 3) can improve the capacity by up to 61% and 46% respectively compared to using neither RAN cache nor ABR (line 1). However, the biggest gains in capacity arise when using our joint RAN caching and processing policies together with BiTRaS (lines 4 and 6). Using the proposed ABR-LRU-P joint caching and processing policy, the capacity improves by 54% compared to having ABR but no cache (comparing lines 3 and 4), by 125% compared with having no RAN cache and no ABR (comparing lines 1 and 4), and by 121% compared to using ABR with the static LRU caching policy (comparing lines 4 and 5). Furthermore, using ABR-P-UPP-P policy can further improve capacity by up to 17% compared with using ABR-LRU-P policy (comparing lines 4 and 6), and by 40% compared to using ABR with Highest Rate LRU policy that caches the highest rate and uses transrating capacity to transrate the videos to the desired video bit rate (comparing lines 6 and 7).

The Probability of Stalling column in Table II shows the total number of stalls across all the video requests divided by the total number of video requests. From the results we can infer that using ABR without any caching can reduce the stalling probability significantly (comparing lines 1 and 3). This improvement in terms of capacity and stall probability comes with the cost of drop in VQM. On the other hand, adding RAN caching can increase the stalling probability (both lines 2 and 5), due to increased number of video requests supported. However, the results show that, when our proposed ABR-aware joint caching and processing policies are used, the stalling probability can be reduced significantly (comparing lines 4 or 6 with lines 2 or 5), though the stalling probability is still higher than using ABR without caching (line 3), while achieving much higher capacity.

The VQM column in Table II shows the VQM of the served videos. Here, without presence of ABR we get VQM value of 1 which is the highest possible score, because videos are always delivered at the highest quality, but the lack of rate adaptation causes excessive stalling, as can be seen in the Probability of Stalling column in Table II, which may be more detrimental to QoE than degraded VQM. When ABR is used without RAN caching, VQM degrades to 0.78 while stalling is almost eliminated. However, when ABR is used with RAN caching and processing with the ABR-LRU-P or ABR-P-UPP-P policies, we observe very little further degradation in VQM score and similar stalling probability, while significant increase in capacity.

Overall, we conclude that: 1) caching and procesisng policy is the determining factor in the capacity and QoE improvement, given the same amount of investment (cache size and processing resource); for example, even with the added cache and processing resource, using "Static LRU" does not improve the capacity and QoE compared with not having any cache or processing resource and 2) using "ABR-P-UPP-P" policy with cache size of 250 Gbit and transrating of 12 Mbps results in 81% improvement in terms of end-to-end video capacity compared with using ABR alone.

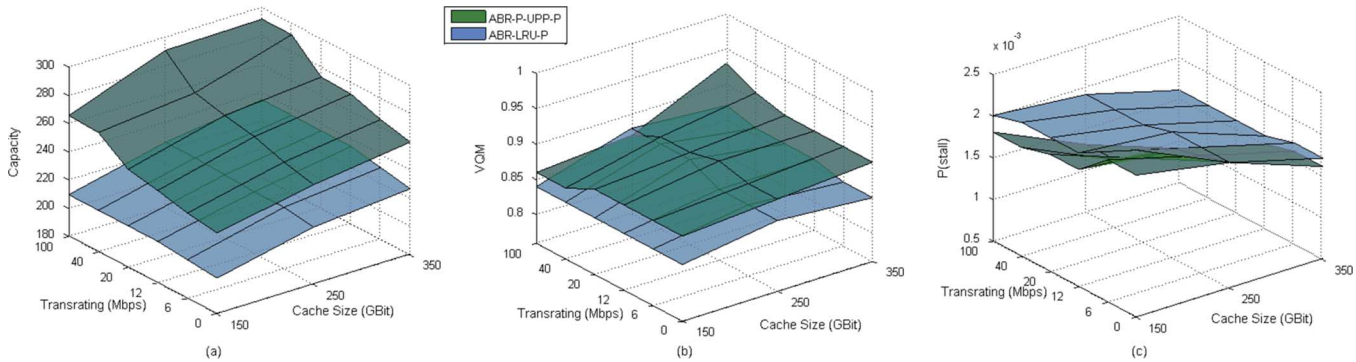

Fig. 11. (a) Capacity versus transrating and cache size. (b) VQM versus transrating and cache size. (c) Stall probability versus transrating and cache size.

D. Impact of Cache Size and Transrating Resources

Here, we study the impact of cache size and transrating capacity on the performance of our proposed joint caching and processing policies.

