Mobile Device Video Caching to Improve Video QoE and Cellular Network Capacity

Hasti A. Pedersen and Sujit Dey
Dept. of Electrical and Computer Engineering,
University of California San Diego, La Jolla, CA
hasti@ucsd.edu, sdey@ucsd.edu

ABSTRACT

As the video resolution, storage, and rendering capabilities of mobile devices improve, consumers tend to watch more videos on their devices. To address this increasing demand, there is a need to improve the capacity of cellular networks while also improving video Quality of Experience (QoE). Ensuring video QoE via wireless links remains a challenge, not only due to the backhaul demand, but also due to the varying wireless channel conditions caused by fading, multiuser interference, and peak traffic loads. Here, we introduce a reactive Mobile Device Caching (rMDC) framework to improve video capacity (number of concurrent video viewing sessions) and QoE in the presence of the challenges explained above. Using rMDC, a mobile device caches videos reactively as it requests them, evicts the videos least likely to be requested according to its neighbors aggregate User Preference Profile (UPP), and sharing video contents using D2D communication. We use a discrete event statistical simulation framework to study the performance of rMDC. Our simulation results demonstrate the effectiveness of rMDC, along with UPP-based caching, in achieving higher capacity and better QoE compared to no mobile device caching.

Keywords

Mobile Device Caching; D2D Communication; Wireless Network Capacity; Video Quality of Experience.

1. INTRODUCTION

According to the Cisco Visual Networking Index (VNI) forecast report, mobile video traffic will grow at a compound annual growth rate of 69% between 2013 and 2018 [4]. Even when offloading video traffic through alternative methods, this forecasted growth in video traffic will put severe strain on mobile networks.

When Internet video is accessed by a mobile device, it must be fetched from the servers of a Content Delivery Network (CDN) and traverse through the mobile carrier Core Network (CN), Radio Access Network (RAN), and wireless channel to reach the mobile device. In [7] [6], the authors proposed video caching in the RAN along with User Preference Profile (UPP)-based caching policies, and showed that caching along with video aware backhaul scheduling can increase the capacity of the RAN backhaul while improving users' QoE. Subsequently, [6] and [8] showed that using Adaptive Bit Rate (ABR) aware RAN caching together with backhaul and wireless channel scheduling further improves end-to-end video capacity of the cellular networks beyond what can be achieved by either ABR or RAN caching individually, while preserving the advantages in terms of QoE obtained by each of them.

In this paper, we extend RAN caching to utilize the storage available on mobile devices as well as D2D links (like Wi-Fi Direct) to off-load the traffic from cellular links and thus further enhance end-to-end cellular video capacity and experienced QoE. Our proposed solution reactively caches videos on mobile devices as they are being requested, and opportunistically shares cache contents using D2D communication. Therefore there is no bandwidth penalty associated with preloading the caches, and our framework performs well even for video consumption in cellular networks where relatively few mobile devices are within D2D range.

Our proposed solution, reactive Mobile Device Caching (rMDC), efficiently utilizes the storage and multiple radio resources available at the mobile devices to reactively cache video contents and share them with neighbor devices over unlicensed bands. For this research, we mostly focus on legacy Wi-Fi and Wi-Fi Direct protocols. Thus, the proposed framework is cellular resource aware as the cellular resources that are consumed to cache a video could have been used for faster downloading of an already ongoing video request. The idea behind the rMDC caching framework is to address the inherent challenges and trade-offs associated with mobile caching to improve the video QoE for mobile devices, as well as capacity of the cellular network.

In summary, the proposed rMDC framework consists of the following two parts: (a) a reactive policy where mobile devices reactively cache video contents and share their cache contents with their neighbor devices through D2D links to off-load traffic from cellular networks; and (b) capability of downloading different parts of the video from multiple sources – e.g. portions of a video from different neighbor devices and the (e)NodeB to speed up downloads; we termed this capability Video Retrieval Process.

Fig. 1 illustrates possible download options for a mobile device in the rMDC framework. In this example a mobile device, UE_1, downloads parts of the video from its own cache,

MSWiM'14, September 21–26, 2014, Montreal, QC, Canada.
Copyright 2014 ACM 978-1-4503-3030-5/14/09 ...$15.00.
http://dx.doi.org/10.1145/2641798.2641820.

