Review Article

Caching Eliminates the Wireless Bottleneck in Video Aware Wireless Networks

Andreas F. Molisch,[1] Giuseppe Caire,[1] David Ott,[2] Jeffrey R. Foerster,[2,3] Dilip Bethanabhotla,[1] and Mingyue Ji[1]

[1] *Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA*
[2] *Intel Corporate Research, 5000 W Chandler Boulevard, Chandler, AZ 85226, USA*
[3] *Intel Corporate Research, 2111 NE 25th Avenue, Hillsboro, OR 97124, USA*

Correspondence should be addressed to Andreas F. Molisch; molisch@usc.edu

Wireless video is the main driver for rapid growth in cellular data traffic. Traditional methods for network capacity increase are very costly and do not exploit the unique features of video, especially asynchronous content reuse. In this paper we give an overview of our work that proposed and detailed a new transmission paradigm exploiting content reuse and the widespread availability of low-cost storage. Our network structure uses caching in helper stations (femtocaching) and/or devices, combined with highly spectrally efficient short-range communications to deliver video files. For femtocaching, we develop optimum storage schemes and dynamic streaming policies that optimize video quality. For caching on devices, combined with device-to-device (D2D) communications, we show that communications within clusters of mobile stations should be used; the cluster size can be adjusted to optimize the tradeoff between frequency reuse and the probability that a device finds a desired file cached by another device in the same cluster. In many situations the network throughput increases linearly with the number of users, and the tradeoff between throughput and outage is better than in traditional base-station centric systems. Simulation results with realistic numbers of users and channel conditions show that network throughput can be increased by two orders of magnitude compared to conventional schemes.

1. Introduction

Demand for video content over wireless networks has grown significantly in recent years and shows no sign of letting up. According to the *Cisco Visual Networking Index* mobile forecast for 2012–2017, mobile video data is expected to grow at a compound annual growth rate of 75 percent to 7.4 exabyes (one million gigabytes) by 2017 [1]. By this time, it is expected to be 66.5 percent of global mobile traffic data (11.2 exabytes), up from 51 percent in 2012. We expect both broadcast and on-demand services will continue to expand, including traditional services like streaming TV content (e.g., sporting events) and newer services like video Twitter, video blogging, cloud-based live video broadcasting, and mobile-to-mobile video conferencing and sharing. Meanwhile, hardware platforms (smart phones, tablets, notebooks, television/set-top boxes, and in-vehicle infotainment systems) continue to push the envelope in performance and graphical quality. More capable processors, better performing graphics, increased storage capacities, and larger displays make devices more powerful and intelligent than ever before. With this increase in device capability comes a corresponding increase in demand for high-quality video data, for example, increasing demand for high-definition (HD) and 3D data types.

As demand for video traffic continues to grow, the quality of experience (QoE) delivered for this traffic becomes increasingly important. From a 2013 report by Conviva [2], 39.3% of video views experienced buffering, 4% of views failed to start, and 63% of the views experienced low resolution. In addition, other reports [3, 4] have shown that these poor QoE events directly impact a user's engagement in viewing the video and hence potential revenue from videos.

The implications of these trends for future wireless networks are significant. While continued evolution in

spectral efficiency is to be expected, the maturity of MIMO, air interfaces using OFDM/OFDMA, and Shannon capacity-approaching codes mean that such spectral-efficiency improvements will not deliver the increased capacity needed to support future demand for video data. Additional measures like the brute force expansion of wireless infrastructure (number of cells) and the licensing of more spectrum, while clearly addressing the problem of network capacity, may be prohibitively expensive, require significant time to implement, or be infeasible due to prior spectrum allocations which are not easily modified.

Recognizing these challenges, Intel and several industry partners jointly developed a program to explore nonincremental, systems-level solutions through university research. Known as *video aware wireless networks* or simply VAWN, the program considers various approaches to enabling a higher capacity in future wireless networks and to enabling a higher quality of user experience for video and video-based services delivered over wireless networks to intelligent mobile devices. Broad strategies explored in the program include unconventional optimizations in video transport within the network, optimizations in video processing to reduce network transmission requirements and improve user experience, and novel network architectures better suited to address future capacity and quality of service challenges specific to video.

The approach taken by the group at the University of Southern California (including several of the authors) exploits a unique feature of wireless video, namely, the high degree of (asynchronous) content reuse. Based on the fact that storage is cheap and ubiquitous in today's wireless devices, this group developed a new network structure that is based on *replacing backhaul by caching*. This approach, first proposed by the USC group in 2010 [5] and expounded and refined in a series of papers [6–15], is at the center of the present overview.

A first approach for exploiting asynchronous content reuse, termed *Femtocaching*, uses dedicated "helper nodes" that can cache popular files and serve requests from wireless users by enabling localized wireless communication. Such helper nodes are similar to femto-BSs, but they have two key differences: they have *added* a large storage. while they *do not have or need* a high-speed backhaul (Note that storage space has become exceedingly cheap: 2 TByte of data storage capacity, enough to store 1000 movies, cost only about $100.). An even higher density of caching can be achieved by using devices themselves as video caches, in other words, using devices such as tablets and laptops (which nowadays have ample storage) as mobile helper stations [7]. The simplest way of using this storage would have each user cache the most popular files. However, this approach is not efficient because many users are interested in similar files, and thus the same videos will be duplicated on a large number of devices. On the other hand, the cache on each device is too small to cache a reasonably large number of files. Thus, it is preferable that the devices "pool" their caching resources, so that different devices cache different files and then exchange them, when the occasion arises, through short-range, highly spectrally efficient, device-to-device (D2D) communications. If a requesting device does not find the file in its neighborhood (or in its own cache), it obtains the file in the traditional manner from the base station (the base station can also control any occuring D2D communications).