Fig. 11(a) shows the effect on capacity when increasing cache size from 150 to 350 Gbit and increasing the transrating capacity from 0 to 100 Mbps for each cache size, with the other parameters unchanged from the baseline configuration. From the figure, we can infer that as the cache size and transrating resource increase, the capacity achieved by both proposed policies increase. For the cache sizes that we study in this paper, increase in cache size always results in increase in capacity. For instance, from Fig. 11(a), we see that when no transrating resource is available, the capacity achieved by ABR-P-UPP-P increases from 237 to 250 to 259 with cache size increasing from 150 Gbit to 250 Gbit to 350 Gbit, respectively, an overall increase of 9% in capacity.

Similarly, increasing transrating capacity also leads to increase in network capacity, although the increase depends on associated cache size, and may stop after a certain transrating capacity. For instance, Fig. 11(a) shows that for cache size of 150 Gbit, increasing transrating capacity from 0 Mbps to 12 Mbps increases the network capacity by 3% and 2% for ABR-P-UPP-P and ABR-LRU-P, respectively; on the other hand, while increasing the transrating resource from 20 Mbps to 100 Mbps increases the network capacity by 6% using ABR-P-UPP-P, it does not further increase the capacity using ABR-LRU-P. For higher cache size of 350 Gbit, increasing transrating resource from 0 to 6 to 12 to 100 Mbps shows continuous increase in network capacity by ABR-P-UPP-P, for a total capacity increase of 12%, due to the availability of more videos with higher bit rate versions in the cache. Furthermore, ABR-P-UPP-P joint caching and processing policy can improve the capacity by 24% compared with ABR-LRU-P joint caching and processing policy when the cache size is 350 Gbit and transrating resource is 100 Mbps.

Next, we study the impact of cache size and transrating capacity on VQM using ABR-LRU-P and ABR-P-UPP-P policies. From Fig. 11(b), we can infer that overall, regardless of the available cache size and transrating resource, VQM is greater than or equal to 0.86 for the ABR-P-UPP-P joint caching and processing policy. Similar trends holds for ABR-LRU-P with VQM of equal or above 0.84. Furthermore, increase in cache size results in slight increase in VQM value. For instance, for transrating capacity of 6 Mbps, increasing the cache size from 150 to 250 to 350 Gbit shows a slight increase in VQM by ABR-P-UPP-P and ABR-LRU-P of about 2%.

Next, we study the impact of cache size and transrating capacity on stalling probability using ABR-LRU-P and ABR-P-UPP-P policies. From Fig. 11(c), we can see that increasing cache size itself reduces stalling probability—for example, for transrating capacity of 12 Mbps, increasing cache size from 150 Gbit to 350 Gbit, reduces the stalling probability by 24% and 23% for ABR-P-UPP-P and ABR-LRU-P policies respectively. Similarly, increasing transrating capacity itself can reduce stalling probability—for example, for cache size of 350 Gbit, increasing transrating capacity from 0 to 100 Mbps reduces stalling probability by 44% and 6% for ABR-P-UPP-P and ABR-LRU-P, respectively. Finally, we can infer that increase in both cache size and transrating capacity can significantly improve the probability of stalling. For instance, increasing the transrating capacity from 0 Mbps to 100 Mbps and cache size from 150 Gbit to 350 Gbit improves the stalling probability by 59% and 36% for ABR-P-UPP-P and ABR-LRU-P policies respectively.

Overall, we can infer that capacity and video QoE of the cellular network is sensitive to cache size and transrating resources and for the settings studied in this paper, increase in cache size always results in higher capacity and better QoE. However, while increase in transrating resource also results in increase in capacity and QoE, its impact highly depends on the cache size available and it plateaus after reaching a certain threshold.

Next, we study the impact of wireless channels and user distributions on the capacity and QoE obtained of the ABR-P-UPP-P and ABR-LRU-P joint caching and processing policies.

E. Impact of Wireless Channel and User Distribution

As explained in Section IV-A, different types of channels and their variations impact the frequency of rate changes due to ABR and as a result impact the performance of our joint caching and processing policies. In this subsection, we quantify the impact of wireless channel variations on the capacity and QoE of the wireless network using the three different channel configurations detailed in Section IV-B: baseline, biased and mixed channel. Table III shows video capacity and QoE achieved by different policies for the three different configurations explained above. Since Highest Rate LRU policy performs the best for the

