Video-Aware Scheduling and Caching in the Radio Access Network

Hasti Ahlehagh and Sujit Dey

***Abstract*—In this paper, we introduce distributed caching of videos at the base stations of the Radio Access Network (RAN) to significantly improve the video capacity and user experience of mobile networks. To ensure effectiveness of the massively distributed but relatively small-sized RAN caches, unlike Internet content delivery networks (CDNs) that can store millions of videos in a relatively few large-sized caches, we propose RAN-aware reactive and proactive caching policies that utilize User Preference Profiles (UPPs) of active users in a cell. Furthermore, we propose video-aware backhaul and wireless channel scheduling techniques that, in conjunction with edge caching, ensure maximizing the number of concurrent video sessions that can be supported by the end-to-end network while satisfying their initial delay requirements and minimize stalling. To evaluate our proposed techniques, we developed a statistical simulation framework using MATLAB and performed extensive simulations under various cache sizes, video popularity and UPP distributions, user dynamics, and wireless channel conditions. Our simulation results show that RAN caches using UPP-based caching policies, together with video-aware backhaul scheduling, can improve capacity by 300% compared to having no RAN caches, and by more than 50% compared to RAN caches using conventional caching policies. The results also demonstrate that using UPP-based RAN caches can significantly improve the probability that video requests experience low initial delays. In networks where the wireless channel bandwidth may be constrained, application of our video-aware wireless channel scheduler results in significantly (up to 250%) higher video capacity with very low stalling probability.**

***Index Terms*—Mobile video, user experience, user preference, video caching, wireless network capacity.**

I. Introduction

With the worldwide growth in the adoption of smartphones and tablets, access to Internet video and video applications from mobile devices is projected to grow very significantly [1]. When Internet video is accessed by a mobile device, the video must be fetched from the servers of a Content Delivery Network (CDN) [2], [3]. CDNs help reduce Internet bandwidth consumption and associated delay/jitter, but the video must additionally travel through the wireless carrier Core Network (CN) and Radio Access Network (RAN) before reaching the mobile device. Besides adding to video latency, bringing each requested video from the Internet CDNs can put significant strain on the carrier's CN and RAN backhaul, leading to congestion, significant delay, and constraint on the network's capacity to serve a large number of concurrent video requests.

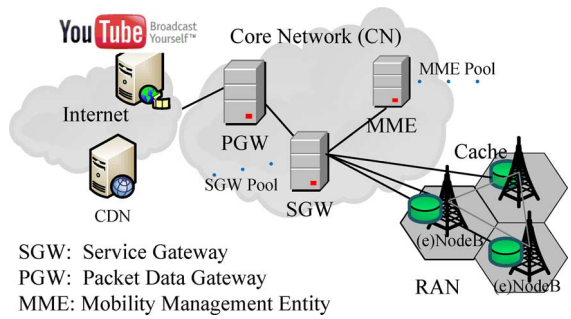


Fig. 1. Video micro-caches at the edge of the RAN.

The above problem will be further exacerbated by the recent advances in radio technologies and architectures like LTE, LTE Advanced, small cells, and Het Nets, which will increase the radio access capacities very significantly, shifting the capacity challenge and congestion problem to RAN backhaul. According to Juniper Research, operators will need to spend almost $840 billion globally over the next five years in order to address serious bottlenecks in their RAN backhaul networks [4]. According to a report just released by Strategy Analytics [5], "as global mobile data traffic grows by another 5 to 6 times over the next five years operators will face a new mobile capacity crunch by 2017 unless they increase traditional backhaul investment levels to match the anticipated growth in Radio Access Networks (RAN) capacity and user traffic." According to the report, there will be potentially a 16-PB shortfall in backhaul capacity by 2017.

To facilitate the tremendous growth of mobile video consumption without the associated problems of congestion, delay, and lack of capacity, in this paper we introduce caching of videos at (e)NodeBs at the edge of the RAN, shown in Fig. 1, so that most video requests can be served from the RAN caches, instead of having to be fetched from the Internet CDNs and travel through the RAN backhaul. However, since the proposed approach will lead to thousands of caches, with each (e)NodeB in the carrier RAN having a cache (and may be access points in Wi-Fi hotspots), we need to use much smaller-sized "micro-caches" for RAN caching, capable of storing thousands of videos, compared to the much larger-sized caches used in

Manuscript received August 10, 2012; revised March 04, 2013 and July 01, 2013; accepted August 02, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. E. Li. Date of publication January 09, 2014; date of current version October 13, 2014. This work was supported by the Intel-Cisco Video Aware Wireless Networks (VAWN) Program and the UC Discovery Grant Program.

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

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.2013.2294111

Internet CDNs capable of holding millions of videos. Hence, there may be a problem with enabling high cache hit ratio for the RAN micro-caches, which may erode the benefits of caching at the edge of the wireless network.

To address the above challenge, we propose novel caching policies, based on new concepts we introduce in the paper: the preference of current video users in a cell, and what videos are least likely and most likely to be watched by the cell users. For those video requests that cannot be found in RAN caches, and hence need to be fetched from Internet CDNs, we propose a video scheduling approach that allocates the RAN backhaul resources to the video requests such that the overall capacity of the network in terms of the number of concurrent video requests is improved, while satisfying video quality-of-experience (QoE) requirements. It has been recently shown that initial startup delay and stalling during playback are associated with increased user abandonment when viewing videos [6]. *Hence, we define QoE requirements as meeting an initial delay requirement and ensuring no or limited stalling during playback.*

Furthermore, all the video requests, whether they are being served from the cache or Internet CDNs, need to traverse through the wireless channel. Even with increased bandwidth expected from Long Term Evolution (LTE), there can be scenarios where the bandwidth constraint of the wireless channel may prevent a video found in the RAN cache, or a video successfully scheduled through the RAN backhaul, to be delivered to the mobile device in a way that satisfies its QoE requirement. In order to address end-to-end video capacity of the network, we also propose a video-aware wireless channel scheduler that will maximize the number of videos that can be delivered through the wireless channel, conscious of the channel conditions and the QoE needs of the videos.

We developed a discrete event statistical simulation framework using MATLAB to study the performance of RAN caches along with caching policies, RAN backhaul, and wireless channel scheduler under realistic video access pattern as well as wireless channel conditions. Our numerical results show that in typical cases, RAN caching with the proposed caching policies leads to significant improvements in terms of video delay and system capacity. Furthermore, we study the effectiveness of our caching policies under different usage scenarios by varying some of the more significant simulation parameters. Results of our study reveal that even under the most unfavorable usage scenarios, our User Preference Profile (UPP)-based caching policies outperform the conventional caching policies. Simulation results show that RAN micro-caches with the proposed UPP-based caching policies, together with the proposed backhaul scheduler, can improve the probability that video requests can meet initial delay of 5 s or less by 72 percentage points, and the number of concurrent video requests that can be served by up to 300% compared to having no caches in the RAN. When the wireless channel is also constrained, use of the proposed video-aware wireless channel scheduler ensures improvement is still significant—e.g., in some typical cases, we achieve end-to-end capacity improvements of 247% compared to having no caches in the RAN and using conventional scheduling algorithms. We achieve end-to-end capacity improvements of 35% compared to RAN micro-caches with conventional caching policies and conventional scheduling algorithms.

A. Related Work and Paper Outline

Significant work has been done in developing CDNs for Internet content [2], [3], as well as caching techniques and locations suitable for Internet content delivery [7]–[9]. As explained earlier, caching at Internet CDNs does not address the problems of delay and capacity for video delivery in wireless networks. Recently, there has been renewed interest in video caching for online videos like YouTube, Hulu, etc. Reference [10] investigated the effectiveness of caching Most Popular Videos (MPV) as published by Hulu, caching using Least Recently Used (LRU) policy, and a combination of the two using traces collected from a university campus. Reference [11] uses temporal similarities during different times of the day to improve the performance of LRU. Reference [12] proposes a rank-based caching technique, where ranking is defined by a combination of video popularity, cost to the server, and size of the video, to replace the videos. Like the Internet caching techniques, the above Internet video caching techniques do not address the problem of video capacity or delay in cellular networks. Moreover, as shown in Section V, conventional Internet video caching techniques like MPV and LRU, which assume large caches, may not be effective for the smaller and distributed RAN micro-caches proposed in this paper.

There has been some research in caching Web content in wireless networks [13] and on mobile devices [14]. A recent study based on HTTP traffic collected in a cellular network shows the potential benefits of caching data in the carrier core network [15]. However, these techniques do not consider the challenges of video caching and delivery and do not consider caching at (e)NodeBs at the edge of the RAN. Caching techniques have also been developed for ad hoc networks, like [14] and [16], which are not applicable to the problem of video caching and delivery in cellular networks. Recently, a promising caching approach has been proposed in [17] to improve the video capacity of cellular networks. However, the approach needs the presence of additional helper nodes (e.g., Femto cells) where videos are cached, and for users to have access to multiple helper nodes, both of which may be hard to satisfy. Moreover, the approach uses most popular videos for caching, which may not lead to optimal results as explained and shown later in this paper. In our approach, we do not need any additional nodes in the wireless network, but rather utilize existing RAN nodes for caching. We develop new RAN-aware video caching algorithms that we demonstrate to be much more effective than caching based on most popular videos. We are not aware of published research in RAN video caching and delivery, including caching policies aware of the preferences of users in a RAN cell, and scheduling of videos to improve video capacity while satisfying QoE, which are the problems we address in this paper.

In summary, the novelty and importance of the contributions in this paper are: 1) the first approach to propose and study video caching at the very edge of a cellular network, with micro-caches at each (e)NodeB, to address the problem of backhaul congestion and video buffering delay; 2) the first approach

to address the associated problem of achieving high cache hit ratio in spite of the limited size of the RAN caches—developing caching policies based on the active users in a cell and their video preferences reflected by their UPPs; 3) demonstrating the effectiveness of the proposed RAN based caching policies in achieving high cache hit ratios when conventional Internet caching approaches cannot; 4) the first backhaul and wireless channel scheduling techniques that address the two increasingly important video user experience metrics, initial delay and stalling, through the novel use of Leaky Bucket Parameters (LBP); and 5) the first study to demonstrate that significant gain in the end-to-end video capacity of a cellular network can be obtained using RAN caching and video-aware scheduling techniques, while meeting initial delay requirements and reducing video stalling significantly. Note none of the proposed approaches affect either the spatial, temporal, or bit-rate (frame) quality of the video sessions.

Furthermore, to understand the potential latency benefits of deploying the caches proposed in this paper within the cellular networks, we have developed and conducted experiments showing the delay advantages of fetching data from a carrier network instead of from Internet CDN. The "TRACEROUTE" from a user equipment (UE) within a US mobile carrier network to the Amazon data center shows that there are 19 hops (nodes) within the path from mobile client to the Amazon data center, of which 1–7 are in the carrier core network, 8–14 are associated with the Internet, and 15–19 are in the Amazon cloud.

Fig. 2(a) and Fig. 2(b), which is a magnified version of Fig. 2(a), shows the distribution of the round-trip time (RTT) to hop 2 (within operator core network) and hop 19 (within amazon cloud) based on 500 rounds of running the TRACEROUTE command. The results show that the RTTs between the mobile client and the cloud node 19 can be significantly larger compared to the RTTs from the client to one of the nodes within the carrier networks. For example, RTTs of greater than 1000 ms are experienced from the cloud node, while not from node 2 inside the operator core network. From Fig. 2(b), we can see the probability of experiencing RTT of 100 ms or less is very high from node 2 (> 0.98), and quite low for node 19 (< 0.43). From Fig. 2(a), it is also evident that the standard deviation of RTT from node 2 is significantly less than from node 19. From the above, we can infer that if data (e.g., videos) can be fetched from within the carrier network, as opposed to from an Internet CDN, we can significantly reduce the large delays experienced from Internet CDNs—thus, significantly increase the probability of meeting smaller and acceptable packet delays like 100 ms, and significantly reduce the variation in delay experienced if we have to fetch from the CDN.

The remainder of the paper is organized as follows. In Section II, we first review relevant prior results on popularity of online video and video categories. Based on these results, we define the video category preferences of active users in a cell and most likely and least likely requested videos of such users. In Section III, we provide an overview of the conventional caching policies and the user preference-based policies we introduce in this paper. Subsequently, in Section IV, we introduce our video scheduling approaches, including backhaul and wireless channel scheduling. Section V outlines our simulation framework and provides experimental results. We conclude the paper in Section VI. In Appendix-A, we study the time complexity of the caching algorithms that we propose in this paper. LBP generation is detailed in Appendix-B.

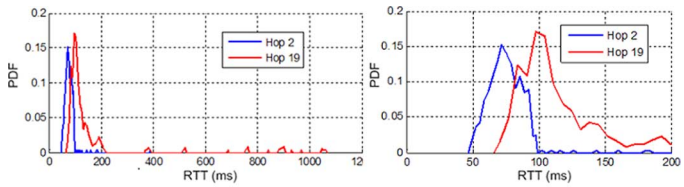


Fig. 2. RTT measurements: (a) probability density function (pdf) of experienced RTT; (b) pdf of experienced RTT magnified for RTT of 0–200 ms.

II. Video Preference of Users in a Cell Site

In this section, we first review previous research on video popularity characteristics and users' video access patterns. We highlight the conclusions that lead us to define our user preference-based approach for video caching. Toward that end, we identify the most likely and least likely requested videos in a cell site, given the current set of active users in the cell.