The remainder of the paper is organized as follows: in Section 2, we describe video coding and video streaming techniques, as well as content reuse and viewing habits. The principle of the new network structure is described in Section 3. The placement of files in helper nodes and devices is discussed in Section 4. Fundamental results about throughput and outage in networks with helper stations and D2D communications are described in Sections 5 and 6, respectively. Conclusions in Section 7 round off the paper.

2. Dynamically Managing Video Quality of Experience

2.1. Video Streaming and Quality Management. Wireless channels are inherently dynamic and time-varying depending on a number of factors: (i) movement of device (walking, driving), (ii) changes in the reflectors in the environment (people moving, objects moving), (iii) changes in location (indoors, outdoors), (iv) changes in selected wireless network (WiFi, cellular), and (v) changes in the amount of traffic using the network (i.e., congestion). For data and web-based applications, some latency due to changes in available network capacity, while annoying, can be tolerated. However, for video-based applications (especially interactive video conferencing, but also—depending on buffering capability—for video playback), simply treating data communications as latency tolerant is not sufficient. In order to improve the end user quality of experience (QoE), it is often desirable to adapt the rate of the streamed video using techniques that take into account such factors as the type of video being streamed (fast motion, complex scenes, and interactive), the available capacity of the network, time variations in network and channel state, client device information (screen size, etc.), and playback buffer state. This section will describe some mechanisms for achieving this dynamic adaptation and the role of emerging standards.

Figure 1 illustrates the potential opportunities for managing streaming video traffic in intelligent ways. The figure shows a simplified view of an end-to-end system, including a video server (left), an end rendering device (right), and the network lying in between. Note that opportunities exist in all three domains of the end-to-end system. For example, the video server may accommodate different devices by supporting multiple streaming rates, or multiple copies (formats, bitrates) of the video content. Alternatively, a server may choose to transcode streamed video on the fly. The decision of whether to transcode or store multiple copies may depend on cost, complexity, and performance considerations. Video management opportunities in the network include, for example, support for content caching. How much and how often may depend on the popularity of the content, and whether the details of a particular network architecture make storing content feasible and inexpensive. At the rendering device, opportunities exist for choosing among multiple
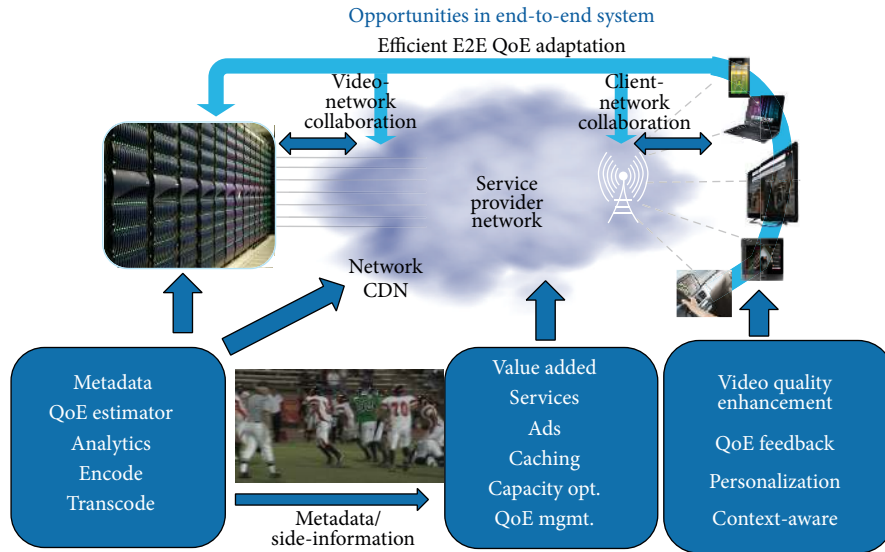
Figure 1: Simplified end-to-end system for video streaming.

video streaming rates given user preferences and making dynamic adjustments during a playback session in response to changes in wireless channel state. Such adjustments may select among the bitrates available from a video source and/or make changes to the display application's buffering strategy.

To better improve user playback experience and to improve the efficiency of data storage and transport, we believe quality of experience (QoE) will be a key metric in future video streaming management. Measures of QoE may take into account the quality of the displayed video (resolution, compression artifacts), rebuffering events, and lost packets. QoE metrics provide an alternative to throughput-based approaches which rely on the often mistaken assumption that higher bitrates invariably mean higher playback quality. A key challenge here, however, is effectively estimating video quality independent of bitrate. Fortunately, a great deal of progress has been made recently by researchers estimating video quality based on both device and content characteristics (see [16–18]). For instance, no-reference approaches to video quality assessment (VQA) can exploit natural video statistics (e.g., DCT) and movement coherency to predict perceived distortions [19]. This information, along with information on channel state, can be used to make automated adjustments in video bitrate at the server or buffering at the display device [17]. In general, QoE metrics enable new opportunities for tighter collaboration between each part of the end-to-end system shown in Figure 1 and for more intelligent control algorithms.

Enhancements to emerging standards are helping to promote QoE-based optimization within end-to-end systems. In particular, standards supporting Dynamic Adaptive Streaming over HTTP (DASH) are being developed by the MPEG and 3GPP standards bodies (see [20–31]). Two recent additions to these standards are (1) the inclusion of QoE feedback metrics from the device to the network and (2) support for providing QoE metrics along with video content that is sent to a device. (In some cases, video QoE metrics

can also be computed directly by the end device.) These additions are important because they enable better system-wide optimization of video transport based on the end user QoE. For example, the device can decide which future segments to request based on the current status of its playback buffer and known quality levels of upcoming segments. This supports a more intelligent balancing of playback quality and rebuffering risk. The network can also make more informed decisions on how to allocate available bandwidth across multiple competing video flows by optimizing the quality jointly across all of them. Using rate-distortion information (a measure of video quality) and playback buffer state for each flow, for instance, a network scheduler can implement QoE-based resource allocation as an alternative to standard proportionally fair throughput schemes.