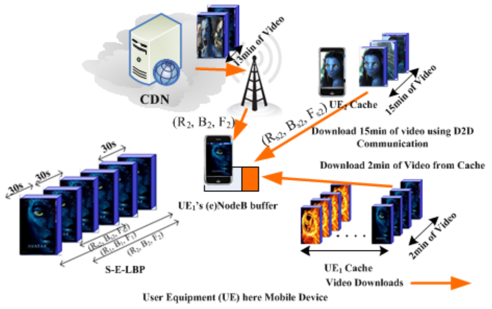

Figure 1: rMDC framework: Reactive Caching, Shared Caches and Possible Download Paths, and S-E-LBP.

acquiring parts of the video from the neighbor mobile device caches and the remaining parts (chunks) of the video from the (e)NodeB. Thus, depending on where the video source is available, the mobile device can obtain its contents from multiple sources.

The concept of data sharing has been widely implemented and adopted through Peer-2-Peer (P2P) protocols such as BitTorrent [5], Gnutella, PeerCast, PeerStreamer [1], and their client implementations. The historic motivation behind P2P protocols were to speed up downloads by working around data rate disparity between uplink and downlink due to the uneven provisioning in the wired networks. Using P2P, instead of one user transferring an entire file to the requesting user, multiple collaborating users that have the contents in their caches, transfer different parts (chunks) of the files to the requesting user. Thus, using multiple uplinks simultaneously improves the overall achieved rate in the P2P networks. The essence of retrieving video contents in rMDC framework is that of P2P protocols, in which a requesting video user receives different video chunks from different sources. However, more stringent scheduling of the users compared to P2P protocols is required due to: (1) cooperation range of devices with rMDC (Wi-Fi range) is smaller than that of P2P (any two users with connections to the Internet can share contents), and (2) P2P protocols do not aim to improve the capacity of the operators' network, and might negatively impact the cable providers' network throughput. On the contrary, rMDC aims to improve overall capacity and QoE.

Like in [7], we propose to use Leaky Bucket Parameters (LBP) associated with a requested video to select a transmission rate that, if met by the network, can ensure playback without stalling and guarantee a maximum initial delay. Here, we extend the LBPs to consider the additional flexibility of partial downloads from different sources of videos: the mobile device's own cache, cache of cooperative mobile devices, and (e)NodeB. We call the proposed LBP, Segmented E-LBP (S-E-LBP). As shown in Fig. 1, unlike LBP that is three tuples generated for the entire video, S-E-LBP is for parts of the videos, for instance, 30s into the video up to 120s. Using LBP instead of S-E-LBP is always an option but the parameters are more conservative.

In summary, the novelty and importance of the contributions of this paper are: (a) it is the first to propose a caching framework along with a video-retrieval approach to opportunistically use the storage and radio capacities of the mobile devices to improve capacity and QoE regardless of the application; and (b) it is the first paper to use S-E-LBP table to ensure initial delay and improve probability of stalling by coordinating between downloads from different sources.

We have developed a simulation framework to demonstrate the effectiveness of the rMDC framework. The simulation results demonstrate significant increase in possible video capacity and video QoE using our approach, as opposed to no UE caching.

1.1 Related Work

The field of mobile caching and D2D communication is rich in contributions and is an active area of research both in academia [3] and industry [12]. The majority of the previous research on D2D communication focused on addressing the challenges of using cellular spectrum for both D2D communication and cellular links by methods of interference reduction or avoidance among cellular and D2D users [3] or methods of estimating Channel State Information (CSI) of D2D links for more efficient scheduling. However, today's mobile devices are equipped with multiple radio interfaces, (e.g., Wi-Fi, Bluetooth, NFC), and using this free spectrum as we propose in this paper can further improve the performance of cellular networks. The efforts in mobile device caching are mostly focused on improving application experience rather than the cellular networks. We believe none of the previous works address the problem that we address here: *Use mobile device cache and radio resources available to opportunistically assist (e)NodeB to improve cellular networks' video capacity and QoE during peak traffic load.*

Browser caching for webpages and their components has been done with size restrictions in mobile device operating systems such as Android and iOS. Recently, YouTube rolled out a feature that preloads videos from the "subscription" or "watch later" list into cache memory [2]. With this feature, users can play back videos from their cache later – as long as they maintain connection to the YouTube servers – and thus experience better video quality. Although this feature improves users' QoE, its effectiveness relies on users updating their "subscription" and "watch later" list. Furthermore, the download process is ignorant of the backhaul and wireless channel bandwidth availability.