A. Popularity of Online Videos and Video Categories

Recently, there have been several studies done on the popularity of online videos. Using empirical analysis, [18] studied several characteristics of videos from popular online video sites YouTube and Daum, such as the overall popularity distribution and distribution within each video category, correlation between age of a video and its popularity, and temporal locality of the videos. One of the relevant results from the study is that video popularity follows a Zipf distribution: 10% of the online videos account for nearly 80% of the views, while the remaining 90% of the videos account for only a total 20% of views [18]. In [19], the authors studied users' access patterns in video traffic from a campus network. Among other interesting conclusions, their results showed that local video popularity can differ significantly from national video popularity. Separately, market research [20] has shown that some video categories can be significantly more popular than others. For example, popular video categories such as "Auto" and "Entertainment" have a 90-days average cumulative views number that is 10 times more than less popular video categories such as Travel, revealing a strong bias toward some video categories.

From above research, we conclude the following: 1) video popularity follows a Zipf distribution; 2) national video popularity does not reflect local video popularity, so the video popularity in different cell sites may be different from each other and the national popularity distribution; and 3) users may have strong preferences toward specific video categories. These results motivate us to define and identify video preferences of the active users in a cell in terms of video categories that they prefer to watch.

B. Cell Site Video Preference

To understand local video popularity in a cell site, we define active user set (AUS) of a cell at any given time as the subset of

users in the cell at that time who either have an active video session or who have watched a video when present in the cell. AUS changes as users enter or leave the cell site. In LTE, (e)NodeBs know the location of the UEs in connected mode, or the general location of UEs in idle mode, and can therefore keep track of AUS of each (e)NodeB. We associate a UPP with each individual user, which we define as the probability that the user u_k requests videos of a specific video category vc_j, $p(vc_j|u_k)$, for all available video categories. The probability that a video belonging to video category vc_j is requested by the active users in a cell, AUS, is the weighted sum of probabilities that vc_j is being selected by each user in the AUS and is given by

$$p_{\mathrm{AUS}}(vc_j) = \sum_{k=1}^{|U|} p(u_k)p(vc_j|u_k). \tag{1}$$

In the above equation, $|U|$ is the cardinality of AUS, and $p(u_k)$ is the probability that user u_k generates a video request. Following this definition, we define the UPP of an AUS as the selection probability of each available video category by the AUS: $\{p_{\mathrm{AUS}}(vc_j)|\forall j = 1 : |VC|\}$. We assume all users are equally likely to generate a video request, so we can rewrite (1) as: $p_{\mathrm{AUS}}(vc_j) = \frac{1}{|U|}\sum_{k=1}^{|U|} p(vc_j|u_k)$.

Next, given the overall video popularity distribution, and the category of each video, we identify the video popularity distribution within each category. In general, a video can belong to multiple video categories, but for convenience of notation, in this paper we assume a video belongs to only one video category. Let $p(v_i)$ be the overall popularity of video v_i across all videos and video categories. Let $p_{vc_j}(v_i) = p(v_i)$ if v_i belongs to category vc_j, else $p_{vc_j}(v_i) = 0$. We can then express popularity of video v_i within video category, vc_j, by

$$p(v_{i,j}) = \frac{p_{vc_j}(v_i)}{\sum_{i=1}^{|V|} p_{vc_j}(v_i)} \tag{2}$$

where $|V|$ is the total number of videos, and the denominator is the sum of the probabilities of all videos belonging to vc_j. Note that video popularity distribution may be available for each category [18], else it can be calculated using (2). Knowing the probability of request of different video categories in a cell corresponding to the current AUS (1), and the popularity of videos in each category, we can now derive $P_{\mathrm{R}}(v_i)$, which is the probability that video v_i is requested given the AUS of the cell

$$P_{\mathrm{R}}(v_i) = \sum_{j=1}^{|VC|} p(v_{i,j})p_{\mathrm{AUS}}(vc_j). \tag{3}$$

We next define two sets that we use for the UPP-based caching policies that we define in Section III: Most Likely Requested (MLR) and Least Likely Requested (LLR) sets. MLR is a subset of videos with P_{R} values greater than a threshold; LLR is a subset of videos from the cache with the least P_{R} value.

III. Cell-Site-Aware Caching Algorithms

In this section, we outline four different caching algorithms: two that are conventionally used by Internet CDNs, MPV and LRU, and two that we propose based on preferences of active users in a cell, P-UPP and R-UPP.

A. MPV

MPV is a proactive caching policy, which caches the "most popular videos" using the (nationwide) video popularity distribution described before. MPV neither updates the caches based on the user requests nor implements any cache replacement policy. The only changes that require cache update are changes in the video popularity distribution. Since the number of videos that are cached depends on the cache size, the performance of MPV in terms of cache hit ratio can be high if implemented for large caches possible for Internet CDNs. However, because of the limited size of the RAN micro-caches proposed in this paper, and because videos requested by active users of a cell may be very different from nationwide most popular videos, the cache hit ratio achieved by MPV policy may not be high when used for RAN micro-caches.

The above implementation of MPV requires retrieval and availability of the (nationwide) video popularity distribution. Alternatively, the cache can infer video popularity information locally by tracking the number of requests to each video after it is first requested. As local MPV retrieves popularity information by keeping track of the number of requests to a video and evicting Least Frequently Used (LFU) videos, we term it LFU [21]. Simply evicting the LFU videos has some drawbacks; for instance, if a video becomes immensely popular over a short period of time only to be quickly forgotten, it will take a long time for the video to be superseded by enough other videos to evict it from the cache—that despite the fact that there is very little interest in the video going forward. Thus, our implementation of LFU caching policy not only keeps track of the number of requests to a video, but also maintains a request counter that is incremented each time a request is made to any video in the cache. Thus, when evicting a video from the cache instead of simply evicting the video with the least number of requests, we evict a video with the lowest ratio between the number of requests to the video and the number of requests to any video for the duration the video has been cached.

B. LRU

LRU [22] is a reactive caching policy, which fetches the video from the Internet CDN and caches it if there is a cache miss. If the cache is full, LRU replaces the video in the cache that has been least recently used. The cache hit ratio of a micro-cache associated with a cell that uses LRU policy depends on the overlap of the video requests of the active users in the cell, and influenced by the degree of overlap of their UPP. The backhaul bandwidth and delay needed to bring videos to the cache will depend on the cache hit ratio since there is no prefetching bandwidth.

C. R-UPP

We propose R-UPP as a reactive caching policy based on the UPPs of the active users in a cell. Fig. 3 depicts the R-UPP caching algorithm. For a video requested that is not present in the cache, R-UPP fetches the video from the Internet CDN and caches it. If the cache is full, R-UPP replaces videos in the cache depending on the UPP of the active users using the LLR

R-UPP

1. **For** new Video Request V
2. **If** $V \in$ Cache
3. Download from Cache
4. **Else**
5. Find cell site UPP based on AUS
6. Calculate P_R for V and the cached videos and generate LLR subset
7. LLR_j subset of LLR videos with least P_R that has to be evicted from cache to fit V
8. **If** $P_R(V) - \sum P_R(LLR_j) > 0$
9. $Cache = Cache + V - \{LLR\}$
10. **Else**
11. Do not cache V
12. **End For, End If, End If**

Fig. 3. R-UPP caching policy algorithm.

P-UPP

1. **If** AUS changed
2. Find cell site UPP based on AUS
3. Calculate request probability P_R based on cell site UPP
4. Calculate MLR and LLR sets based on cell site UPP
5. **For** each video i in sorted list of MLR set, MLR_i
6. LLR_i subset of LLR videos with least P_R that has to be evicted from cache to fit MLR_i
7. **If** $P_R(MLR_i) - \sum P_R(LLR_j) > T_e$
8. Update the cache with MLR_i and evict LLR_j; Update MLR and LLR;
9. **End If, End For, End If**
10. **If** V $\in$ Cache
11. Download V from Cache
12. **Else**
13. Download from Backhaul
14. **End If**

Fig. 4. P-UPP caching policy algorithm.

set introduced in Section II-B, and in case of ties according to the LRU replacement policy. More specifically, when there is a cache miss, we calculate the request probabilities of the videos in the cache as well as the requested video, forming an LLR subset (lines 5 and 6). We calculate the difference between the newly requested video probability and the request probability of the subset of LLR videos from the cache with least P_{R} values that need to be evicted in order to free up space for the new video; only if the difference in request probability is greater than zero, we effectuate the cache update (lines 8 and 9). If multiple videos of the same size and min P_{R} value in the LLR are found and only one requires eviction, we use the LRU policy to select the one to be replaced. The above approach ensures that the cached videos maintain the highest probability of being requested again by the current active users of the cell.

D. P-UPP

We propose P-UPP as a proactive caching policy, which preloads the cache with videos that are most likely to be requested based on the UPP of the active users of the cell. Fig. 4 describes the P-UPP caching algorithm. At the beginning, and every time the AUS changes due to user arrival or departure, video request probabilities are calculated using (3), and videos belonging to the Most Likely Requested set, MLR, are loaded into the cache (lines 2–5). However, if the AUS changes frequently, this proactive policy may lead to high computational complexity and, more importantly, high backhaul bandwidth. Hence, we propose a hybrid solution where the cache is only updated if the expected cache hit ratio improvement due to replacement exceeds a preset threshold.

More specifically, for each video i from the MLR set to be added to the cache, we calculate the difference between its request probability and the request probability 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 in request probability is greater than a threshold, T_e, we effectuate the cache update (lines 5–8). To further improve the P-UPP algorithm from [23] by reducing the risk that cache maintenance downloads cause user requests to be blocked, we allow user requests to temporarily reassign bandwidth that was previously assigned to the cache maintenance downloads. In addition, to ensure that enough bandwidth is allocated to the video session, we promote a maintenance session to a user download if the associated video is being requested by a user while being downloaded for the cache. If on the other hand the cache wants to download a video that is already being downloaded by a video client, it will be copied to the cached as well as the downloading user.

In Appendix-A, we study the time complexity of the proposed caching algorithms in order to evaluate their feasibility of implementation.

While we expect the UPP-based cache policies, R-UPP and P-UPP, to result in higher cache hit ratios than conventional MPV and LRU policies, still all videos not found in the cache need to be fetched from the Internet CDNs, through the wireless channel and RAN backhaul network. In Section IV, we will discuss the implication on video delay, and hence video QoE. We propose a scheduling approach that allocates wireless channel and backhaul resources in a way that increases the overall capacity of the system.

IV. Video-Aware Scheduling for RAN Backhaul and Wireless Channel

For each video request that results in a cache miss, the corresponding video needs to be fetched from an Internet CDN. For proactive policies, MPV and P-UPP, fetching the missed videos is in addition to the videos that need to be fetched proactively to update the cache. Depending on the number of concurrent video requests, the number of cache misses, the number of proactively fetched videos, and frequency of prefetching, the backhaul bandwidth may not be sufficient for all the videos that need to be brought through the backhaul. In addition, all the videos, regardless of where they are being served from (cache or CDN), need to traverse through the wireless channel; hence, the successful scheduling of the video request depends on both backhaul and wireless channel bandwidth availability, of which the latter is dependent on the fading state of all users' wireless channels. Furthermore, a joint scheduling of the backhaul and wireless channel resources is required so that once a video session is scheduled on the wireless channel, the backhaul has put enough data in the (e)NodeB buffer to support a seamless data transfer to the user over the wireless channel. There can be various possible ways of scheduling the video fetches and allocating the backhaul and wireless channel resources. One approach is to try to serve all the pending fetches, which may result in some fetches getting significantly delayed, resulting in unacceptable

video playback delay and stalling. In this paper, we take an alternative approach: Our proposed backhaul and wireless channel scheduler aim to maximize the number of videos that can be served, while ensuring each video served meets its QoE requirements, as will be defined later. If it is not possible to schedule the request with satisfactory QoE, i.e., commence playback before a preset timeout, the request is blocked by the scheduler. If, however, playback has started and served rate drops, video request experience stalling.

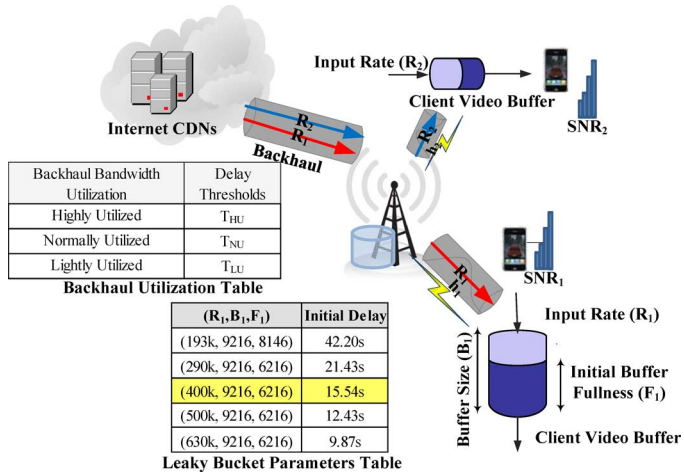
Unlike backhaul scheduling where, once admitted, we can ensure the user's video QoE due to the fixed-rate nature of the channel, scheduling of the wireless channel (also known as RAN scheduling) is best-effort given the time-varying channel condition of each user. Some users may consistently experience bad channel conditions due to cell-edge conditions or fluctuating channel conditions due to fading etc.