2.2. Content Reuse. Wireless video distinguishes itself from other wireless content through its strong content reuse; that is, the same content is seen by a large number of people. However, in contrast to TV, the bulk of wireless video traffic is due to asynchronous *video on demand*, where users request video files from some cloud-based server at arbitrary times. As indicated in Section 1, caching can exploit *content overlap*, even in the presence of *asynchronous requests*. In other words, a few popular videos (YouTube clips, sports highlights, and movies) account for a considerable percentage of video traffic on the Internet, even though they are viewed at different times by different people. Numerous experimental studies have indicated that Zipf distributions are good models for the measured popularity of video files [32–34]. Under this model, the frequency of the ith popular file, denoted by $P_r(f)$, is inversely proportional to its rank:

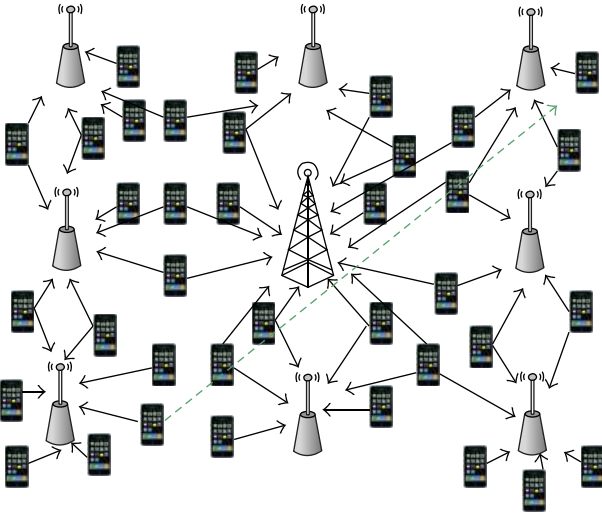$$P_r(f) = \frac{1/f^{\gamma_r}}{\sum_{j=1}^{m} (1/j^{\gamma_r})}, \quad 1 \le f \le m. \tag{1}$$

Figure 2: A sample network scenario with multiple helpers and users and a mobile user (green path).

Figure 3: An illustration of D2D caching networks, where each user (device) can cache M files and we let users be served through high spectral-efficiency D2D links denoted by solid lines in the figure. The dotted lines denote the controlling signal from the base station to the devices.

The Zipf exponent γ_r characterizes the distribution by controlling the relative popularity of files. Larger γ_r exponents correspond to higher content reuse; that is, the first few popular files account for the majority of requests. Here, m is the size of the library of files that are of interest to the set of considered users (note that the library size can be a function of the number of considered users n; we assume in the following that m increases like n^{α}, where $\alpha \geq 0$).

A further important property of the library is that it changes only on a fairly slow timescale (several days or weeks); it can furthermore be shaped by content providers, for example, through pricing policies, or other means.

Note, however, some caveats concerning the general applicability of the work in the remainder of the paper. It applies principally to a setting where a content library of relatively large files (e.g., movies and TV shows) is refreshed relatively slowly (e.g., on a daily basis) and where the number of users consuming such a library is significantly larger than the number of items in the library. This may apply to a possible future implementation of movie services, while collections of short videos (like YouTube) show wider ranges of interests. In short, this paper reflects a set of results and approaches that are relevant in the case where the caching phase (placement of content in the caches) occurs with a clear time-scale separation with respect to the delivery phase (the process of delivering video packets for streaming to the users) and where the size of the content library is moderate with respect to the users' population.

3. Network Structure

3.1. Helper Stations and File Requests.
We first consider the network structure with helper stations. The wireless network consists of multiple helper stations $\mathscr{H}$, talking to multiple users $\mathscr{U}$; a central base station may be present to serve users that cannot find the files they want in the helper stations. An examp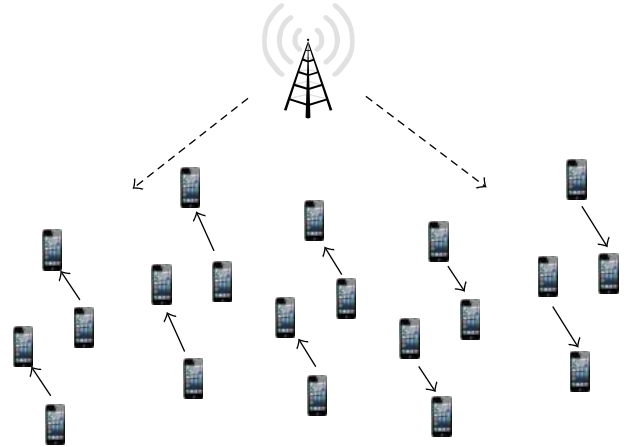le network is shown in Figure 2, which describes a sample network scenario with multiple helpers and users. Each user requests a video file from a library $\mathscr{F}$ of possible files. We denote the set of helpers in the vicinity of user u as $\mathscr{N}(u)$. Similarly, $\mathscr{N}(h)$ denotes the set of users in the vicinity of helper h. The helpers may not have access to the whole video library, because of backhaul constraints and/or caching constraints. In general, we denote by $\mathscr{H}(f)$ the set of helpers that contain file $f \in \mathscr{F}$. Hence, user u requesting file f_u can only download video chunks from helpers in the set $\mathscr{N}(u) \cap \mathscr{H}(f_u)$. In Section 5, we consider the problem of devising a dynamic scheduling scheme such that helpers feed the video files sequentially (chunk by chunk) to the requesting users. Given the high density of helpers, any user is typically in the range of multiple helpers. Hence, in order to cope with user-helper association, load balancing, and intercell interference, an efficient video streaming policy is described in Section 5 which allows the users to dynamically select the helper node to download from and determine adaptively the video quality level of the download.