Recently, [10] and [9] proposed a promising caching solution to improve the video capacity of cellular networks. However, the former approach uses Most Popular Videos ranking to distribute contents among mobile devices, which may not lead to optimal results as explained and shown in [7]. The latter requires presence of additional helper nodes (e.g., Femto cells) where videos are cached, which may be hard to satisfy throughout a network.

The remainder of this paper is organized as follows: In section 2, we first explain the objective of our work and then provide an overview of our solution: the rMDC framework. Subsequently, in section 3, we detail our proposed mobile device UPP-based proactive caching policies. In section 4, we introduce the video retrieval process and buffer management. Section 5 outlines our simulation framework, and provides experimental results. We conclude the paper in section 6.

2. OVERALL APPROACH

Fig. 2 shows an overview of the rMDC framework, including network and peer establishment as well as caching, and video retrieval steps. Once a mobile device becomes active in a cell, the neighbor establishment process commences.

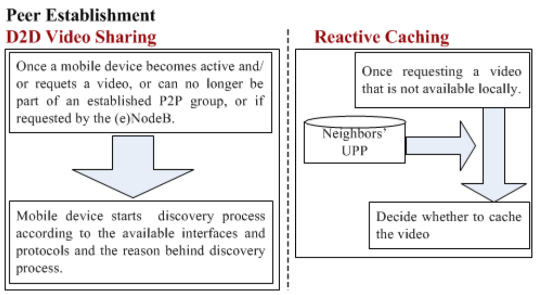
Peer Establishment
D2D Video Sharing
Once a mobile device becomes active and/ or requets a video, or can no longer be part of an established P2P group, or if requested by the (e)NodeB.
Mobile device starts discovery process according to the available interfaces and protocols and the reason behind discovery process.
Reactive Caching
Once requesting a video that is not available locally.
Neighbors' UPP
Decide whether to cache the video

Figure 2: rMDC Network and Peer Establishment

In rMDC approach, a mobile device sends a request to the R-UPP cache agent at the (e)NodeB with a listing of its cache contents and location. Meanwhile, a mobile device may self-initiate a peer establishment process, or initiate the process upon a request from the cache agent at the (e)NodeB. A mobile device self-initiates the peer establishment process if (a) it does not belong to a D2D group and it becomes active in a cell or requests a video that is not available in its cache, or (b) it can no longer stay connected with an already established group. During the peer establishment process, a mobile device starts a discovery process to identify the cooperating mobile devices in its vicinity. For instance, following Wi-Fi direct protocol [11], a mobile device starts looking in social channels, envisioned in Wi-Fi direct protocol, for a group or it initiates a group itself. The cache agent running on the (e)NodeB tracks the mobile devices' peer associations and potential peers for future video sharing. Occasionally, (e)NodeB sends a group or peer change request to a mobile device if the alternative peer association facilitates downloads such as when, for instance, the requested video is available in a mobile device within the vicinity of the requesting mobile device, but not the group that the mobile device is currently associated with.

In the event of a new video request, a mobile device first searches its own cache and then neighbor caches, or caches within its Wi-Fi direct group in an attempt to retrieve the contents locally. If parts of the video are not available locally, the R-UPP cache agent at the (e)NodeB first tries to identify another device in the requesting mobile device's vicinity (but not within the same group) that has the content and initiate a group change request. Ultimately, if it does not find the missing content in any neighboring device, then the R-UPP cache agent schedules the remaining chunks either from the eNodeB cache, if it is a cache hit, or downloads them from the CDN, and schedules the video through the cellular link. Within the rMDC framework, when the mobile device downloads the requested video, it caches the video reactively according to the R-UPP mobile device caching policy described in section 3. Thus, after watching one or more videos, without adding any additional burden on the cellular link for downloading the requested video, a mobile device is capable of sharing those videos using D2D communication with other mobile devices within its Wi-Fi communication range.