Many RAN schedulers have been proposed in the literature with the goal to schedule the requests for the wireless channel. The goal of these schedulers is either to maximize overall throughput of the network or to optimize for user rate fairness or some combination of both. Several video-aware scheduling techniques have been also proposed that consider video frame structures and requirements [24]–[30]. Video-aware schedulers mostly perform rate allocation either to maximize individual user utility functions, sum utility functions, or an overall video QoE goal per system, or allocate video packets or frames to achieve an objective goal, e.g., minimize distortion [30]. However, none of the existing video-aware scheduling techniques addresses initial delay and stalls of a video session, which are the objectives of our proposed scheduling approach. Moreover, existing scheduling techniques do not consider jointly scheduling RAN backhaul and wireless channel resources to maximize end-to-end video capacity. The novelty of our proposed scheduling approach is in: 1) addressing video QoE of scheduled users by using video Leaky Bucket Parameters (introduced later in this section) to determine and meet whenever possible minimum data rates that will satisfy initial delay requirements and minimize stalling; and 2) jointly scheduling RAN and RAN backhaul resources, in the context of RAN cache misses, so as to increase end-to-end video capacity.

In this section, we first define video QoE requirements and video capacity. Next, we describe our proposed scheduling approaches for backhaul and wireless channel, in order to satisfy the QoE requirements of as many video requests as possible, and hence improving end-to-end video capacity under given backhaul and wireless channel bandwidth constraints.

A. Video QoE and Capacity

We consider video QoE as consisting of two aspects: the initial buffering delay the player has to wait before it can start playing and the number of stalls during the video session. The initial playback delay is needed to fill the client buffer to a certain level so to absorb any variations in the network's data transmission rate as well as the variations in video frame sizes to ensure the decoding process can proceed smoothly without any stalls once playback has started.



Backhaul Bandwidth Utilization	Delay Thresholds
Highly Utilized	T_{HU}
Normally Utilized	T_{NU}
Lightly Utilized	T_{LU}

Backhaul Utilization Table

(R_1,B_1,F_1)	Initial Delay
(193k, 9216, 8146)	42.20s
(290k, 9216, 6216)	21.43s
(400k, 9216, 6216)	15.54s
(500k, 9216, 6216)	12.43s
(630k, 9216, 6216)	9.87s

Leaky Bucket Parameters Table

Fig. 5. Example scheduling scenario: LBPs for video requests, backhaul utilization states, and initial delays.

In this paper, we use LBPs to determine the initial delay requirement. In most video coding standards [31], [32], a compliant bit stream must be decoded by a Hypothetical Reference Decoder (HRD) connected to the output of the encoder emulating a decode buffer, a decoder, and a display unit. The Generalized HRD (GHRD) [31] generates LBPs that consist of N 3-tupples (R, B, F) corresponding to N sets of transmission rates and buffer size parameters for a given bit stream. An LBP tuple guarantees that as long as the minimum transmission rate is maintained at R bits/second, 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 and without buffer overflow. Consequently, F/R is the initial delay that the decoder needs to wait to guarantee a stall-free playback. R is the minimum rate required at which the constant bit rate video should be delivered to the user. If a lower R is chosen, while the video playback rate remains constant, more video bits must be buffered (higher F) before playback can start in order to guarantee playback without stalling, and therefore the buffer size requirement increases.

B is the minimum buffer size needed to contain a bit stream without the danger of decoder buffer overflow. In this paper, we assume that B is not a constraint and is large enough even for data rate as low as 10 kb/s and the longest video sequences encountered.

Fig. 5 shows example LBPs associated with a video client buffer and the resulting initial delays. For example, to achieve the initial delay of 15.54 s without stalling, a minimum transmission rate of 400 kb/s is required. The LBP values presented in Fig. 5 are specific to one video, and they differ depending on the characteristics of the video (e.g., low motion versus high motion, duration of the video, etc.). From the example in Fig. 5, we can infer that as the initial delay decreases, the data rate required for transmission increases.

At the beginning of the video session, a video client can use the LBPs for the requested video to request a data rate and select the corresponding initial delay that satisfies its QoE requirements. As shown in Fig. 5, the higher the data rate requested, the less the initial delay. However, if all the video clients greedily

select the highest data rates, there may be unnecessary wireless channel resources consumed, or it may result in congestion in the RAN backhaul, leading to fewer requests served concurrently. Consequently, we need the scheduler to allocate data rates judiciously. Given the RAN backhaul rate constraint, the wireless channel condition, and a distribution of video requests, we define video capacity as the number of concurrent video requests that can be served while meeting each request's QoE requirement (maximum acceptable initial delay, and no or minimal stalling). Our scheduling approach is to seek to maximize video capacity by allocating to each requesting video client the lowest valid LBP bit rate that satisfies its maximum acceptable initial delay, and hence also ensuring no or minimal stalling during the video session. Ideally, a video playback session without any stalling is desired. However, the LBP's minimum rate through the wireless channel cannot be guaranteed during a video session because dynamics in the user's wireless channel condition may make it impossible to achieve the minimum rate and as a result eliminate stalling. Note that as the acceptable initial delay decreases (user QoE requirement increases), the data rate required to meet the QoE requirement increases, and as a result, the blocking probability may increase and capacity decrease. Of course, if the total number of concurrent video requests is low, an increase in the users' QoE may not result in increase in blocking probability.

If a video request is a cache miss, the video should be fetched from the CDN and traverse through the backhaul to reach the (e)NodeB buffer. Hence, for ensuring the video client's QoE, joint scheduling between backhaul and wireless channel is desired. Scheduling of the video through the backhaul is done using the backhaul scheduler, which ultimately delivers the data to the (e)NodeB buffer. The wireless channel scheduler schedules users and transfers the data from the (e)NodeB buffer through the wireless channel. Therefore, although we cannot guarantee that no stalling will occur in the wireless channel, we continue to enforce the no-stalling requirement for scheduling of videos through the backhaul in order to guarantee that if the wireless channel can schedule the users, there are enough data available in the (e)NodeB buffer to transmit through the channel.

B. Backhaul Scheduling Approach

The goal of our backhaul scheduling is to support as many concurrent video sessions as possible while ensuring initial delay below an acceptable threshold and no or minimal stalling. We introduce two backhaul scheduling techniques in this paper: 1) collaborative client and backhaul scheduler where the scheduler recommends an initial delay and video clients request a rate accordingly using LBP; 2) optimization-based scheduler where users request the lowest rate that satisfies their initial delay requirements using LBP and the backhaul scheduler adjusts the data rates dynamically to make use of spare backhaul capacity.

1) Collaborative Client and Backhaul Scheduler: A simple scheduling approach would be for the client to look up the data rate corresponding to, or right below, the maximum delay that it can tolerate and communicate it to the backhaul scheduler, and the backhaul scheduler would try to grant the requested rate. However, this simple approach would deprive the client of lower initial delays when the backhaul has spare bandwidth. The motivation for the collaborative approach is to facilitate better initial delays to the clients when backhaul is lightly utilized while providing higher capacity in highly utilized conditions.

In this approach, we define three backhaul utilization states—lightly utilized, normally utilized, and highly utilized—and associate a maximum delay to each of these states, termed T_{HU}, T_{NU}, and T_{LU}, respectively. At any given time, the backhaul is in one of the above utilization states, depending on the number of videos that need to be fetched through the backhaul (including videos that need to be prefetched by proactive caches). The maximum delay associated with each utilization state is the backhaul scheduler's estimate of initial delays that it can offer to its clients depending upon the utilization state. T_{HU} is set to the maximum acceptable initial delay for the users, as successfully scheduling a video request should amount to satisfying the initial delay requirement. On the other hand, in normally utilized and lightly utilized states, the backhaul scheduler would like to offer lower possible initial delays to the requesting clients. Hence, T_{NU} is set to lower value than T_{HU}, and T_{LU} is set to even lower value than T_{NU}. T_{LU} is set to the delay that results in an average data rate that matches the network expectation of the highest supportable rate. For instance, using analysis of the LBP tables of a representative set of videos, the scheduler can come up with the delay and rate association of the videos. Using empirical association with network QoE and the expected number of potential video requests in the network, the scheduler can predict the data rate required to support the desired QoE. To come up with the empirical association, it is important to consider the video popularity distribution and come up with the adjusted weighted average data rate depending upon the video request probability. T_{NU} value is set to somewhere between T_{HU} and T_{LU} to provide an intermediate step. Note that in general more than three utilization states can be defined and used, with corresponding maximum delay thresholds.

When a video is requested, the backhaul scheduler offers to the client the delay associated with its utilization state. The client chooses a delay from the LBP just below the smaller of the maximum delay offered by the backhaul scheduler and the clients' individual delay requirement and requests the corresponding bit rate. Thus, if the backhaul is in a lightly utilized condition, the client may end up choosing a delay much lower than its initial delay requirements, and hence a more aggressive data rate than the simple approach described above. If the backhaul scheduler cannot grant the requested rate, it notifies the client. Subsequently, the client reverts to selecting a delay at or lower than its maximum acceptable delay, and the corresponding data rate lower than its previous request; if this rate also cannot be granted, the backhaul scheduler blocks the request.

Fig. 5 shows an example of backhaul utilization states and associated delays, assuming video clients within the network require maximum initial delay of 30 s (clients' minimum QoE requirement)—i.e., T_{HU} is set to 30 s, as any delay below 30 s does not satisfy the client's QoE requirements. Assuming the LBP table shown is representative of other videos, meaning delay of 10 s requires bandwidth of around 600 kb/s on average with

an upper and lower bound for majority of videos, T_{LU} is set to 10 s, and subsequently T_{NU} is set to 20 s, a value between 30 and 10 s. As mentioned earlier in this section, the delay of 10 s is just an estimate of delay provided by the backhaul scheduler to support better QoE. Thus, if the delay of 10 s or right below 10 s for a video, according to LBP table, results in higher required bandwidth than the scheduler can grant, the video client requests a delay higher than 10 s and below its maximum acceptable QoE requirement. Consider a video request when the backhaul is in the normal utilization state: The backhaul scheduler offers the client the delay corresponding to the normally utilized state, which is 20 s in this case. The client selects from its LBP an initial delay lower than 20 s, which is 15.5 s, and requests the corresponding data rate of 400 kb/s. However, if the backhaul scheduler does not have the bandwidth to support the 400-kb/s rate, the client will use its LBP to select an initial delay lower than its maximum acceptable initial delay (30 s), which is 21.4 s, and request 290 kb/s bit rate. If this rate also cannot be granted, the scheduler blocks the request.

Note that determining the best delay values associated with each backhaul utilization state may be difficult, given that the LBPs associated with different videos may vary widely, and given that the backhaul spare capacity varies due to new video requests entering the system and existing ones completing. Moreover, this method does not utilize the spare backhaul capacity to finish downloads faster and therefore does not free up bandwidth for future peak demand periods. Hence, we introduce an alternative backhaul scheduler next.

2) Optimization Based Scheduler: In this approach, each video client requests the lowest rate that satisfies its initial delay requirement using the LBP. Based on the requested rate and current loading of the backhaul, the scheduler decides whether to admit or reject the user. To avoid stalling, the scheduling algorithm needs to ensure that the download rate does not fall below the minimum requested rate at any time during the transmission; for this reason, the scheduler refrains from admitting new video requests if there is not enough bandwidth to maintain the minimum required rates of the scheduled requests. On the other hand, if there is additional bandwidth after scheduling all the requested videos, the spare capacity can be used to accelerate the ongoing downloads with the intent to finish downloads faster and free up bandwidth for later use. We formulate the scheduling problem as maximizing an objective function $f(b_i)$, under the constraint that the bandwidth of the ith flow, b_i, should be greater than the initially scheduled (minimum) rate, $R_{\min}^{(i)}$, and the sum of the bandwidth of all scheduled flows that go through the RAN backhaul must be equal to capacity limit, C_1

$$\begin{aligned} \text{Maximize:} \quad & f(b_i) \\ \text{Subject to:} \quad & b_i \geq R_{\min}^{(i)} \qquad \forall i \in F_1 \\ & \sum_{i \in F_1} b_i \leq C_1. \end{aligned}$$

F_1 is the set of flows that go through the RAN backhaul, and $R_{\min}^{(i)}$ is the minimum required rate of the ith video request. The objective function $f(b_i)$ is defined depending on how the additional bandwidth (after allocation of $R_{\min}$) is intended to be distributed among the scheduled video flows. In this paper, since we want to allocate the additional bandwidth equally among ongoing video flows, we use the objective function, $f(b_i) = \max \sum_{i \in F_1} \log(b_i - R_{\min}^{(i)})$. For other ways of allocating the additional bandwidth, $f(b_i)$ can be defined appropriately. For example, for weighted proportional fair allocation, $f(b_i) = \max \sum_{i \in F_1} w_i U(b_i)$, where w_i is the weight given to each allocation and $U(b_i)$ is the utility function for the bandwidth allocated to flow b_i [33]. Alternative to the maximization formulation that we have proposed here, one can define a polytope containing all and the only solutions to the constraints as follows: $\{b_i | b_i \geq R_{\min} \forall i \in F_1; \sum_{i \in F_1} b_i = C_1\}$.

Fig. 6 shows the overall algorithm for rate allocation. Note the above optimization formulation is executed only after all the initial video bandwidth assignments ($R_{\min}^{(i)}$) are decided based on LBP (line 1). Meaning that after any new video request (line 2), we first make sure that the new video request can be admitted based on its LBP and minimum required rate of all existing video sessions (line 3), otherwise the newly arrived video request is blocked (lines 4–6). Only after all the videos receive their $R_{\min}$ (lines 2–8), we execute the maximization to further optimize the rate by using the additional bandwidth (line 9).