3.2. Device-to-Device (D2D) Caching Networks.
When users also have the ability of prefetching (video) files, instead of requesting the files from the base station or the helpers, we allow users to make requests from other users and get served via high-spectral-efficiency D2D links (see Figure 3). If the D2D links are not available for some users (see Section 6.2), then these unserved users are treated as in outage and in practice; they can be simply served by the base station or the helpers. To make the network model tractable, we consider the download of the video files instead of streaming and neglect the issue of rate adaptation. In addition, we consider a simple grid structure, which is formed by n user nodes $\mathscr{U} = \{1, \dots, n\}$ placed on a regular grid on the unit square, with minimum distance $1/\sqrt{n}$. (See Figure 4(a); we will replace this grid structure by the random uniform distribution of the nodes when mentioned specifically.) Let each user $u \in \mathscr{U}$
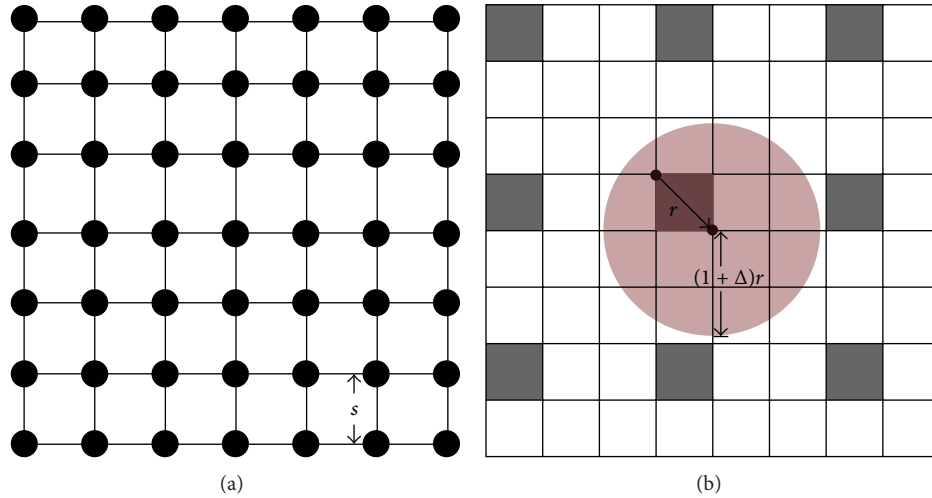
FIGURE 4: (a) Grid network with $n = 49$ nodes (black circles) with minimum separation $s = 1/\sqrt{n}$. (b) An example of single-cell layout and the interference avoidance TDMA scheme. In this figure, each square represents a cluster. The gray squares represent the concurrent transmitting clusters. The red area is the disk where the protocol model imposes no other concurrent transmissions. r is the worst case transmission range and Δ is the interference parameter. We assume a common r for all the transmitter-receiver pairs. In this particular example, the TDMA parameter is $K = 9$.

request a file $f \in \mathcal{F} = \{1, \dots, m\}$ in an i.i.d. manner, according to a given request probability mass function $P_r(f)$, which is assumed to be a Zipf distribution given by (1) with parameter $0 < \gamma_r < 1$ [35]. Moreover, we let each user cache M files. The BS keeps track of which devices can communicate with each other and which files are cached on each device. Such BS-controlled D2D communication is more efficient (and more acceptable to spectrum owners if the communications occur in a licensed band) than traditional uncoordinated peer-to-peer communications.

Communications between nodes follow the protocol model [36] (In the simulations of Section 6.4, we relax the protocol model constraint and take interference into consideration by treating it like noise). Namely, transmission between user nodes u and v is possible if their distance $d(u, v)$ is less than or equal to some fixed transmission range r and if there is no other active transmitter within distance $(1 + \Delta)r$ from destination v, where $\Delta > 0$ is the interference control parameter. Successful transmissions can take place at rate C_r bit/s/Hz, which is a nonincreasing function of the transmission range r [9]. In this model, we do not consider power control (which would allow different transmit powers and thus transmission ranges), for each user. Moreover, we treat r as a design parameter that can be set as a function of m and n (Since the number of possibly requested files m typically varies with the number of users in the system n, and r can vary with n, r can also be a function of m). All communications are assumed to be single-hop (see also Section 6). These model assumptions allow for a sharp analytical characterization of the throughput scaling law including the leading constants. In Section 6, we will see that the schemes designed by this simple model yield promising performance also in realistic channel propagation and interference conditions.

For many of our derivations, we furthermore subdivide the cell into equal-sized, disjoint groups of users that we call

"clusters" of size (radius) r, with g_c nodes in it. To further simplify the mathematical model, we assume that only nodes that are part of the same cluster can communicate with each other. If a user can find the requested file inside the cluster, we say there is one *potential link* in this cluster; when at least one link is scheduled, we say that the cluster is "active." We use an *interference avoidance* scheme, such that at most one link can be active in each cluster on one time-frequency resource.

4. File Placement

The proposed system operates in two steps: (i) file placement (caching) and (ii) delivery. These two processes happen on different timescales: the cache content needs to change only on a timescale of days, weeks, or months, that is, much slower than the actual delivery to the users. Thus, caches could be filled either through a very slow backhaul or through cellular connection at night time, when the spectral resources are not required for other purposes.