TABLE III
CAPACITY AND QOE PERFORMANCE OF POLICIES UNDER DIFFERENT WIRELESS CHANNEL AND USER DISTRIBUTIONS

ABR/Caching Policy	Cap.	P(stall)	VQM
No ABR, No RAN Cache (baseline)	99	0.0164	1
No ABR, No RAN Cache (biased)	99	0.016	1
No ABR, No RAN Cache (mixed)	99	0.029	1
ABR, RAN Cache [HR-LRU] (baseline)	187	0.003	0.85
ABR, RAN Cache [HR-LRU] (biased)	143	0.0032	0.87
ABR, RAN Cache [HR-LRU] (mixed)	190	0.003	0.85
ABR, RAN Cache [ABR-LRU-P] (baseline)	223	0.002	0.84
ABR, RAN Cache [ABR-LRU-P] (biased)	206	0.0058	0.88
ABR, RAN Cache [ABR-LRU-P] (mixed)	226	0.0029	0.86
ABR, RAN Cache [ABR-P-UPP-P] (baseline)	262	0.0018	0.90
ABR, RAN Cache [ABR-P-UPP-P] (biased)	223	0.00225	0.90
ABR, RAN Cache [ABR-P-UPP-P] (mixed)	267	0.00164	0.92

baseline configuration, we select it to compare the performance of our proposed policies against.

From the table, we see that biased distribution of users within the cell site decreases capacity of ABR-P-UPP-P by 15% while it impacts the capacity of ABR-LRU-P only by 8% compared with the baseline configuration. This is due to the fact that ABR-P-UPP-P results in a higher number of concurrent users admitted than the other policies, and hence having more users at the cell edge in the biased configuration results in higher required (e)NodeB power to admit the edge users and thus negatively impacts video capacity more than in the case of the other policies.

We also observe that mixed configuration results in very minor capacity improvements of 1.9% and 1.4% for ABR-P-UPP-P and ABR-LRU-P compared with the modified baseline configuration respectively. This minimal change in capacity despite more variations experienced at the scheduling intervals with baseline channel [Fig. 10(b)], indicates that BiTRaS and VAWS do not unncessarily change the rate due to short term channel variations.

Overall, the simulation results show that our proposed policies perform significantly better than not using ABR and RAN cache, and also far better than Highest Rate LRU caching policy, in any of the configurations considered. For instance, with biased configuration that negatively affects the capacity for all the caching policies, ABR-P-UPP-P performs better than Highest Rate LRU by 56%, and better than no ABR and no RAN cache by 125%.

Table III shows that VQM for our caching policies is the same or better than the VQM for the Highest Rate LRU across all the wireless channel configurations. For instance, for the mixed configuration, VQM is 0.85 for Highest Rate LRU, 0.86 for ABR-LRU-P and 0.92 for ABR-P-UPP-P policy. Furthermore, Table III shows that the probability of stalling is significantly lower when using our policies for any of the configurations considered. For instance, when using mixed configuration, ABR-LRU-P and ABR-P-UPP-P reduce probability of stalling by 3% and 45% respectively relative to Highest Rate LRU policy, and by 90% and 94% respectively relative to using no ABR and no RAN caching. From results not presented in this paper due to space limitation, we infer that users experience initial delay of between 5.5 s to 7.5 s across all of the policies compared here.

V. CONCLUSION

We proposed a novel video rate adaptation algorithm that uses video frame characteristic through E-LBP to change the video bit rate and transmission rate served in response to the varying wireless channel conditions and network utilization to improve the video QoE while also improving the capacity. Furthermore, we proposed two joint video caching and processing algorithms that are ABR-aware and aim to improve the capacity and QoE of the wireless network. Using simulation results, we show our rate adapation along with our joint caching and processing policies produce superior results compared with using commonly deployed caching algorithms.

ACKNOWLEDGMENT

The authors would like to thank Prof. R. Impagliazzo and Prof. D. Politis for discussions of complexity analysis and hypothesis testing, respectively.