Note that the LBP should span useful delay/rate pairs that are the delays that the scheduler/client could be interested in achieving for the initial delay. Intermediate values not directly available in the table may be derived using interpolation of existing table values.

Any video scheduled through the RAN backhaul or found in the (e)NodeB cache needs to further traverse through the wireless channel before reaching the client's buffer. For wireless channel, the rate allocated to a video session cannot be maintained throughout the video session and rate scheduling should be done in small intervals to adapt to changing channel conditions in order to admit as many users as possible. As a result, we cannot apply the backhaul scheduling algorithms we proposed here to the wireless channel scheduler. In Section IV-C, we propose to make RAN scheduling video-aware using the LBPs of the videos, such that the number of videos that can be transmitted from the (e)NodeBs to the clients can be maximized.

C. Video-Aware Wireless Channel Scheduler (VAWS)

In this paper, the wireless channel is assumed small-scale flat fading with log-normal shadow fading path loss modeled according to 3GPP TR 36.814 V0.4.1 [34] Urban Macro (UMa) and Urban Micro (UMi) models. With these models, some users experience good average channel conditions, and some experience degraded channel conditions depending on their specific channel realization, e.g., whether or not there is line-of-sight (LOS) communication, and the distance from the (e)NodeB. These models are congruent with the typical nomadic video usage scenario in cellular networks as users that are located in the vicinity of the (e)NodeB experience a better wireless channel and as a result have higher data rates than the users that are located at the cell edge.

For this paper, we envision an LTE system where the users are assigned subcarriers and power so that there is no dominant interference from the serving cell or neighboring cells. The details of the specific subcarrier allocation (i.e., not just *how many*, but also *which* subcarriers a user is assigned) are outside the scope

Optimization based Scheduler

F_j: set of already scheduled video requests
F_k: set of newly arriving video requests without assigned rate

1. New video requests to select $R_{min}^{(i)}$, min required rate for i^{th} video, using LBP
2. If $F_k \neq \emptyset$ or any download completed
3. Allocate b_i for $\forall i \in \{F_j \cup F_k\}$ s.t. for $\forall i$ $b_i = R_{min}^{(i)}$
4. If $\sum_{i \in F_k} R_{min}^{(i)} + \sum_{i \in F_j} R_{min}^{(i)} \geq C_1$
5. Start blocking new users until $\sum_{i \in F_k} b_i \leq C_1 - \sum_{i \in F_j} b_i$
6. End
7. Move scheduled videos from F_k to F_j. Clear F_k.
8. End
9. Distribute remaining bandwidth to maximize: $f(b_i) = \sum_{i \in F_1} log(b_i - R_{min}^{(i)})$ for $\forall i \in F_j$.

Fig. 6. Optimization-based backhaul scheduling algorithm.

of this work as our main goal is to evaluate the performance of different caching policies under reasonably realistic channel conditions. Without the presence of dominant interference, we can model interference using a band-limited white noise process as proposed by 3GPP [35] and combine it with the thermal noise, which allows us to use a simple rate estimate given by the Shannon capacity formula as

$$R_i = B_i \log_2(1 + \mathrm{SNR}_i). \tag{4}$$

where B_i is the channel bandwidth of the ith user in hertz, SNR_i is the signal-to-noise-ratio for the ith user, and R_i is the achieved rate for user i in bits per second (b/s). Furthermore, like in [36], we assume that the channel state of each user allocated across all tones is equal to power multiplied by square of the user's channel gain and that the SNR can be expressed as

$$\mathrm{SNR}_i = P_i |h_i|^2 / B_i N_0. \tag{5}$$

where P_i is the allocated power for user i, h_i is the channel gain for user i, which includes the combined effect of small- and large-scale fading, and N_0 is the noise and interference power spectral density. The goal is to allocate power P_i and bandwidth B_i to maximize overall cell throughput while satisfying the ith user's minimum rate requirements, $R_{\min}^{(i)}$, which is explained in Section IV-B and is the rate that, if sustained, guarantees a video session without stalling.

The proposed power and bandwidth allocation scheme consists of two phases. The first phase is to attempt to assign enough subcarriers to satisfy $R_{\min}$ of each user, assuming equal power assignment per subcarrier and starting with the video request that has the best channel condition. The second phase is to reallocate power, first to ensure $R_{\min}$ of each user that was allocated subcarriers in the first step, and then using waterfilling to assign the remaining power optimally to the users that were given subcarriers. Optionally, we may again execute the first step to reevaluate the subcarrier assignment based on the power allocation of the second phase of the previous iteration and subsequently assign power—i.e., repeating steps 1 and 2 multiple times to get improved power and subcarrier assignment. The second iteration is to address the case, where some users are assigned so much power by waterfilling in the second phase of the previous iteration that they are able to meet the minimum rate requirement with fewer subcarriers than were first assigned. The iterative process does not result in any gain if all users are assigned subcarriers during the first phase of the previous iteration or if there is no excess power to redistribute using waterfilling in the second phase of the previous iteration.

Because the channel is flat fading, subcarriers can be assigned independently for each user—i.e., no subcarrier performs better or worse for a given user, so we only need to determine *how many* subcarriers to assign. Had the channel been frequency-selective, we would also have to decide *which* carriers to assign.

With assumption of equal power allocation per subcarrier assigned to video request i, we can derive the number of subcarriers assigned to a video request as follows:

$$n_{\mathrm{sub}}^{(i)} = \left\lceil \frac{R_{\min}^{(i)}}{B_{\mathrm{sub}} \log_2 \left(1 + \frac{P_{\mathrm{sub}}^{(i)} |h_i|^2}{B_{\mathrm{sub}} N_0}\right)} \right\rceil. \tag{6}$$

We use $P_{\mathrm{sub}}^{(i)}$ to denote the power per subcarrier for the ith user, which is initially set to the total power constraint, $\bar{P}$, divided by total number of subcarriers, N_{sub}. The bandwidth of each subcarrier is B_{sub}, which is equal to total bandwidth divided by total number of subcarriers. Subcarriers are assigned to the video requests with the best channel conditions first and until the total number of available subcarriers has been assigned. If there are unused subcarriers available, after all video requests have been assigned enough subcarriers to meet the minimum rate requirements, the remaining subcarriers are divided in a round-robin fashion starting with the user with the best channel.

In second phase, to refine the initial uniform power allocation, since there is no dominant interference between users, we can use (4) and (5) to allocate the power corresponding to the minimum rate requirements for each user independently of other users

$$P_{\min}^{(i)} = \frac{B_i N_0}{|h_i|^2} (2^{R_{\min}^{(i)}/B_i} - 1). \tag{7}$$

In the above equation, $R_{\min}^{(i)}$ is user i's minimum rate requirement, and $P_{\min}^{(i)}$ is the required minimum power to achieve that rate.

Because of the first phase, we know that (7) will either keep or reduce the allocated power per subcarrier for each user. After assigning $P_{\min}^{(i)}$ to each user, whatever power remains unassigned will be allocated optimally using waterfilling algorithm.

Fig. 7 shows our video-aware wireless channel rate allocation and admission algorithm. In the event of a new video request (without loss of generality, we assume arrival of one video request at a time), a limited handshake between wireless channel and backhaul is envisioned to ensure that neither backhaul nor wireless channel bandwidth is wasted for a request that is blocked in the other resource. For each new video request, if there is enough backhaul bandwidth available to schedule the video through the backhaul, we add the video to $List_{\mathrm{buffer}}$ which keeps track of the videos that are in initial buffering stage. Otherwise, we block the request due to insufficient backhaul resources. Other than the above initial handshake,

Video Aware Wireless Channel Scheduler (VAWS)

$List_{buffer}$: List of video requests that are in buffering mode
$List_{playback}$: List of video requests that are in playback mode
$List_b$: List of videos that have been removed from the minimum rate allocation list in a specific scheduling interval; this might be because the user temporarily experiences a bad channel condition

// For new video request V_i and all the videos in playback or buffering mode, that have data ready for transmission in the (e)NodeB buffer, follow the algorithm below

Execute the following part at fixed intervals
1. Initialize per-sub-carrier power to $\bar{P}/N$

Phase 1: Subcarrier Assigment
2. Assign subcarrier to satisfy R_{min} of each download, assuming known per-user per-subcarrier power assignement using eq. (6); starting with the video request with the best channel condition.
3. Assign remaining subcarriers using Round Robin

Phase 2: Power Assignment
4. Reallocate power, first to ensure R_{min} of each user given subcarrier assignment from previous step using eq. (7)
5. Calculate $P' = \bar{P} - \sum_{j=1}^{N} P_{min}^{(j)}$; $N = |List_{playback}| + |List_{buffer}|$
6. **If** $P' > 0$
7. Allocate P' using waterfilling algorithm among N ongoing video sessions
8. **End If**

Refinement Step
9. Repeat Phase 1 assuming per subcarrier power allocation derived from phase 2
10. Repeat Phase 2 assuming new subcarrier allocation from step 9
11. For all the video downloads calculate resulting rates, R_{inst}^{i}

12. **For** all videos from $List_{buffer}$
13. Let T_{init}^{i} be the time spent to achieve initial buffer fullness for video i according to the LBP
14. **If** $T_{init}^{i} > T_e$
15. Block the video request and remove it from $List_{buffer}$
16. **Else**
17. Add the video request to $List_{playback}$
18. **End If, End For**
19. **For** each ongoing video, $V_i \in List_{playback}$
20. Let $R_{avg}^{i} = (1-\alpha)R_{avg}^{i} + \alpha R_{inst}^{i}$
21. **If** $R_{avg}^{i} \leq \beta_1 R_{min}^{i}$
22. Add V_i to the list of videos that experience blocking
23. **End If**
24. **If** $R_{avg}^{i} \geq \beta_2 R_{min}^{i}$ & $|List_b| \neq 0$
25. Remove i from subcarrier and power allocation during next scheduling interval
26. **End if, End For**

Fig. 7. Video-aware wireless channel scheduling algorithm.

the data exchange between backhaul and wireless channel is through the (e)NodeB buffer.

We execute the wireless channel scheduler at fixed intervals (e.g., every 10 ms) to derive the subcarrier and power allocation for all videos in the playback or buffering mode.

As explained earlier in this section, we assign subcarriers in phase 1 (lines 2 and 3) assuming equal power allocation using (6), and we refine the power allocation in phase 2 (lines 4–8). In the second phase, we first allocate power to satisfy $R_{\min}$ for all video requests using (7) (line 4). After minimum power allocation, the remaining power, P', is allocated among ongoing video requests, which were assigned subcarriers in the first phase, using waterfilling algorithm [36], [37] (lines 5–8). After allocating excess power using waterfilling, some users may end up having so much power per subcarrier that their min rate requirements can be met with less subcarriers than were initially given by Phase 1. Thus, as an enhancement, we repeat Phases 1 and 2 twice to refine the allocation (lines 9 and 10). Once the power allocation is finalized, we calculate the instantaneous channel rate, $R_{\text{inst}}^{(i)}$ for all ongoing video requests (line 11).

After rate allocation, during the same scheduling interval, the video requests that were not able to fill their initial video buffer T_{init} in a timely manner are blocked and removed from the $List_{\text{buffer}}$. Video sessions that can fill their initial buffer within acceptable time (T_e) are promoted to $List_{\text{playback}}$ (line 17), and their video playback commences.

As explained at the beginning of this section, we measure video QoE using the initial delay and probability of stalling; furthermore, we designed the backhaul scheduler so that once a video request is admitted with a rate of $R_{\min}$, the achieved backhaul rate is at least $R_{\min}$ for the entire video session. However, because of the inevitable variations in the wireless channel, a similar rate guarantee for the wireless channel is not feasible. We use an IIR filter to estimate the average rate achieved from the instantaneous channel rate in a way that takes into account the history of the rate allocation (line 20). We classify the video downloads based on the average achievable rate. In line 21, β_1 ($\beta_1 \leq 1$) is a threshold used to identify the users that achieve a rate below factor of $R_{\min}$. This threshold is used to identify the users that may experience stalling depending on the value of β_1. To quantify stalling as a performance metric, we model the UE's decoding buffer by adding bits to it at the scheduled rate while subtracting bits corresponding to the actual frame sizes of the video being played back, so that any buffer underflow accurately indicates video playback stalling.

In addition, to avoid unfair allocation of rate—i.e., some video requests achieve average rates of well above their $R_{\min}$, while other video requests in playback mode are blocked frequently (i.e., $|List_b| \neq 0$) and experience stalling—we temporarily suspend video requests with $R_{\text{avg}}^{(i)}$ above $\beta_2 R_{\min}^{(i)}$ in subsequent scheduling intervals until the average rate drops below $\beta_2 R_{\min}^{(i)}$ (lines 24–26). Next, we explain how the backhaul and wireless channel schedulers coordinate with each other in our end-to-end video delivery system.