In the rMDC framework, the buffer manager running at the mobile device is responsible for retrieving video chunks in time for the playback from different neighbor caches and/or the cellular links. We provide details of retrieval process and content coordination in section 4. Next, we describe our R-UPP caching policy that assists downloads of neighbor devices using D2D links.

```
R-UPP Mobile Device Caching Algorithm
R-UPP Cache agent running at the (e)NodeB
1.  For each active mobile device in a cell site
2.   Calculate the aggregate UPP of the device itself and all the
     neighbor devices
3.  End

R-UPP Mobile Device Caching Policy running at the
(e)NodeB
1.  For new Video Request $V_j$
2.   If chunks $V_j \in U_i$ cache
3.    transfer video chunks from cache to playback buffer, $B_i$
4.   End
5.   If any remaining chunks $c_j$ of $V_j \notin B_i$ and $c_j$ of $V_j \in$ Neighbor
Device
6.    Download the chunks $c_j$ from the neighbor device
7.   End
8.   If any chunks of $V_j \notin B_i$
9.    Schedule missing chunks of $V_j$ through cellular links
10.   Calculate $P_R$ for $V$ and the cached videos and generate LLR
subset
11.   $LLR_j$: subset of LLR videos with least $P_R$ that has to be
evicted
      from cache to fit $V$
12.   If $P_R(V_j) - \sum P_R(LLR_j) > 0$
13.    Cache = Cache + $V_j$ - $\{LLR_j\}$
14.   Else
15.    Do not cache $V_j$
16.   End For, End If, End If
```

Figure 3: P-UPP Mobile Device Caching Algorithm.

3. R-UPP MOBILE DEVICE CACHING

We first proposed the R-UPP caching algorithm in [7] to improve the cache hit ratio of relatively small sized caches at the (e)NodeBs of the cellular networks without the overhead of preloading the caches. The R-UPP caching policy reactively caches videos as they are being requested and evicts videos according to the average cell site UPP.

In this paper, we propose R-UPP mobile device caching policy to facilitate video content sharing locally through D2D links among neighboring devices, without the associated overhead of consuming cellular resources to cache videos just for the purpose of D2D communication. The approach of reactive caching is especially suitable within the setting presented in this paper, as existence of D2D links within the cellular network is opportunistic and therefore cannot be taken for granted.

Fig. 3 depicts the R-UPP mobile device caching algorithm. Whenever there's a change in a mobile device's neighbor list, the R-UPP cache agent running at the (e)NodeB calculates the aggregate neighbor UPP and communicates it to the mobile device (lines 1-3). R-UPP mobile device caching policy calculates the request probability of the videos in its cache using the following equation:

$$P_R(v_{Uj}) = \sum_{c=1}^{|VC|} p_{vc}^{(U)}(c) p_j(c) \qquad (1)$$

where $|VC|$ is the total number of video categories, $p_{vc}^{(U)}(c)$ is the probability that a device in neighbor group U requests a video belonging to category c, and $p_j(c)$ is the request probability for video j given category c. Like in [7], here, we assume, the category of each video is known.

Upon a video request, if the request is a cache hit in the mobile device's cache, U_i, the buffer manager transfers video chunks to the playback buffer (line 3). For any chunks of the video not found in the cache, the mobile device requests it locally using D2D communication (lines 5-6). If the requested chunk is not available locally (line 8), the mobile device fetches it from the (e)NodeB. If the UE cache is full, R-UPP caching policy calculates the request probability of the videos in its cache as well as the request probability of the requested video using eq. 1, forming a Least Likely Requested (LLR) subset (lines 10-11). R-UPP mobile device caching policy calculates the difference between request probability of the newly requested video, and that of the subset of LLR videos from the cache with the 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, does R-UPP caching policy effectuate the cache update (lines 12-13). The above approach ensures that the cached videos maintain the highest probability of requests by the current neighbor devices.

4. VIDEO RETRIEVAL PROCESS

The rMDC framework aims to improve video capacity of cellular networks, as well as user's video QoE by reducing the stalling probability and initial delay. A mobile device with rMDC framework may potentially receive video chunks from many different sources: receiving parts of a video from its own cache, parts from neighbor devices, and parts from the cell site. Although these multiple download paths result in cellular capacity savings, as well as improved QoE, retrieving video chunks on time to ensure stall-free playback is important. Thus, as in [7], rMDC framework uses LBP to facilitate playback with bounded initial delay and without stalling.