4.1. File Placement in Helper Stations. We start out with the case where complete files are stored in the helper stations. If the distance between helpers is large and each MS can connect only to a single helper, each helper should cache the most popular files, in sequence of popularity, until its cache is full. However, when each MS can communicate with multiple helpers, the question on how to best assign files to different helpers becomes more complicated. Consider the case in Figure 5, users U_1 and U_2 would prefer helper H_1 to cache the M most popular files since this minimizes their expected downloading time. Similarly, user U_4 would prefer that helper H_2 also caches the M most popular files. However U_3 would prefer H_1 to cache the M most popular files and H_2 the second M most popular (or the opposite), thus creating a distributed cache of size $2M$ for user U_3. Thus we can
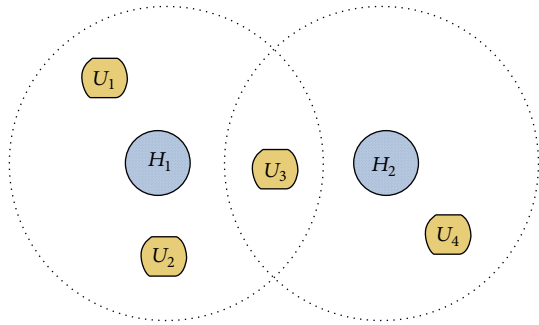
FIGURE 5: Example for caching when users have access to multiple helpers.

see that in the distributed caching problem, the individual objectives of different users may be in conflict, and we need sophisticated algorithms to find an optimum assignment.

Let us assume for the moment that (1) the network topology is known; (2) the long-term average link rates are known; (3) the user demand distribution (file popularity) is known. However, the actual demands are not known beforehand, so that caching placement must be done only based on the *statistics* of the user requests. Our goal is to minimize the average download time. We distinguish further between uncoded and coded caching. In the uncoded case, video-encoded files are cached directly (with the possibility of storing the same file in multiple locations). In the coded case, we consider placing coded chunks of the files on different helper stations, such that obtaining any sufficiently large number of these chunks allows reconstruction of the original video file (e.g., using the scheme in [37]).

In [9] we showed that the uncoded-placement problem is NP-complete. However, it can be formulated as the maximization of a monotone submodular function over matroid constraints, for which a simple greedy strategy achieves at least 1/2 of the optimum value. For the coded case, the optimum cache placement can be formulated as a convex optimization problem, for which optimum solutions can be found through efficient algorithms. In general, the optimum value of delay obtained with the coded optimization is better than the uncoded optimization because any placement matrix with integer entries is a feasible solution to the coded problem. In this sense, the coded optimization is a convex relaxation of the uncoded problem.

We conclude this section by mentioning that the conditions under which we derived the optimum caching are rarely fulfilled in practice. While the user demand distribution $P_r(f)$ may be well estimated and predicted, the network topology is typically time-varying with dynamics comparable or faster than the file transmission; therefore reconfiguring the caches at this time scale is definitely not practical. However, further computer experiments have also shown that the cache distribution obtained when the mobile stations are in "typical" distances from the helpers also provides good performance for various other realizations of random placement of nodes. Furthermore, distributed random caching turns out to be "good enough" as we shall see in Section 6.

Hence, comparing optimal placement with random caching yields useful insight on the potential performance gap lost by a decentralized approach. Interestingly, in any reasonable network configuration it turns out that such a gap is very small.

4.2. File Placement for D2D Communications. Also for D2D communications, the question of which files should be cached by which user are essential. Building on the protocol model explained in Section 3.2, a critical question for each user is whether the file it is interested in can be found within the communication radius r from its current location. In other words, in order to enable D2D communication it is not sufficient that the distance between two users be less than r; users should also find their desired files in the cache of another device with which they can communicate. The decision of what to store can be taken in a centralized or distributed way, called deterministic and random

In deterministic caching a central control (typically the BS) orders the devices to cache specific files. Similar to the situation in femtocaching, we assume that the location of the caching nodes, and the demand distribution, is known. Finding the optimal deterministic file assignment for the general case follows the same principles as for femtocaching outlined above. A simplification occurs when the devices are grouped into *clusters* such that only communication within the cluster is possible (for more details see Section 6). In this case the deterministic caching algorithm is greatly simplified: the devices in the cluster should simply cache the most popular files in a disjoint manner; that is, no file should be cached twice in the cluster. Deterministic caching is only feasible, if the location of the nodes and the channel state information (CSI) are known a priori, and remains constant between the filling of the cache and the actual file transmission; thus it applies only if the caching nodes are fixed wireless devices. It is also useful for providing upper performance bounds for other caching strategies. In random caching, each device randomly and independently caches a set of files according to a common probability mass function. In our earlier papers, we assumed that the caching distribution is also a Zipf distribution, though with a parameter γ_c that is different from γ_r and which has to be optimized for a particular γ_r and r. Since the Zipf distribution is characterized by a single parameter, this description gives important intuitive insights about how concentrated the caching distribution should be.

In [14], we found that the optimal caching distribution P_c^* that maximizes the probability that any user finds its requested file inside its own cluster is given (for a node arrangement on a rectangular grid as described above) by

$$P_c^*(f) = \left[1 - \frac{\nu}{z_f} \right]^+, \quad f = 1, \ldots, m, \tag{2}$$

where $\nu = (m^* - 1) / \sum_{f=1}^{m^*} (1/z_f)$, $z_f = P_r(f)^{1/(M(g_c-1)-1)}$, $m^* = \Theta(\min\{(M/\gamma_r)g_c, m\})$, and $[\Lambda]^+ = \max[\Lambda, 0]$.

5. Adaptive Streaming from Helper Stations

We now turn to the delivery phase, in particular for the femtocaching (helper station). We furthermore concentrate on the case that the video files are *streamed*, that is, replay at the receiver starts before the complete file has been transmitted. Such streaming is widely used for standard video-on-demand systems, using protocols such as Microsoft Smooth Streaming (Silverlight), Apple HTTP Live Streaming, and 3GPP Dynamic Adaptive Streaming over HTTP (DASH). We have adapted such on-demand streaming to our caching architectures, in particular the network setup with helper stations. Dividing each video stream into chunks, we solve the problem of "which user should get a video 'chunk', at what quality, from which helper station."