D. Joint Backhaul and Wireless Channel Scheduling

As explained in [38], the study of multiple access systems (wireless channel scheduling) is typically done assuming a full-buffer model—i.e., the assumption that once the wireless channel scheduler is about to schedule a user, there is enough content in the (e)NodeB buffer to transmit to the user without interruption at any data rate. Another more realistic model is a finite traffic model; we assume such a system with a buffer for each user at the (e)NodeB where the data from the backhaul is stored pending transmission through the wireless channel by our VAWS as explained in Section IV-C. Upon a video request, using the video's LBP table, the UE requests bit rate $R_{\min}$ that satisfies its initial delay requirements. Subsequently, wireless channel and backhaul schedulers try to meet the rate requirement if the request is not blocked. The backhaul grants the request if there is enough backhaul bandwidth available to

support the requested rate, and it delivers the video bits to the (e)NodeB buffer, pending transmission through the wireless channel. If no data are available in the (e)NodeB buffer, the wireless channel scheduler cannot serve the UE during the corresponding scheduling interval regardless of the channel conditions. Joint backhaul and wireless channel scheduler cooperates in two error conditions: 1) if the backhaul scheduler blocks a video request due to the lack of backhaul resources, VAWS also blocks the video request; 2) if VAWS blocks a video request during the initial buffering, the associated backhaul resource is released as well; 3) if multiple video requests are pending admission and the system is overutilized, video requests associated with the UEs with better wireless channel are admitted first. The VAWS can obtain information about the UE's wireless channel state through the measurements that the UE performs and periodically feeds back to the (e)NodeB. After the admission of a user, we make sure that the backhaul scheduler maintains $R_{\min}$, as explained in Section IV-A, so that once VAWS is ready to schedule a UE, there is enough data in the (e)NodeB buffer to ensure uninterrupted scheduling of the wireless channel.

V. Simulation Framework and Results

A statistical Monte Carlo discrete event simulation framework was developed using MATLAB to compare the relative performance of the caching policies, as well as to validate the effectiveness of the proposed backhaul and wireless channel scheduling techniques. We use Monte Carlo discrete event simulation, where the implementation consists of a number of iterations where the innermost loop corresponds to one video request per iteration that is being evaluated for all the cache policies. There is an outer loop over a set of different simulation parameters (for instance, cache sizes or user interarrival time), and finally, the outermost loop repeats the entire simulation using a new set of inputs for increased statistical significance. All the cache policies are evaluated over multiple trials of the same configuration with randomized video popularity, video category, video size, user UPP, request generation, etc., generated from the specified distributions in Table I so that the observed statistics (capacity, hit ratio, etc.) and differences between these statistics for different caching policies are statistically significant with 95% confidence interval. Although we do not show the lower and upper ranges associated with the 95% confidence interval in the figures when presenting results, using multiple trials, we have ensured that the achieved confidence interval for all our simulation results does not exceed $\pm 1\%$ of the estimated statistic. Confidence interval of 95% indicates that if the experiment is repeated, the results can be reproduced within $\pm 1\%$ of the estimated statistic 95% of the time [39]. Next, we explain our simulation parameters and present the results.

Table I lists the parameters used for our base set of simulation results—base scenario. The base scenario reflects realistic system configuration and video requests by using distributions or simulation parameters that have been obtained by other research [18]—e.g., through Internet measurements, network monitoring, or marketing research [20], [41]. To ensure simulation speed, we restrict the total number of videos, available for download, to 100 000, distributed uniformly across 250 video categories and following a Zipf popularity distribution with $\alpha = 0.8$ (to model video distribution according to [18]). The video duration is exponentially distributed, with the frame size distribution as reported in [40], with mean of 8 min and truncated to a maximum of 30 min and a minimum of 2 min. We assume the video codec bit rate is uniformly distributed between 200 kb/s (QVGA quality) and 2 Mb/s (HD quality). The simulation assumes 5000 potential mobile users with Poisson arrival and departure with mean interarrival time of 40 s and user active time of 45 min (time the user is present in a cell whether actively downloading video or not). Video clients' maximum acceptable initial delay is set to 30 s. We use an M/M/∞ queuing model to estimate the number of active users [42]. The total number of concurrent active users follows Little's theory given by: $N_{\text{active}} = \lambda_{\text{u}} W$, where $1/\lambda_{\text{u}}$ is mean user interarrival time and W is the user active time. Denoting the mean video request interarrival time per user to be $1/\lambda_{\text{v}}$, the expected mean video request interarrival time, N_{v}, is given by $N_{\text{v}} = \lambda_{\text{v}} N_{\text{active}}$. To generate a video request, a user is selected randomly from the AUS, and a video request is generated based on the user UPP and the popularity ranking of videos. For the results reported below, we assumed a backhaul bandwidth of 100 Mb/s, and the micro-cache size varies between 10–200 GB.

TABLE I
Simulation Parameters

Variable	Distribution/Parameters Value
Total Number of Video Requests	100,000
Total Number of Videos	100,000
Video Popularity Distribution	Zipf, $\alpha = 0.8$
Video Frame Size Distribution	As proposed in [40]
Number of Video Categories	250
Video Size	Exp, min=2, mean=8, max=30(min)
Video Bit Rate	Uniform, 200kbps(QVGA), 2Mbps(HD)
Total number of mobile users	5,000
UPP Distribution Across VCs	Exponential, mean=2
User Arrival/Departure Model	Poisson: User inter-arrival time=40s ($1/\lambda_u$), User active time= 2700s (N_{active})
Video Request Arrival	Poisson, Mean inter-arrival time per user = 480s
Max Acceptable Initial Delay	30 seconds
Cache Size	10Gbytes to 200Gbytes
Backhaul Bandwidth	100 Mbps

Next, we study the effectiveness of the UPP-based policies in improving cache hit ratio, user QoE (initial delay and playback with limited stalling), and capacity. We compare the UPP-based policies with LRU, LFU, and MPV using the base simulation parameters as well as variations of some of the more significant simulation parameters like Zipf, user dynamics (i.e., mean user interarrival time and user active time), and user UPP distribution (how biased the user requests are toward specific video categories). As both MPV and P-UPP are proactive caching policies, in the simulations we present in this section, we preload them respectively with the most popular videos and the most likely videos to be requested based on initial AUS. Later in this section, we compare our video-aware wireless channel scheduler with scheduling without consideration for video QoE and

finally present the end-to-end capacity results. Furthermore, we analyze robustness of our caching algorithms in Section V-G.

A. Performance of the Caching Policies in Base Scenario

Fig. 8(a) shows the performance of the different cache policies in terms of cache hit ratio achieved for a given cell for different cache sizes with the base scenario configuration. It is evident that the UPP-based cache policies perform significantly better than the conventional cache policies for all cache sizes. For example, when the cache size is 200 GB, P-UPP and R-UPP achieve cache hit ratios of 0.71 and 0.68, respectively, while the LFU, LRU, and MPV policies achieve cache hit ratios of 0.61, 0.58, and 0.35, respectively. Note that though Fig. 8(a) shows P-UPP and R-UPP achieving similar cache hit ratios, from other simulation results, with different parameters like lower P-UPP update threshold (Section III.D), not presented in this paper due to space limitation, we observe that P-UPP can perform up to 6 percentage points better than R-UPP in terms of cache hit; this, however, comes with the cost of high required backhaul bandwidth.

Fig. 8(b) shows the mean RAN backhaul bandwidth required by different policies in the base simulation. For example, with cache size of 200 GB, we require 28 Mb/s backhaul bandwidth for P-UPP, 31 Mb/s for R-UPP, 38 Mb/s for LFU, 40 Mb/s for LRU, and around 62 Mb/s for MPV cache policy. Note that if there were no video caching at the edge of the RAN [no cache in Fig. 8(b)], the backhaul bandwidth needed to bring all the requested videos would be 94 Mb/s. From the results, we can infer that UPP-based caches significantly reduce the backhaul loading.

B. Effect of Zipf Parameter on Cache Performance

Fig. 8(c) shows the performance of different cache policies in terms of cache hit ratio achieved when α is 0.6. In this scenario, the tail of the Zipf distribution is fatter than in the base scenario, and the most popular videos are less popular than the base scenario when α is 0.8. With this parameter change, given the same cache size as the base scenario, the expected cache hit ratio is going to be lower.

For instance, from Fig. 8(c), the cache hit ratio of MPV degrades by 15 percentage points from 0.35 to 0.20 for cache size of 200 GB when α changes from 0.8 to 0.6. However, the cache hit ratio of P-UPP and R-UPP decreases by 12 and 11 percentage points, respectively, and LRU and LFU by 15 percentage points compared to the base scenario. We can infer from the results that the cache hit ratio of MPV is most dependent on the MPV distribution and MPV caching policy is more effective for higher values of α.

Fig. 8(d) shows the mean backhaul bandwidth required versus cache size when $\alpha = 0.6$. For a cache size of 200 GB, we require 40 and 41 Mb/s backhaul bandwidth for P-UPP and R-UPP, respectively, 52 Mb/s for LFU, 54 Mb/s for LRU, and 78 Mb/s for MPV. Due to the lower cache hit ratio across all the policies with $\alpha = 0.6$ compared to the base scenario, we can infer that the required backhaul bandwidth for all the policies increases, but MPV is affected the most.

C. Effect of User UPP Distribution on Cache Performance

Since the base scenario assumes exponential cell site UPP, in this section we want to also study what happens when the cell site UPP is not biased, for instance, when the cell site UPP is uniformly distributed across the VCs, where it becomes more difficult to anticipate what videos the user may request.

Fig. 8(e) shows the impact of the UPP distribution on the performance of the caching policies. Uniform cell site UPP implies that although each user still has personal preferences for certain video categories, as a whole, the cell site UPP will be unbiased and users may request a video from any of the 250 video categories with equal probability. The cell site UPP distribution does therefore not contain any information that can help determine which video is more likely to be requested. From Fig. 8(e), it is evident that the P-UPP and MPV caching policies perform better than the remaining cache policies when the UPP is uniform. The cache hit ratios of P-UPP and MPV policies are the same at around 0.37, while the cache hit ratios of R-UPP, LFU and LRU are 0.27, 0.25, and 0.21, respectively. We can infer that although the cache hit ratio of P-UPP has dropped by 34 percentage points from the baseline, the users' UPP does not have any effect on MPV cache hit ratio. This is because MPV ignores cell site UPP.

Fig. 8(f) shows the mean backhaul bandwidth required by the different policies. As expected, when the cache hit ratio decreases, the mean backhaul bandwidth required increases. For example, with cache size of 200 GB, we require 61 Mb/s backhaul bandwidth for P-UPP and MPV, 71 Mb/s for R-UPP, 73 Mb/s for LFU, 75 Mb/s for LRU, and around 94 Mb/s for no cache. Note that if there were no video caching at the edge of the RAN (no cache in Fig. 8(f)), the backhaul bandwidth needed to bring all the requested videos would be 94 Mb/s; thus, by caching even with uniform UPP, we can reduce backhaul bandwidth by about 35%. From the results, we see that with uniform user UPP, the P-UPP caching policy is superior to the no-cache and LRU caching policies, and that uniform user UPP results in degradation of LFU, LRU, and R-UPP. In the case of uniform cell site UPP, P-UPP and MPV effectively become identical because no local video category preference exists, so it is expected that they perform identically.

D. Effect of User Dynamics on Cache Performance

User dynamics in a cell (how frequently potential video users enter a cell site, and for how long they stay active within that cell) may result in frequent or seldom changes to the cell site UPP; depending on the cache policy, frequently changing cell site UPP may require more cache updates than an infrequently changing cell site UPP. Hence, we wanted to study the impact of high user dynamics on the performance of the cache policies. Fig. 8(g) and (h) shows the performance of the caching policies under high user dynamics. To stimulate high user dynamics, we reduced the user interarrival time from 40 to 10 s and mean user active time from 2700 to 360 s. In addition to smaller user interarrival time, this setup results in the total number of users in the system decreasing from 67 to 36 users, and as a result, the user arrivals and departures result in larger deviations from the mean

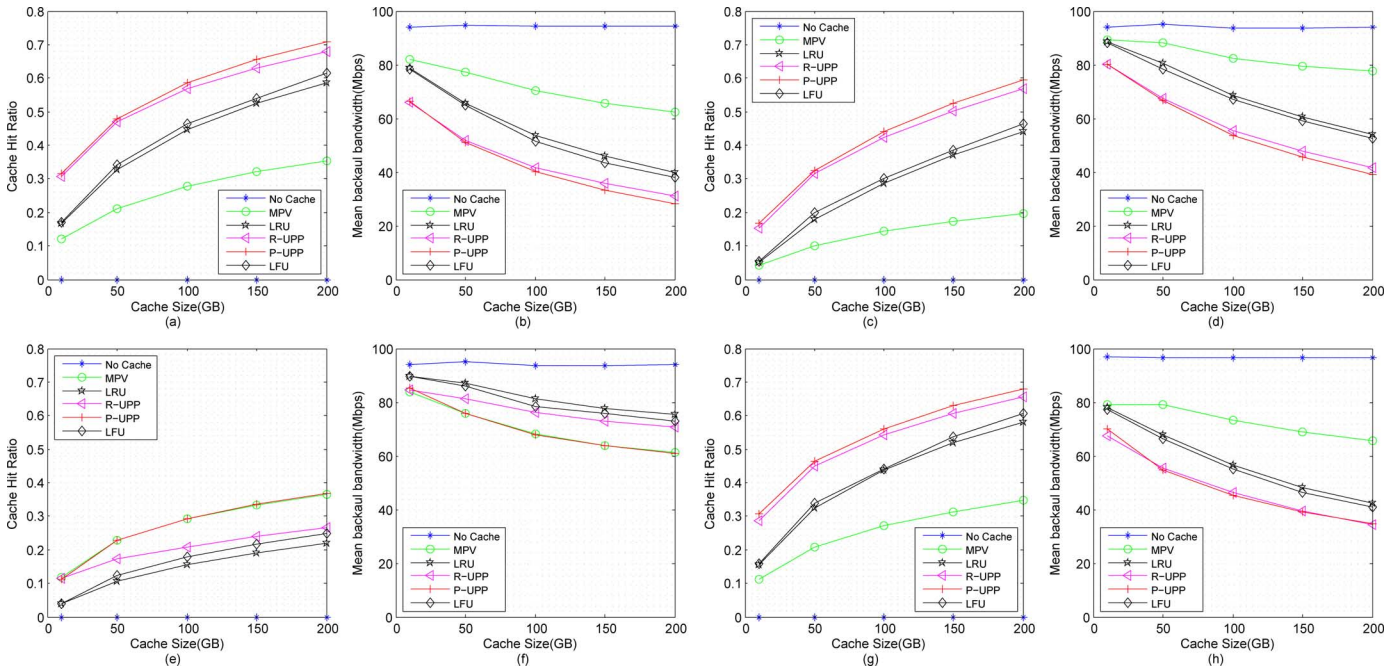


Fig. 8. Performance of the caching policies under base scenario and significant simulation parameters: (a) Cache hit ratio versus cache size (base scenario); (b) mean backhaul BW required versus cache size (base scenario); (c) cache hit ratio versus cache size (Zipf parameter scenario, $\alpha = 0.6$); (d) mean backhaul BW required versus cache size (Zipf parameter scenario, $\alpha = 0.6$); (e) cache hit ratio versus cache size (UPP distribution scenario, uniform UPP distribution); (f) mean backhaul BW required versus cache size (UPP distribution scenario, uniform UPP distribution); (g) cache hit ratio versus cache size (user dynamics scenario, high dynamics); (h) mean backhaul BW required versus cache size (user dynamics scenario, high dynamics).