LBPs consist of N 3-tupples (R, B, F) corresponding to N sets of transmission rates and resulting buffer size parameters for a given bit stream [7]. An LBP tuple guarantees that as long as the average 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 can proceed without any buffer underflow (stalling) or buffer overflow. As part of the rMDC framework, we introduce Segmented E-LBP (S-E-LBP) table. With S-E-LBP, a video is divided into segments for which individual LBP parameters are given. Fig. 1 shows example S-E-LBP for a video sequence, where (R1, B1, F1) are the LBP corresponding to the entire sequence of video, (R2, B2, F2) are the LBP parameters applicable to downloading from 30s into the video and to the end of the video, and (Rs2, Bs2, Fs2) are the LBP for the segments of the video that starts 30s within the video up to 240s in the video. Fig. 1 shows an example that demonstrates the S-E-LBP usage. In the example, UE_1 finds chunks corresponding to the first 2 minutes of the movie Avatar in its own cache, retrieves these chunks without delay, and subsequently starts the playback. Meanwhile, the mobile device achieves a cache hit in the neighbor mobile device (UE_2) and downloads additional chunks (corresponding to 15 minutes of video playback) from that device. When requesting the video from the neighboring device, UE_1 can select a rate that is associated with the initial delay of 2 minutes (because the mobile device already has in its cache the chunks corresponding to the first 2 minutes of that video). Thus, the network can potentially associate a lower transmission rate to this mobile device if sufficient capacity is not available. Finally, the remainder of the video is fetched through the cellular link with the initial delay of up to 2+15=17 minutes maximum; thus requiring a much lower transmission rate through the cellular link and helping increase its capacity.

```
Video buffer management and video playback algorithm
1. If $V_j \in U_i$ cache
2.    Transfer video bits to playback buffer
3.    Identify the missing chunks of the video from the beginning
4.    If enough bits in video buffer
5.       Starts the playback right away
6.    End If
7.    Request remaining chunks from neighbor devices
8.    If any chunk is not found locally
9.       Request download from (e)NodeB according to S-E-LBP
         table to ensure stall free playback
10. End If, End If
```

Figure 4: Buffer Management and Playback Algorithm.

Fig. 4 shows the buffer management and playback algorithm. If any part of the video is available in the mobile device's cache, the buffer manager transfers the video chunks to the video playback buffer. Subsequently, it identifies the missing chunks from the beginning of the video and schedules download from the neighbor devices or (e)NodeB with transmission rates according to S-E-LBP.

5. SIMULATION RESULTS

In this section, we evaluate the impact of our proposed rMDC framework on the capacity and user video QoE using the MATLAB statistical simulation framework we developed earlier [6][8]. Like in [8], we assume a database of 20,000 videos following a Zipf popularity distribution with exponent value of -0.8. The video duration is exponentially distributed with mean of 8 minutes and truncated to a maximum of 30 minutes and a minimum of 2 minutes. We assume the video codec bit rate is uniformly distributed between 200kbps (QVGA quality) and 2Mbps (HD quality). The simulation assumes a pool of 5000 potential users and uses a Poisson arrival and departure model with average user active time of 45 minutes and inter-arrival time of 12 seconds. Video requests are generated independently per active user and follow a Poisson process with a mean inter-arrival time between requests. In terms of video QoE, we assume maximum acceptable initial delay of 10s for each user – i.e. playback should start no later than 10s after a video is requested. In terms of resources, we assume 1000Mbps for backhaul bandwidth (backhaul is not bottleneck) and the baseline wireless channel is modeled with parameters listed in table II of [8]. The results presented in this paper cover 5,000 video requests per simulation.

Next, we study the effectiveness of rMDC framework in improving capacity and video QoE. We also demonstrate the impact of cache size and D2D communication range on video capacity and QoE. Fig. 5(a) shows capacity for no mobile device caching with and without ABR, rMDC framework with cache size (C) ranging from 500Gbit to 5000Gbit and with D2D communication ranges (R) of 100m and 500m. 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 inter-arrival time such that the steady-state target blocking rate is achieved and noting the number of concurrent video requests sustained at that specific user inter-arrival time. Fig. 5(a) shows that significant improvement can be achieved in terms of capacity using rMDC framework. Furthermore, the figure shows that as the D2D range increases, the capacity increases. For instance, for cache size of 5000Gbit, increasing the D2D communication range from 100m to 500m results in 24% improvement in