REFERENCES

[1] Microsoft Download Center, "Smooth streaming technical overview," Mar. 2009.
[2] S. Akhshabi *et al.*, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," *ACM Multimedia Syst.*, pp. 157–168, 2011.
[3] H. Ahlehagh and S. Dey, "Video caching in radio access network: Impact on delay and capacity," in *Proc. IEEE Wireless Commun. Networking Conf.*, 2012, pp. 2276–2281.
[4] H. Ahlehagh and S. Dey, "Video aware scheduling and caching in the radio access network," *IEEE Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, Aug. 2014.
[5] D. Pisinger, "Algorithms for knapsack problems," Ph.D. dissertation, DIKU, Univ. Copenhagen, Copenhagen, Denmark, 1995.
[6] H. Kellerer *et al.*, "Knapsack problems," 2004.
[7] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic HTTP streaming," in *Proc. ACM Int. Conf. Emerging Networking EXperiments Technol.*, Dec. 2012, pp. 109–120.
[8] C. Liu *et al.*, "Rate adaptation for adaptive HTTP streaming," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst.*, New York, NY, USA, 2011, pp. 169–174.
[9] J. Martin *et al.*, "Characterizing Netflix bandwidth consumption," in *Proc. IEEE Consumer Commun. Networking Conf.*, 2013, pp. 230–235.
[10] X. Wang *et al.*, "Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users," *IEEE Wireless Commun. J.*, vol. 20, no. 3, pp. 72–79, Jun. 2013.
[11] B. Tarbox, "Complexity considerations for centralized packaging vs. remote packaging," *Technical Forum Proc.*, Spring 2012.
[12] B. Tarbox, "Intelligent caching in an ABR multi-format CDN world," *Technical Forum Proc.*, Spring 2012.
[13] W. Zhang *et al.*, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE. Trans. Multimedia*, vol. 15, no. 6, pp. 1431–1445, Aug. 2013.
[14] J. Ribas-Corbera *et al.*, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst.*, vol. 13, no. 7, pp. 674–687, Jul. 2003.
[15] A. K. Moorthy *et al.*, "Video quality assessment on mobile devices: Subjective, behavioral and objective studies," *IEEE J. Sel. Topics Signal Process.*, vol. 6, no. 6, pp. 652–671, Oct. 2012.
[16] A. Balachandran *et al.*, "Developing a predictive model of quality of experience for internet video," in *Proc. ACM SIGCOMM*, Hong Kong, Aug. 2013, pp. 339–350.
[17] 3GPP, "TR36.814 v1.0.0: Evolved universal terrestrial radio access (E-UTRA); further advancements for E-UTRA physical layer aspects," 2009 [Online]. Available: www.3gpp.org
[18] B. L. Bowerman *et al., Forecasting, Time Series, and Regression*, 4th ed. Boston, MA, USA: Cengage Learning, 2007.
[19] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York, NY, USA: Plenum, 1972, pp. 85–103.
[20] R. Motwani and P. Raghavan, *Randomized Algorithms*, 1st ed. Cambridge, U.K.: Cambridge Univ., 1997.
[21] D. N. Politis, "Computer-intensive methods in statistical analysis," *IEEE Signal Process. Mag.*, vol. 15, no. 1, pp. 39–55, Jan. 1998.
[22] R. Neapolitan, *Foundations of Algorithms*, 3rd ed. Sudbury, MA, USA: Jones and Bartlett, 2003.

[23] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, no. 3, pp. 385–463, May 2004.
[24] C. Koufogiannakis and N. E. Young, "A nearly linear-time PTAS for explicit fractional packing and covering linear programs," *Algorithmica*, vol. 70, pp. 648–674, Dec. 2014.
[25] [Online]. Available: http://en.wikipedia.org/wiki/normal_distribution
[26] P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: Wiley, 1990.
[27] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1st ed. San Francisco, CA, USA: Freeman, 1979.

Hasti A. Pedersen received the M.S. degree in electrical engineering from Worcester Polytechnic Institute, Worcester, MA, USA, in 2004. She is currently working toward the Ph.D. degree at the University of California at San Diego, La Jolla, CA, USA.

Prior to her doctoral work, she was a Senior Staff Software Engineer with Motorola Mobility Inc., where she worked on cable networks, and before that she was a Firmware Software Engineer developing software for 3G wireless networks.

Sujit Dey (SM'03–F'14) received the Ph.D. degree in computer science from Duke University, Durham, NC, USA, in 1991.

He is a Professor with the Department of Electrical and Computer Engineering, University of California at San Diego (UCSD), La Jolla, CA, USA, where he heads the Mobile Systems Design Laboratory, which is engaged in developing innovative mobile cloud computing architectures and algorithms, adaptive multimedia and networking techniques, low-energy computing and communication, and reliable system-on-chips, to enable the next-generation of mobile multimedia applications. He is the Director of the UCSD Center for Wireless Communications. He also serves as the Faculty Director of the von Liebig Entrepreneurism Center and is affiliated with the Qualcomm Institute. He served as the Chief Scientist, Mobile Networks, at Allot Communications from 2012 to 2013. He founded Ortiva Wireless in 2004, where he served as its founding CEO and later as CTO till its acquisition by Allot Communications in 2012. Prior to Ortiva, he served as the Chair of the Advisory Board of Zyray Wireless till its acquisition by Broadcom in 2004. Prior to joining UCSD in 1997, he was a Senior Research Staff Member with the NEC C&C Research Laboratories, Princeton, NJ, USA. He has coauthored over 200 publications, including journal and conference papers, and a book on low-power design. He is the co-inventor of 18 U.S. patents, resulting in multiple technology licensing and commercialization.

Dr. Dey was the recipient of six IEEE/ACM Best Paper Awards and has chaired multiple IEEE conferences and workshops.