cell site UPP. To be fair, we captured the caching policies' effectiveness under high user dynamics scenario, while ensuring that the total number of video requests to the cache—$N_{\mathrm{v}} = \lambda_{\mathrm{v}}\lambda_{\mathrm{u}}W$ as explained in the beginning of this section—is comparable to the base scenario; thus, we decreased the video request generation interval per user from 480 to 250 s.

Fig. 8(g) shows that the user dynamics do not significantly affect the cache hit ratio compared to the base scenario. For example, when the cache size is 200 GB, P-UPP and R-UPP achieve cache hit ratios of 0.68 and 0.65, respectively, and LFU, LRU, and MPV policies achieve cache hit ratios of 0.61, 0.58, and 0.35, respectively. Compared to the cache hit ratio of the base scenario, we infer the high user dynamics degrade the cache hit ratio of the P-UPP and R-UPP by 3 percentage points. The cache hit ratio of all the other policies remains the same. Although we expected negative impact on UPP-based policies, due to the further optimization of the caching policies that we explained in Section IV, we have reduced the side effects of the user's dynamics.

Next, we analyze the impact of high user dynamics on the mean backhaul bandwidth required by different cache policies. From Fig. 8(h), with cache size of 200 GB, we require mean backhaul bandwidth of 35 Mb/s for P-UPP and R-UPP, 41 Mb/s for LFU, 42 Mb/s for LRU, and around 65 Mb/s for MPV caching policy. Note that if there were no video caching at the edge of the RAN, the backhaul bandwidth needed to bring all the requested videos would have been around 96 Mb/s. We can infer that the mean backhaul bandwidth required increases by 25% for P-UPP, 13% for R-UPP, 8% for LFU, 5% for LRU, and 4% for MPV compared to the base scenario. This increase in backhaul bandwidth requirements for P-UPP is due to additional required updates to proactively update the cache as the average cell site UPP changes frequently.

E. *Effect of Caching on Capacity and Achieved Delay*

To better understand the impact of RAN caching and our proposed policies on the capacity of the wireless network, we performed a set of experiments to measure the capacity for different cache sizes when ensuring the user's QoE, here initial delay.

Fig. 9(a) shows capacity versus cache size with base scenario configuration except for the user interarrival time. Note that each point in this graph captures the case where the blocking probability is exactly 0.01, which is achieved by changing the user interarrival time such that the steady-state target blocking rate is achieved and noting the number of concurrent video requests generated at that specific user interarrival time. For instance, for cache size of 200 GB, the capacity is 103 concurrent videos served in the cell without RAN caching, 189 with MPV, 275 with LRU, 346 with R-UPP, and 419 with P-UPP. We can infer from Fig. 9(a) that P-UPP performs about 21% better than R-UPP, about 52% better than LRU, about 122% better than MPV, and 300% better than when there is no RAN caching. The superiority of P-UPP in terms of capacity is due to the high cache hit ratio that it can achieve.

Fig. 9(b) shows the cumulative distribution function (CDF) of video delivery delay when the cache size is 200 GB and user interarrival time is 12 s, which corresponds to a heavily loaded system. Given that the user load is identical, but caching performance differs, the associated blocking probability is different for each caching policy. We register an infinite delay for all the blocked video requests, which results in the CDF not reaching 1.00 within the displayed delay region. From other simulation

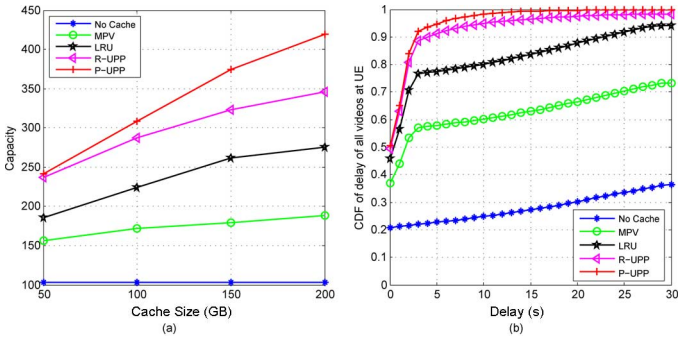


Fig. 9. Performance of the caching policies: (a) capacity versus cache size; (b) CDF of delay of all videos at UE.

results, not presented in this paper, if we measure the delay for each cache policy when the blocking probability is 0.01—i.e., at different number of concurrent video requests—the delay for all the policies is similar, but the number of users achieving that delay is significantly higher for the cache policies with high capacity. This shows that improved caching performance will result in either higher peak capacity or a reduction of delay when operating below capacity limits. From Fig. 9(b), we infer that the probability of achieving an initial delay of, for instance, 5 s or less is about 0.23 when no RAN cache is available, 0.58 for MPV, 0.77 for LRU, 0.91 for R-UPP, and 0.95 for P-UPP. These results show that using micro-caches at the RAN greatly improves the probability that video requests can meet initial delay requirements, in particular when the desired initial delay is low. The results also reveal the superiority of the UPP-based policies, compared to conventional caching policies, in achieving better initial delay given higher capacity. From Fig. 9(a) and (b), we infer that given the comparable achieved delay, we can support significantly more concurrent video requests using the UPP-based policies, and when similar number of videos sessions are concurrently active, we achieve a lower delay.

From the results, not presented here due to space constraint, we also concluded that the system load does not affect the cache hit ratio—meaning an increase in the number of requests to the cache does not result in higher cache hit ratio. However, the mean required backhaul bandwidth increases toward the backhaul bandwidth capacity, and video requests experience increased delay due to the high network utilization. In addition, some video requests are blocked as the initial buffering delay requirement cannot be met.

F. Effect of Wireless Channel Scheduler

So far, we have assumed unlimited wireless channel bandwidth when analyzing RAN cache performance. In this section, we consider realistically constrained wireless channel conditions and study the effectiveness of our proposed VAWS plugin (Section V-C) by comparing with the results of scheduling the wireless channel without consideration for video QoE.

The path-loss model for the wireless channel that we used for this work follows 3GPP TR 36.814 V0.4.1 UMa and UMi models. In this section, we primarily present results using the UMa model, and provide a summary of results obtained using the UMi model. The small-scale fading model is simplified to the case of single-path static channel. The wireless channel specific parameters that are used for the simulations are listed in Table II.

TABLE II
WIRELESS CHANNEL 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$_{10}$(20MHz)
Noise figure	7 dB
Interference Margin	5.5 dB
Cell Radius	1.2 km

To understand the impact of VAWS on the end-to-end capacity of the wireless network, we repeated the same experiments reported in Fig. 9(a), but this time with additional wireless channel constraints for no RAN caching and RAN caching with the conventional and UPP-based caching policies. As was the case for the backhaul-only scheduler, capacity is defined as the number of concurrent video requests served when the blocking probability is 0.01. However, unlike the previous experiments reported in Fig. 9 where the blocking is only due to the lack of backhaul resource, in this case blocking can be due to either lack of backhaul or wireless channel resources. Fig. 10(a) shows the end-to-end capacity as a function of cache size, with and without the use of video-aware wireless channel scheduling in addition to the use of backhaul scheduling. To avoid cluttering the figure, we report results using one conventional caching policy (LRU), one UPP-based policy (P-UPP), and without RAN caching. Here, the maximum allowed initial delay is 30 s; any video request experiencing delay of more than 30 s is blocked. When no RAN caching is used, the end-to-end capacity is 102 concurrent video sessions. The end-to-end capacities when using P-UPP and LRU caching policies with cache size of 200 GB, along with waterfilling-based wireless channel scheduling (no VAWS), are 292 and 264, respectively. End-to-end capacity can be significantly improved by using VAWS, from 292 to 357 in the case of UPP caching. As shown in Fig. 10(a), we can see an improvement of 22% in terms of concurrent video requests served using VAWS. The results presented in Fig. 10(a) indicate that while the constrained wireless channel limits the end-to-end capacity for P-UPP [compared to results in Fig. 9(a)], our video-aware wireless channel scheduler reduces the negative impact in terms of video requests served. Note that when no RAN caching is used, the backhaul limits the end-to-end capacity of the network, so the impact of the wireless channel limitations is minimal. Next, we study the effect of wireless channel scheduler, together with RAN caching, on initial delay that users experience.

Fig. 10(b) shows the CDF of the delay when the cache size is 200 GB and user interarrival time is 12 s. Here, unlike Fig. 9(b), the results are affected by the limitations in wireless capacity and impact of different wireless channel scheduling algorithms. As before, we register an infinite delay for each blocked video request. As expected, we observe lower capacity and increased delay when considering wireless channel limitations, but we also note that by choosing VAWS algorithm, we can recoup much of the loss and approach the results observed when not

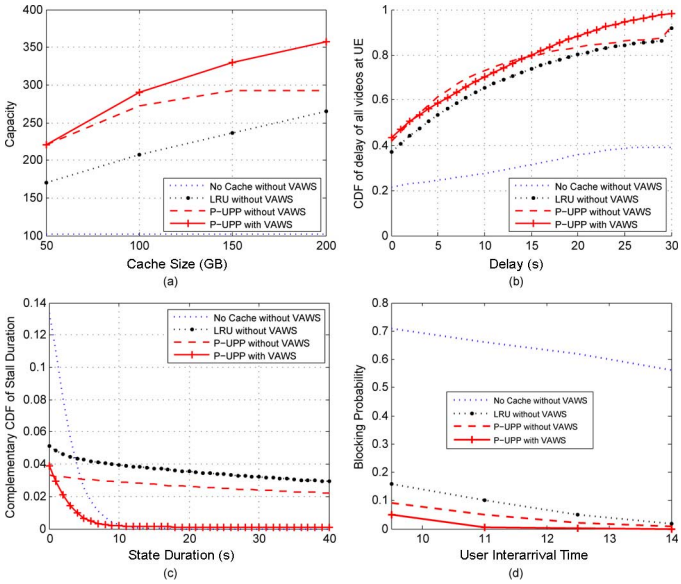


Fig. 10. Performance of video-aware wireless channel scheduler: (a) End-to-end capacity versus cache size; (b) CDF of delay of all videos at UE; (c) complementary CDF of stall duration; (d) blocking probability.

considering wireless channel limitations. From Fig. 10(b), we infer that the probability of achieving a delay that exceeds 25 s with P-UPP caching algorithm is 0.05 with VAWS and 0.14 without VAWS. The probability of achieving a delay of 25 s or more with LRU caching policy without VAWS is 0.16, and without any RAN cache and without VAWS is 0.61. We see an improvement of 56 percentage points in terms of reducing the number of video requests experiencing delay of more than 25 s with P-UPP and VAWS compared to no cache and conventional scheduler.

Fig. 10(c) shows complementary CDF of stall duration for cache policies under study. From the figure, we can see that using VAWS significantly reduced the probability of stalling and stall duration. For example, we see that the probability of stall with delay of 10 s or higher is almost 0 for P-UPP with VAWS, while it is around 0.03 for P-UPP caching policy without VAWS and 0.04 with LRU without VAWS. The no-cache case does not experience any stall with duration of above 20 s because blocking happens exclusively in the backhaul, which does not lead to increased delay, but instead reduces the overall capacity. However, when not using RAN caching, even though we are limited by the RAN backhaul and operating well below the capacity of the wireless channel due to the aggressive scheduling of water-filling algorithm that optimized for users' channel conditions and total throughput, we still see a high number of stalls with short durations. Our results show that using P-UPP with VAWS can improve the total number of stalls with delay of 5 s or higher by 1600% compared to P-UPP without VAWS.