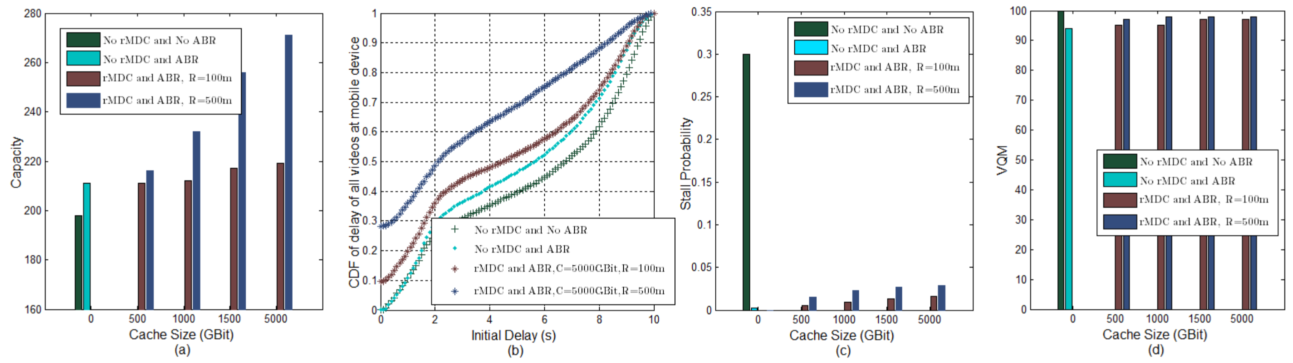

Figure 5: Performance of rMDC framework: (a) End-to-end capacity for rMDC with different D2D range and cache size; (b) CDF of initial delay of all videos at mobile device with rMDC mobile device caching framework; (c) Stalling probability for rMDC with different D2D range and cache size; (d) Average VQM for rMDC with different D2D range and cache size configuration.

terms of capacity. We also observe that as the D2D range increases the impact of cache size on capacity becomes more significant. For instance, for D2D range of 500m, increasing cache size from 500Gbit to 5000Gbit results in 25% capacity improvements while for D2D range of 100m, it results in 4% capacity improvement. Overall allowing for D2D with communication range of 500m and reactive cache size of 5000GBit results in 28% and 37% improvement in terms of capacity compared to not using rMDC framework with and without ABR respectively.

Next, we study the impact of rMDC framework on video QoE. Fig. 5(b) shows the CDF of initial delay when rMDC framework with cache size of 5000Gbit and D2D communication range of 100m and 500m are used. Fig. 5(b) shows that caching at the mobile device improves initial delay. For instance, using rMDC framework with cache size of 5000Gbit and D2D range of 500 improves the probability of achieving an initial delay of less than 1s by 26% compared to not using rMDC framework. Furthermore, Fig. 5(b) illustrates that increase in D2D communication range from 100m to 500m for cache size of 5000Gbit, results in 8% improvements in terms of experienced initial delay.

Fig. 5(c) and Fig. 5(d) show the probability of stalling and achieved average VQM for rMDC framework. Fig. 5(c) demonstrates that using ABR results in 136-fold reduction in the number of video requests that experience stalling. From the figures, we can infer that there can be significant improvement in terms of capacity and initial delay using our proposed rMDC mobile device caching approach, with only marginal degradation in stalling probability and almost similar VQM, compared to just using ABR without mobile device caching.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a novel video mobile device caching framework that uses storage and radio resources available at the mobile device along with our proposed user preference profile based reactive caching policy to improve capacity and QoE of the cellular networks.

Using simulation results, we showed that using the proposed reactive caching policy and the video retrieval policy can result in capacity and video QoE improvements.

7. REFERENCES

[1] http://en.wikipedia.org/wiki/p2ptv.

[2] www.youtube.com.

[3] A. Asadi and V. Mancuso. A survey on device-to-device communication in cellular networks. *submitted to IEEE Communications Surveys and Tutorials.*

[4] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2013-2018. *Cisco White Paper.*

[5] B. Cohen. Incentives build robustness in bittorrent. *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[6] H.Ahlehagh and S.Dey. Video aware scheduling and caching in the radio access network. *To appear in IEEE Transactions on Networking, VOL. 22, NO. 5. (IEEE Xplore).*

[7] H.Ahlehagh and S.Dey. Video caching in radio access network: Impact on delay and capacity. *Proceedings of the 2012 IEEE Wireless Communications and Networking Conference (WCNC 2012)*, 2012.

[8] H.Ahlehagh and S.Dey. Adaptive bit rate capable video caching and scheduling. *Proc. IEEE Wireless Communication and Networking Conference*, 2013.

[9] N. Golrezaei et al. Femtocaching: Wireless video content delivery through distributed caching helpers. *International Conference on Computer Communications (INFOCOM)*, 2012.

[10] N. Golrezaei et al. Wireless device-to-device communications with distributed caching. *IEEE International Symposium on Information Theory (ISIT)*, 2012.

[11] Wi-Fi Alliance Technical Committee P2P Task Group. Wi-fi peer-to-peer (p2p) technical specification. 2010.

[12] X. Wu et al. Flashlinq: A synchronous distributed scheduler for peer-to-peer ad hoc networks. *IEEE Allerton Conference on Communication, Control, and Computing, pp. 514-521*, 2010.