5.1. Problem Formulation. We represent a video file as a sequence of chunks of equal duration. Each chunk may contain a different number of source-encoded bits, due to variable bit-rate (VBR) coding (see Section 2), and the same video file is encoded at different quality levels, such that lower quality levels correspond to fewer encoded bits. These quantities can vary across video files, and even for the same video they can vary across both chunks and quality levels. For example, the same compression level may produce a different user quality index as well as a different bit requirement from one chunk to the next, depending on if the video chunk is showing a constant blue sky or a busy city street.

In our system, the requested chunks are queued at the helpers, and each helper h maintains a queue Q_{hu} pointing at each of the users u in its vicinity. We pose the *network utility maximization* (NUM) problem of maximizing a concave and component-wise nondecreasing network utility function $\phi_u(\cdot)$ of the users' long-term average quality indices $\overline{D}_u$, subject to stability of the queues Q_{hu} at all the helpers. The concavity of the network utility function imposes some desired notion of fairness between the users. The problem formulation is given as follows:

$$\text{maximize} \quad \sum_u \phi_u\left(\overline{D}_u\right)$$
$$\text{subject to} \quad \overline{Q}_{hu} < \infty \quad \forall (h, u). \tag{3}$$

We solve this problem in [11] using the Lyapunov Drift Plus Penalty approach and obtain a policy that decomposes naturally into two distinct operations that can be implemented in a decentralized fashion: (1) congestion control; (2) transmission scheduling.

5.2. Congestion Control. Congestion control decisions are made at each streaming user, which decides from which helper to request the next chunk and at which quality index this shall be downloaded. For every time slot t, each $u \in \mathcal{U}$ chooses the helper in its neighborhood having the shortest queue; that is,

$$h_u^*(t) = \text{argmin}\left\{Q_{hu}(t) : h \in \mathcal{N}(u) \cap \mathcal{H}(f_u)\right\}. \tag{4}$$

Then, it determines the quality level $m_u(t)$ of the requested chunk at time t as follows:

$$m_u(t) = \text{argmin}\left\{Q_{h_u^*(t)u}(t) B_{f_u}(m, t) - \Theta_u(t) D_{f_u}(m, t)\right\}, \tag{5}$$

where $B_{f_u}(m, t)$ and $D_{f_u}(m, t)$ are the size in bits and the quality index (could be some subjective measure of video quality, for example, SSIM (structural similarity index)), respectively, of chunk t at quality level m. $\Theta_u(t)$ is a virtual queue introduced to solve the NUM problem. Notice that the streaming of the video file f_u may be handled by different helpers across the streaming session, but each individual chunk is entirely downloaded from a single helper. Notice also that in order to compute the above quantities, each user needs to know only *local information* formed by the queue backlog $Q_{hu}(t)$ and the locally computed virtual queue value $\Theta_u(t)$. This scheme is reminiscent of the current adaptive streaming technology for video on demand systems, referred to as DASH (Dynamic Adaptive Streaming over HTTP) [26, 38], where the client (user) progressively fetches a video file by downloading successive chunks and makes adaptive decisions on the quality level based on its current knowledge of the congestion of the underlying server-client connection. Our policy generalizes DASH by allowing the client to dynamically select the least backlogged server, for each chunk.

5.3. Transmission Scheduling. At time slot t, the general transmission scheduling consists of maximizing the weighted sum rate of the transmission rates achievable at scheduling slot t. Namely, the network of helpers must solve the max-weighted sum rate (MWSR) problem:

$$\text{maximize} \quad \sum_{h \in \mathcal{H}} \sum_{u \in \mathcal{N}(h)} Q_{hu}(t) \mu_{hu}(t)$$
$$\text{subject to} \quad \boldsymbol{\mu}(t) \in \mathcal{R}(t), \tag{6}$$

where $\mathcal{R}(t)$ is the region of achievable rates supported by the network at time t and $\mu_{hu}(t)$ is the scheduled rate from helper h to user u in time slot t. We particularize the above general MWSR problem to a simple physical layer system.

Macrodiversity. In this physical layer system, referred to as "macrodiversity," the users can decode multiple data streams from multiple helpers if they are scheduled with nonzero rate on the same slot. In this case, the rate region $\mathcal{R}(t)$ is given by the Cartesian product of the following orthogonal access regions:

$$\sum_{u \in \mathcal{N}(h)} \frac{\mu_{hu}(t)}{C_{hu}(t)} \leq 1, \quad \forall h \in \mathcal{H}, \tag{7}$$

where $C_{hu}(t)$ is the peak rate from helper h to user u in time slot t. In the macrodiversity system, the general MWSR problem (6) decomposes into individual problems, to be solved in a decentralized way at each helper node. The solution is given by each helper h independently choosing the user $u_h^*(t)$ given by

$$u_h^*(t) = \text{argmax}\left\{Q_{hu}(t) C_{hu}(t) : u \in \mathcal{N}(h)\right\}, \tag{8}$$

with rate vector given by $\mu_{hu_h^*(t)}(t) = C_{hu_h^*(t)}(t)$ and $\mu_{hu}(t) = 0$ for all $u \neq u_h^*(t)$. Notice that here, unlike conventional cellular systems, we do not assign a fixed set of users to each helper. In contrast, the helper-user association is dynamic and results from the transmission scheduling decision. Notice also that despite the fact that each helper h is allowed to serve its queues with rates $\mu_{hu}(t)$ satisfying (7), the proposed policy allocates the whole tth downlink slot to a single user $u^* \in \mathcal{N}(h)$, served at its own peak-rate $C_{hu^*}(t)$.