Fig. 10(d) shows the effect of different user interarrival times, caches, and VAWS on blocking probability. As expected, the blocking probability decreases with increasing user interarrival time. More importantly, significant reduction in blocking probability can be obtained by using P-UPP caching with VAWS, compared to other alternatives. For example, for user interarrival time of 9.5 s, the blocking probability is 0.71 with no

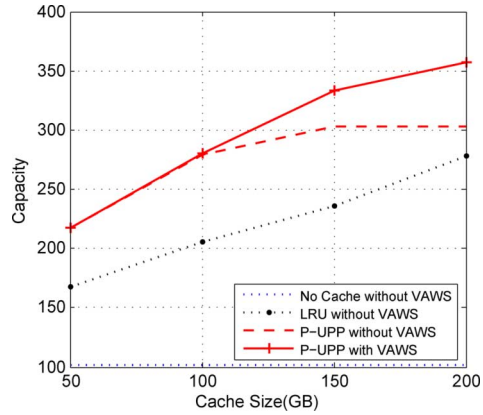


Fig. 11. Capacity versus cache size for UMi wireless channel model.

RAN caching, 0.16 with RAN caching using LRU, and 0.05 and 0.09 for P-UPP with and without VAWS, respectively. Our simulation results show that RAN micro-caches with the proposed UPP-based caching policies, together with the proposed video-aware scheduling approaches improve the capacity by up to 247% over the no-cache, no-video QoE-based wireless scheduling case while maintaining similar or better QoE.

While the results presented above were obtained using the UMa channel model, we have also studied the performance of our proposed techniques using the UMi model. Fig. 11 shows that using the UMi channel model and the particular cell radius of 1000 m, the end-to-end capacity obtained by each caching policy with and without VAWS is comparable to the end-to-end capacity obtained by the same caching policy and VAWS combination using UMa model and 1200-m cell radius. The end-to-end capacities when using P-UPP and LRU caching policies with cache size of 200 GB, along with water-filling-based wireless channel scheduling (no VAWS), are 303 and 278, respectively. End-to-end capacity can be significantly improved by using VAWS, from 303 to 357 in the case of P-UPP caching. These results show that the conclusions for UMa model hold true for UMi as well.

G. Robustness Analysis

To study how the UPP caching algorithms perform when the exact UPP distribution of the video users in a cell site is not available, we introduced three different types of UPP estimation error and summarized the result of our simulations here. The first scheme assumes that the VC rankings in a UPP remain the same but the probability changes by a random amount. To generate this type of error, we multiply the selection probability of each VC by $(1+e)$, where e is uniformly distributed on the interval $[-\epsilon;\epsilon]$. Introducing this type of error to our base scenario yields similar results to those presented in Fig. 8(a) and (b), and our caching algorithms are robust in response to this type of errors. The second type of error introduced was inspired by the Kendall tau-distance [43] and swaps the ranking of adjacent VCs, selecting adjacent pairs at random and swapping their popularity ranking. The simulation results showed that our UPP-based caching algorithms are very robust to this type of error as well. Third, we introduced an error where the user with some probability requests a video from a video category that is not

TABLE III
ROBUSTNESS ANALYSIS OF THE CACHING ALGORITHMS

Probability of selecting unexpected VC	**0.3**	**0.1**	**0.05**	**0**
P-UPP Cache Hit Ratio	0.59	0.66	0.69	0.71
R-UPP Cache Hit Ratio	0.55	0.64	0.66	0.68

expected given the users preference profile. Our caching algorithms are sensitive to this type of error, and we have presented the results in Table III. Table III shows the resulting cache hit ratios attained by the proposed UPP-based caching policies for different probabilities of selecting an unexpected VC for cache size of 200 GB. For example, the table shows when the probability of users requesting an unexpected VC is 0.05, the difference in cache hit ratio from the results presented in Fig. 8(a) is 0.02 for P-UPP and 0.025 for R-UPP.

Note that the first two types of error are consistent with a user that behaves as expected—i.e., requests videos according to his preference—and simply model the inaccuracy in determining the true user preference profile. The last of the three types of errors is modeling a user that behaves unexpectedly and requests videos that are not consistent with the user's historical preferences. This could, for example, be if the user has taken up a new interest recently, or maybe responding to an unusual event.

VI. CONCLUSION AND FUTURE WORK

In this paper, we demonstrated the feasibility and effectiveness of using micro-caches at the edge of the RAN, coupled with new caching policies based on video preference of users in the cell and a new scheduling technique that allocates RAN backhaul and wireless channel bandwidth in coordination with requesting video clients. Our simulation results show that the new RAN micro-caching-based video delivery approach can significantly increase the number of concurrent video requests that can be served while meeting initial delay requirements.

In this paper, we primarily addressed constant-bit-rate videos, and multiple quality or resolution versions of the same video used for progressive download, or adaptive bit rate (ABR) streaming, are treated as different videos. Consequently, depending on the cache policy (proactive or reactive) and the nature of requests, multiple quality/resolution versions of the same video may have to be cached at the same (e)NodeB, leading to inefficient use of the limited sized RAN caches. In the future, techniques need to be developed to address the above problem, in particular in the context of ABR streaming, which is gaining popularity. With ABR-capable streaming, multiple bit-rate versions of a video can be requested by a mobile client during a video session due to varying network conditions, making the problem of efficient video caching even more difficult. In [44], we have proposed a new architecture supplementing (e)nodeB caches with limited processing capability, and a joint caching and video processing algorithm that not only caches the videos with the bit rate that is most likely to be requested by the users within the cell, but also uses the (e)NodeB processor to transcode the videos to the desired bit rate. Preliminary results show the feasibility of improving the cache hit ratio significantly over the current approach that needs to cache all bit rate versions. In the future, we plan to further improve ABR caching and extend ABR-capable caching for proactive caching policies. Furthermore, we plan to extend our RAN caching approach to hierarchical caching at the gateways of the core network to support mobility of users across cells and increase network capacity.

APPENDIX

A. Time Complexity Analysis of the Caching Algorithms

Time complexity analysis of the caching algorithms is important as it measures the amount of time required to execute critical cache operations, for instance fetching the videos from the cache (cache lookup) and making caching decisions for proactive policies or replacement decisions for reactive policies (cache maintenance). However, not all operations are equally time-sensitive, and additionally, some operations can be done offline. Specifically, cache lookup is time-sensitive because it has a direct impact on the users' experience, and it is usually an online process as most often the user requests are generated on the fly and are not a known sequence of requests. Cache maintenance is less time-critical, but it has impact on future requests. For instance, for reactive caching policies, it is important to identify and replace the cached videos considered for replacement as soon as possible in order to have the video ready for impending requests. However, the time sensitivity of cache maintenance is lower than the cache lookup. In this section, we first explain the complexity of cache lookup, which is common across all the caching policies. Later, we describe the complexity of cache maintenance separately for each cache policy. For efficiency of implementation, we propose an optimal data structure for each of our caching algorithms.

Throughout this section, we assume that sorting of the videos based on their popularity (national video popularity) can be done offline every time the ranking of the videos changes, and hence the sorting of the videos does not directly impact the time complexity of the operations that rely on the videos being sorted. The optimal data structure for the cache lookup is a sorted linked list to keep track of the videos based on their adjusted popularity and a hash table of pointers to the sorted list to map the videos to their popularity ranking. Adjusted popularity refers to national video popularity for MPV, least recently used for LRU, and geographical popularity of videos according to the cell site AUS for UPP-based policies. For all the available videos that can be cached, the idea is to have a hash key that is long enough that it can map videos with high probability. We then create a mapping from the hash table to the cache entry pointer in the linked list. With such a data structure, the access time depends on the probability of hash collision [45] and is given by $O(N/k)$, where N is the number of videos in the cache and k is the number of unique hash keys in the table. Therefore, on average, if a video is in the cache, we can identify and retrieve the video from the cache in $O(N/k)$ or declare a cache miss and fetch the video from the Internet CDN. The worst-case time complexity happens when all the videos in the cache are mapped to one hash value, which is $O(N)$. Many variants of the hash table are available for implementation considerations [46].

MPV: MPV cache maintenance requires updating of the cache when the national video popularity changes—as mentioned before, we assume that national popularity distribution

of the videos is maintained offline—and subsequent populating of the cache with videos corresponding to the new ranking. In order to save on bandwidth usage instead of evicting the entire MPV cache and repopulating it with the newly ranked videos that have complexity of $O(N)$, we resort the cache and only replace those videos evicted from the cache. The average and worst-case time complexity of the sorting algorithm is that of a general sorting algorithm: $O(N\log(N))$ [45]. If major changes in the national popularity ranking of the videos do not occur often, the MPV cache maintenance is negligible.

LRU: In the LRU caching policy, when adding a video to the cache, the video is placed at the head of a doubly linked list. On eviction of a video from the cache, the video is removed from the tail of the list. The time complexity of the maintenance operation depends on whether the request results in a cache hit or miss. If there is a cache hit, the corresponding entry in the linked list must be moved to the head of the list, which can be done with $O(1)$ time complexity. In the event of a cache miss that triggers a successful video download (backhaul not blocked), the time complexity is the time it takes to add the requested video to the beginning of the list and to remove any evicted video(s) from the end of the list, which are both $O(1)$. Hence, even for a cache miss, the average time complexity of maintenance for LRU is of the order of $O(1)$. The average number of times the cache update needs to be executed per request depends on the probability of getting a cache miss and successfully scheduling the download from the backhaul or probability of cache hit. This probability is given by $(1-p_{\rm h})(1-p_{\rm b})+p_{\rm h}$, where $p_{\rm h}$ is the probability of cache hit and $p_{\rm b}$ is the probability of backhaul blocking; if the video request is blocked, no additional operation is required.

R-UPP/P-UPP: R-UPP and P-UPP caching policies require keeping track of the average cell site UPP (steps 5 and 2 of the R-UPP/P-UPP algorithms, respectively). We assume that information about the users' UPP is already available and can be retrieved from a central server with no additional time/memory complexity to the caching algorithms. For keeping track of the AUS UPP, we require $|U|\times|VC|\times W$ bits of memory to store all the active users' UPP, where $|U|$ is the cardinality of active user set, $|VC|$ is the total number of video categories, and W is the number of bits used to quantify the user preferences for each category. Furthermore, the users are being added/removed one at a time, so for each user added/removed, we adjust the average cell site UPP by a simple update algorithm

$$Add: UPP^{(t)} = (|U|\times UPP^{(t-1)} + UPP_i)/(|U|+1)$$
$$Remove: UPP^{(t)} = (|U|\times UPP^{(t-1)} - UPP_i)/(|U|-1).$$

In the above equations, $UPP^{(t)}$ represents the average UPP at time t, and UPP_i is the user preference profile of the ith user. Calculating the average UPP has time complexity $O(|VC|)$ because we need to average across all the video categories. The number of times this operation is required depends on the number of active users in the cell site and frequency of the AUS change.

Next, we need to study the probability of request for each video and calculate MLR and LLR sets explained in Sections III-C and III-D (steps 6 and 7 of the R-UPP and steps 3 and 4 of the P-UPP caching algorithms). As explained in the algorithms, each time the AUS or national video popularity changes, the ranking of the videos in the cache or potential videos for caching may change and require regenerating LLR and MLR sets for R-UPP and P-UPP, respectively, and consequently resorting of the cache. Alternatively, the resorting of the list for both R-UPP and P-UPP can happen at fixed intervals—e.g., every 1–10 min—to reduce computational overhead. The time complexity of calculating $P_{\rm R}$ for R-UPP and P-UPP caching policies is not the same, so we explain the time complexity for each policy separately.

To calculate $P_{\rm R}$ for R-UPP caching policy, in the event of AUS change, we need to go through the list of cached videos and recalculate the popularity of each cached video. Average time complexity of this operation is $O(N\times|VC|)$ because we need to go through the cache and use (3) to calculate $P_{\rm R}$—however, here we assume that each video belongs to only one video category so the complexity reduces to $O(N)$. After an AUS change or a change in national video popularity, the ranking of the videos in the cache may change, so the list is resorted accordingly in $O(N\log(N))$ average operations [45], or worst-case time complexity of $O(N\log(N))$ for instance using merge sort. Furthermore, in the event of a video request, if the video request is a cache miss, we first need to calculate $P_{\rm R}$ for the video request, and if it is higher than the request probability of the LLR, it will cause eviction with $O(1)$ time.

The complexity analysis for the P-UPP follows the R-UPP time complexity analysis, with the difference that the ranking of the videos is done across all the available videos instead of just the cached videos. However, as going through all the videos in the universe is not feasible, we envision performing the analysis on the subset of the videos, not currently cached, with highest popularity within each video category. We start by calculating $P_{\rm R}$ for all the videos in the cache and sort them according to the updated request probability. The time complexity of this operation is $O(N\log(N))+O(N)$. Next, we calculate $P_{\rm R}$ for the most popular video, not already in the cache, within each video category and select the one with the highest request probability to replace the video in the cache with the lowest request probability if the new video has higher request probability than the video from the cache that it replaces. This iteration continues until the most likely requested candidate video has lower request probability than the least likely requested video in the cache. For each step of the iteration, one new request probability must be calculated, and the video with the highest probability among the video categories must be selected as the cache replacement candidate. The worst-case time complexity of the above operation occurs when all cache entries are replaced with complexity $O(|VC|\times N)$, while the best case is $O(|VC|)$, which occurs when no replacement is required. In conclusion, the average maintenance complexity of the P-UPP caching policy is $O(N\log(N))$, dominated by the sorting of the cache.

In terms of actual execution time measured in our MATLAB implementation, for a cache size of 200 GB with approximately 3030 videos in the cache and average duration of 8 min, user interarrival time of 16 s, and total number of video categories of 247, it takes on average 8.5 ms to calculate $P_{\rm R}$ for R-UPP

TABLE IV
AVERAGE/WORST-CASE TIME COMPLEXITY OF THE CACHING ALGORITHMS

Caching Policy	Maintenance Average/Worst-case Analysis	Frequency of maintenance
MPV	$O(N)$ or $O(N\log(N))$	When national video popularity changes
LRU	$O(1)$	Once per video request
R-UPP	$O(N\log(N))$	When national video popularity or AUS changes
P-UPP	$O(N\log(N))$	

Note: Cache Lookup has $O(1)$ time complexity for all cache policies

caching policies (all videos in the cache) and 98.3 ms to calculate P_{R} for P-UPP caching policies (optimized search of all available videos as explained in this Appendix). The former is done for each new video request that causes a cache miss, while the latter is done only when AUS changes. The above example shows that the overhead of calculating P_{R} will not be limiting to serve requested videos in real time.

Table IV summarizes the time complexity of the caching policies that we discussed in this section.

B. Leaky Bucket Parameters Generation

Although the objective of HRD incorporated as part of most codecs, e.g., x264, is not to generate LBP parameters, but to create an encoded stream that complies with the decoder buffer initial delay and max buffer size given a transmission rate, the GHRD, which is part of the JM codec published by Joint Video Team (JVT), creates LBPs. For instance, implementation of the LBPs is in file "leaky_bucket.c" for JM codec [32]. Here, we present the approach we used to generate LBPs for each encoded video. Encoded video frames have variable bit rates, and the decoder needs to decode video frames, each of different sizes in order to display the video sequence on the screen. Thus, we can model the UE's video buffer as a queue, where the input has a constant bit rate, representing the minimum rate by which the channel delivers the encoded video bits and the output has a variable rate representing the rate at which the decoder removes variable size frames from the queue for decoding. As such, for generating LBP the objective is to find the initial fullness level F for a D/G/1 queue, with input rate R and output rate R_{out}, so that the queue never becomes empty. We define the equivalent problem: Find the largest queue deficit of a G/D/1 queue with input rate R and output rate R_{out} when starting the queue empty and allowing negative buffer level, representing a temporary deficit in the download rate compared to the video decoder rate. At time T (in frames), the total queue size N_{Q} can be described by the following recursion:

$$N_{\mathrm{Q}}(T) = T\frac{R}{f} - \sum_{n=1}^{T} b_n = N_{\mathrm{Q}}(T-1) + \frac{R}{f} - b_T \quad (8)$$

where b_n is the frame size in bits of frame n, and f is the frame rate in frames per second. The initial buffer fullness to guarantee that the buffer never runs empty assuming a video of length L is then

$$F = -\min_{T=1,\ldots,L}(N_{\mathrm{Q}}(T)). \quad (9)$$

As an example, we encoded CIF format Paris video sequence with 1065 frames [47] and generated the LBP for different transmission rates

$R_{\min}(\mathrm{kb/s})$	$B_{\min}(\mathrm{Bits})$	$F_{\min}(\mathrm{Bits})$	$D_{\min}(\mathrm{s})$
100	13982975	13982975	139.8
500	878232	878232	1.7
600	296752	296752	0.49
700	146896	137552	0.19
1000	146896	137552	0.14
5000	146896	137552	0.03

From the above data, it is apparent that, as the transmission rate increases, the initial delay and buffer size requirements decrease, until rates about 700 kb/s, where the buffer size stabilizes, as it needs to at least buffer the largest and initial I-frame.

REFERENCES

[1] Cisco, San Jose, CA, USA, "Cisco visual networking index: Global mobile data," White Paper, 2010–2015.

[2] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Commun. ACM*, vol. 49, no. 1, pp. 101–106, Jan. 2006.

[3] M. Pathan and R. Buyy, *A Taxonomy of CDNs, Content Delivery Networks*. Berlin, Germany: Springer-Verlag, 2008.

[4] Juniper Research, Basingstoke, U.K., "Press release: Mobile network upgrades of up to $840bn required over next five years to meet burgeoning data demand, finds Juniper Research," [Online]. Available: http://www.juniperresearch.com/viewpressrelease.php?id=336&pr=259

[5] S. Rudd, "Closing the backhaul gap," Strategy Analytics, 2013 [Online]. Available: https://www.strategyanalytics.com/default.aspx?mod=reportabstract viewer&a0=8236

[6] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," in *Proc. ACM Conf. Internet Meas. Conf.*, 2012, pp. 211–224.

[7] A. Wierzbicki, "Internet cache location and design of content delivery networks," *Web Eng. Peer-to-Peer Comput.*, vol. 2376, Lect. Notes Comput. Sci., pp. 69–82, 2002.

[8] A. Balamash and M. Krunz, "An overview of Web caching replacement algorithms," *Web Eng. Peer-to-Peer Comput.*, vol. 2376, Lect. Notes Comput. Sci., pp. 44–56, 2002.

[9] M. Cherniack, E. F. Galvez, M. J. Franklin, and S. Zdonik, "Profile-driven cache management," in *Proc. 19th ICDE*, 2003, pp. 645–656.

[10] Krishnappa, S. Khemmarat, L. Gao, and M. Zink, "On the feasibility of prefetching and caching for online TV services: A measurement study on Hulu," in *Proc. 12th PAM*, Mar. 2011, pp. 72–80.

[11] D. Marinca, A. Hamieh, D. Barth, and K. Khawam, "Cache management using temporal pattern based solicitation for content delivery," in *Proc. 19th ICT*, 2012, pp. 1–6.

[12] G. Nair and P. Jayarekha, "A rank based replacement policy for multimedia server cache using Zipf-like law," *J. Comput.*, vol. 2, no. 3, Mar. 2010.

[13] J. Z. Wang, Z. Du, and P. K. Srimani, "Network cache model for wireless proxy caching," in *Proc. 13th IEEE MASCOTS*, Sep. 2005, pp. 311–314.

[14] W. H. O. Lau, M. Kumar, and S. Venkatesh, "Cooperative cache architecture in support of caching multimedia objects in MANETs," in *Proc. WoWMoM*, 2002, pp. 56–63.

[15] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, and S. Sen, "To cache or not to cache: The 3G case," in *Proc. IEEE Internet Comput.*, 2011, pp. 27–34.

[16] N. Wakamiya, M. Murata, and H. Miyahara, "Video streaming systems with cooperative caching mechanisms," in *Proc. SPIE Int. Symp.*, 2002, pp. 305–314.

[17] N. Golrezaei, K. Shanmugam, A. G. Dimakis, and A. F. Molisch, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, 2012, pp. 1107–1115.

[18] M. Cha, H. Kwak, and P. Rodriguez, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Network.*, vol. 17, no. 5, pp. 1357–1370, Oct. 2009.
[19] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global cache local: YouTube network traces at a campus network—Measurements and implications," in *Proc. ACM Multimedia Comput. Netw.*, Santa Clara, CA, USA, 2008.
[20] Reelseo, "Research—Most popular video sites and categories," [Online]. Available: http://www.reelseo.com/most-popular-video-sites-categories/
[21] D. Lee, J. Choi, J.-H. Kim, and S. H. Noh, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.
[22] N. Laoutaris, "A closed-form method for LRU replacement under generalized power-law demand," in *Proc. MMCN*, San Jose, CA, USA, Jan. 2008.
[23] H. Ahlehagh and S. Dey, "Video caching in radio access network: Impact on delay and capacity," in *Proc. IEEE WCNC*, 2012, pp. 2276–2281.
[24] S. Schwarz, C. Mehlfuhrer, and M. Rupp, "Low complexity approximate maximum throughput scheduling for LTE," in *Proc. 44th Annu. Asilomar Conf. Signals, Syst., Comput.*, 2010, pp. 1563–1569.
[25] R. Kwan and C. Leung, "A survey of scheduling and interference mitigation in LTE," *J. Elect. Comput. Eng.*, vol. 2010, Jan. 2010, Art. no. 1.
[26] S. Sarabjot, S. Singh, O. Oyman, A. Papathanassiou, D. Chatterjee, and J. G. Andrews, "Video capacity and QoE enhancements over LTE," in *Proc. Int. Conf. Commun. Video Wireless Workshop*, Jun. 2012, pp. 7071–7076.
[27] D. Wang, V. S. Somayazulu, and J. R. Foerster, "Efficient cross-layer resource allocation for H.264/SVC video transmission over downlink of an LTE system," in *Proc. 1st IEEE WoWMoM Workshop Video Everywhere*, 2012, pp. 1–6.
[28] R. A. Berry and E. M. Yeh, "Cross-layer wireless resource allocation," *IEEE Signal Process. Mag.*, vol. 21, no. 5, pp. 59–68, Sep. 2004.
[29] M. Shehada, S. Thakolsri, Z. Despotovic, and W. Kellerer, "QoE-based cross-layer optimization for video delivery in long term evolution mobile networks," in *Proc. WPMC*, 2011, pp. 1–5.
[30] A. DuaC. W. Chan, N. Bambos, and J. Apostolopoulos, "Channel, deadline, and distortion (CD^2) aware scheduling for video streams over wireless," *IEEE Trans. Wireless Commun.*, vol. 9, no. 3, pp. 1001–1011, Mar. 2010.
[31] J. Ribas-Corbera, P. A. Chou, and S. L. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst.*, vol. 13, no. 7, pp. 674–687, Jul. 2003.
[32] K. Sühring, "JM codec," [Online]. Available: http://iphome.hhi.de/suehring/tml/
[33] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, pp. 237–252, 1998.
[34] 3GPP, "TR36.814 v1.0.0: Evolved universal terrestrial radio access (E-UTRA); Further advancements for E-UTRA physical layer aspects," 2009 [Online]. Available: http://www.3gpp.org
[35] 3GPP, "TS36.101 v11.3.0, Evolved universal terrestrial radio access (E-UTRA), User equipment (UE) radio transmission and reception," 2013 [Online]. Available: http://www.3gpp.org
[36] A. Lozano, A. M. Tulino, and S. Verdu, "Optimum power allocation for multiuser OFDM with arbitrary signal constellations," *IEEE Trans. Commun.*, vol. 56, no. 5, pp. 828–837, May 2008.
[37] D. P. Palomar and J. R. Fonollosa, "Practical algorithms for a family of waterfilling solutions," *IEEE Trans. Signal Process.*, vol. 53, no. 2, pp. 686–695, Feb. 2005.
[38] P. Ameigeiras, Y. Wang, J. Navarro-Ortiz, P. E. Mogensen, and J. M. Lopez-Soler, "Traffic models impact on OFDMA scheduling design," *EURASIP J. Wireless Commun. Netw.*, vol. 2012, p. 61, 2012.
[39] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York, NY, USA: Wiley, Apr. 1991.
[40] D. M. B. Masi, M. J. Fischer, and D. A. Garbin, "Video frame size distribution analysis," *Telecommun. Rev.*, vol. 19, pp. 74–86, Sep. 2008.
[41] X. Cheng, C. Dale, and J. Liu, "Statistics and social networking of YouTube videos," in *Proc. IEEE ACM/IEEE Int. Symp. Quality Service*, Jun. 2–4, 2008, pp. 229–238.
[42] R. Gallager and Bertsekar, *Data Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, 1992.
[43] F. Qiu and J. Cho, "Automatic identification of user interest for personalized search," in *Proc. 15th Int. World Wide Web Conf.*, 2006, pp. 727–736.
[44] H. Ahlehagh and S. Dey, "Adaptive bit rate capable video caching and scheduling," in *Proc. IEEE WCNC*, Apr. 2013, pp. 1357–1362.
[45] R. Neapolitan and K. Naimipour, *Foundations of Algorithms*, 3rd ed. Sudbury, MA, USA: Jones & Bartlett.
[46] J. Morris, "Data structures and algorithms," 1998 [Online]. Available: http://www.cs.auckland.ac.nz/~jmor159/PLDS210/hash_tables.html
[47] Arizona State University, Phoenix, AZ, USA, "YUV video sequences," [Online]. Available: http://trace.eas.asu.edu/yuv/

Hasti Ahlehagh received the Master of Science degree in electrical engineering from Worcester Polytechnic Institute, Worcester, MA, USA, in 2004, and is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of California, San Diego, La Jolla, CA, USA.

Before pursuing the Ph.D. degree, she worked as a Senior Staff Software Engineer with the Connected Home Division of Motorola Mobility, Inc., and earlier as a Firmware Software Engineer writing software for 3G wireless networks.

Sujit Dey 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, 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 also serves as the Faculty Director of the von Liebig Entrepreneurism Center. He is affiliated with the California Institute of Telecommunications and Information Technology (Calit2) and the UCSD Center for Wireless Communications. He served as the Chief Scientist, Mobile Networks, with Allot Communications from 2012 to 2013. He founded Ortiva Wireless in 2004, where he served as its founding CEO and later as CTO and Chief Technologist until its acquisition by Allot Communications in 2012. Prior to Ortiva, he served as the Chair of the Advisory Board of Zyray Wireless until its acquisition by Broadcom in 2004, and as an advisor to multiple companies including ST Microelectronics and NEC. 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 close to 200 publications, including journal and conference papers, and a book on low-power design. He is the co-inventor of 17 US and two international patents, resulting in multiple technology licensing and commercialization.

Prof. Dey has been the recipient of six IEEE/ACM Best Paper awards and has chaired multiple IEEE conferences and workshops.