5.4. Algorithm Performance. It can be shown that the time average utility achieved by the proposed policy comes within $O(1/V)$ of the utility of a genie-aided T-slot look ahead policy for any arbitrary sample path with a $O(V)$ tradeoff in time averaged backlog. Thus, the scheme provably achieves optimality of the network utility function under dynamic and arbitrarily changing network conditions; details of the proof can be found in [11].

5.5. Prebuffering and Rebuffering Chunks. The NUM problem formulation (3) does not take into account the possibility of stall events, that is, chunks that are not delivered within their playback deadline. This simplification has the advantage of yielding the simple and decentralized scheduling policy described in the previous sections. However, in order to make such a policy useful in practice we have to force the system to work in the *smooth streaming regime*, that is, in the regime where the stall events have small probability. This can be done by adaptively determining the prebuffering time T_u for each user u on the basis of an estimate of the largest delay of queues $\{Q_{hu} : h \in \mathcal{N}(u)\}$.

We define the size of the playback buffer Ψ_t as the number of playable chunks in the buffer not yet played. Without loss of generality, assume that the streaming session starts at $t = 1$. Then, Ψ_t is recursively given by the updating equation: ($1\{\mathcal{K}\}$ denotes the indicator function of a condition or event $\mathcal{K}$.)

$$\Psi_t = \max\left\{\Psi_{t-1} - 1\left\{t > T_u\right\}, 0\right\} + \left|a_t\right|, \tag{9}$$

where $|a_t|$ is the number of chunks that are completely downloaded in slot t. Let A_k denote the time slot in which chunk k arrives at the user and let W_k denote the delay with which chunk k is delivered. Note that the longest period during which Ψ_t is not incremented is given by the maximum delay to deliver chunks. Thus, each user u needs to adaptively estimate W_k in order to choose T_u. In the proposed method, at each time $t = 1, 2, \ldots$, user u calculates the maximum observed delay E_t in a sliding window of size Δ, by letting

$$E_t = \max\left\{W_k : t - \Delta + 1 \leq A_k \leq t\right\}. \tag{10}$$

Finally, user u starts its playback when Ψ_t crosses the level ξE_t, that is, $T_u = \min\{t : \Psi_t \geq \xi E_t\}$, where ξ is a tuning parameter. If a stall event occurs at time t, that is, $\Psi_t = 0$ for $t > T_u$, the algorithm enters a rebuffering phase in which the same algorithm presented above is employed again to determine the new instant $t + T_u + 1$ at which playback is restarted.

5.6. Extensions. In [12], we consider extensions and improvements of our work. In Sections 5.3 and 5.2, we treated the case of single-antenna base stations and, starting from a network utility maximization (NUM) formulation, we devised a "push" scheduling policy, where users place requests to sequential video chunks to possibly different base stations with adaptive video quality, and base stations schedule their downlink transmissions in order to stabilize their transmission queues. In [12], we consider a "pull" strategy, where every user maintains a request queue, such that users keep track of the video chunks that are effectively delivered. The pull scheme allows to download the chunks in the playback order without skipping or missing them. In addition, motivated by the recent/forthcoming progress in small cell networks (e.g., in wave-2 of the recent IEEE 802.11ac standard), we extend our dynamic streaming approach to the case of base stations capable of multiuser MIMO downlink, that is, serving multiple users on the same time-frequency slot by spatial multiplexing. By exploiting the "channel hardening" effect of high dimensional MIMO channels, we devise a low complexity user selection scheme to solve the underlying max-weighted rate scheduling (6), which can be easily implemented and runs independently at each base station.

5.7. Preliminary Implementation. As observed in Sections 5.3 and 5.2, users send their chunk requests to the helpers having the shortest queue pointing at them. Then, transmission scheduling decisions are made by each helper, which maximizes at each scheduling decision time its downlink weighted sum rate where the weights are provided by the queue lengths. The scheme can be implemented in a decentralized manner, as long as each user knows the lengths of the queues of its serving helpers, and each helper knows the individual downlink rate supported to each served user. Queue lengths and link rates represent rather standard protocol overhead information in any suitable wireless scheduling scheme. We have also implemented a version of such scheme on a testbed formed by Android smartphones and tablets, using standard WiFi MAC/PHY [10].

6. Performance of D2D Caching Networks

We now turn to D2D networks, that is, architectures where the devices themselves act as caches. In contrast to our analysis of femtocaching, we consider here only the download of video *files* (i.e., no streaming) and also neglect the issue of video rate adaptation (these are topics of ongoing research). In this section, we first outline the principle and intuitive insights. We then discuss the fundamental scaling laws, both for the sum throughput in the cell (disregarding any fairness considerations) and for the tradeoff between throughput and outage. Combining D2D transmission with coding and multicasting is also discussed.

6.1. Principle and Mathematical Model. As outlined in Section 3.2, we consider a network where each device can cache a fixed number M video files and send them, upon request, to other devices nearby. If a device cannot obtain a file through D2D communications, it can obtain it from

a macrocellular base station (BS) through conventional cellular transmission.

Consider a setup in which clustering is used (see Section 3.2) and assume furthermore deterministic caching. The main performance factor that can be influenced by the system designer is the cluster size; this is regulated through the transmit power (we assume that it is the same for all users in a cell but can be optimized as a function of user density, library size, and size of the caches). Increasing cluster size increases the probability for finding the desired file in the cluster, while it decreases the (spatial) reuse of time-frequency transmission resources.

There are a number of different criteria for optimizing the system parameters. One obvious candidate is the total network throughput. It is maximized by maximizing the number of active clusters. In [39], we showed that, for deterministic caching, the expected throughput can be computed as

$$E\{T\} = \frac{1}{r^2} \sum_{k=0}^{n} \left(1 - \prod_{i=1}^{k} \left(1 - \left(P_{\mathrm{CVC}}(k) - P_{\mathrm{r}}(f_i) \right) \right) \right) \tag{11}$$
$$\times \Pr[K = k],$$

where $P_{\mathrm{CVC}}(k)$ is the probability that the requested file is in the Common Virtual Cache (the union of all caches in the cluster), that is, among the k most popular files. $\Pr[K = k]$, the probability that there are k users in a cluster, is deterministic for the rectangular grid arrangement, and

$$\Pr[K = k] = \binom{n}{k} \left(r^2 \right)^k \left(1 - r^2 \right)^{n-k}, \tag{12}$$

for random node placement.

6.2. Theoretical Scaling Laws Analysis. We now turn to scaling laws, that is, determine how the capacity scales up as more and more users are introduced into the network. We are dealing with "dense" networks, such that the user density increases, while the area covered by a cell remains the same. As mentioned in Section 4.2, for the achievable caching scheme, we consider a simple "decentralized" random caching strategy, where each user caches M files chosen independently on the library $\mathcal{F}$ with probability $P_c^*(f)$ given by (2).

We furthermore deal again with the "clustered" case; that is, the network is divided into clusters of equal size $g_c(m)$. A system admission control scheme decides whether to serve potential links or ignore them. The served potential links in the same cluster are scheduled with equal probability (or, equivalently, in round robin), such that all admitted user requests have the same average throughput $\mathbb{E}[T_u] = \overline{T}_{\min}$ (see [14] for formal definitions), for all users u, where expectation is with respect to the random user requests, random caching, and the link scheduling policy (which may be randomized or deterministic, as a special case). To avoid interference between clusters, we use a time-frequency reuse scheme [40, Ch. 17] with parameter K as shown in Figure 4(b). In

particular, we can pick $K = (\lceil \sqrt{2}(1 + \Delta) \rceil + 1)^2$, where Δ is the interference parameter defined in the protocol model.

In [8] we established lower and upper bounds for the throughput of D2D communications (this was done under the assumption of random node distribution and caching according to a Zipf distribution). The main conclusion from the scaling law is that for highly concentrated demand distribution, $\gamma_r > 1$, the throughput scales linearly with the number of users, or equivalently the per-user throughput remains constant as the user density increases; the number of users in a cluster also stays constant. For heavy-tailed demand distributions, the throughput of the system increases only sublinearly, as the clusters have to become larger (in terms of number of nodes in the cluster), to be able to find requested files within the caches of the cluster members.

In [14] we tightened the bounds and extended them to the case of throughput-outage tradeoff. Qualitatively (for formal definition see [14]), we say that a user is in outage if the user cannot be served in the D2D network. This can be caused by the fact that (i) the file requested by the user cannot be found in the user's own cluster and (ii) that the system admission control decides to ignore the request. We define the outage probability p_o as the average fraction of users in outage. At this point, we can define the throughput-outage tradeoff as follows.

Definition 1 (throughput-outage tradeoff). For a given network and request probability mass function $\{P_r(f) : f \in \mathcal{F}\}$, an outage-throughput pair (p, t) is *achievable* if there exists a cache placement scheme and an admission control and transmission scheduling policy with outage probability $p_o \leq p$ and minimum per-user average throughput $\overline{T}_{\min} \geq t$. The outage-throughput achievable region $\mathcal{T}(P_r, n, m)$ is the closure of all achievable outage-throughput pairs (p, t). In particular, we let $T^*(p) = \sup\{t : (p, t) \in \mathcal{T}(P_r, n, m)\}$.

Notice that $T^*(p)$ is the result of the optimization problem:

$$\begin{aligned} \text{maximize} \quad & \overline{T}_{\min} \\ \text{subject to} \quad & p_o \leq p, \end{aligned} \tag{13}$$

where the maximization is with respect to the cache placement and transmission policies. Hence, it is immediate to see that $T^*(p)$ is nondecreasing in p.

The following results are proved in [14] and yield scaling law of the optimal throughput-outage tradeoff under the clustering transmission scheme defined above.

Although the results of [14] are more general, here we focus on the most relevant regime of the scaling of the file library size with the number of users, referred to as "small library size" in [14]. Namely, we assume that $\lim_{n \to \infty}(m^\alpha/n) = 0$, where $\alpha = (1 - \gamma_r)/(2 - \gamma_r)$. Since $\gamma_r \in (0, 1)$, we have $\alpha < 1/2$. This means that the library size m can grow even faster than quadratically with the number of users n. In practice, however, the most interesting case is where m is sublinear with respect to n (see [14] for justifications.).

Remarkably, any scaling of m versus n slower than $n^{1/\alpha}$ is captured by the following result.

Theorem 2. *Assume* $\lim_{n \to \infty}(m^\alpha/n) = 0$. *Then, the throughput-outage tradeoff achievable by one-hop D2D network with random caching and clustering behaves as follows:*

$$T^*(p) \geq \begin{cases} \dfrac{C_r}{K}\dfrac{M}{\rho_1 m} + \delta_1(m), \\[4pt] \quad p = (1-\gamma_r)\,e^{\gamma_r - \rho_1}, \\[6pt] \dfrac{C_r A}{K}\dfrac{M}{m(1-p)^{1/(1-\gamma_r)}} + \delta_2(m), \\[4pt] \quad p = 1 - \gamma_r{}^{\gamma_r}\left(\dfrac{Mg_c}{m}\right)^{1-\gamma_r}, \\[6pt] \dfrac{C_r B}{K}m^{-\alpha} + \delta_3(m), \\[4pt] \quad 1 - \gamma_r{}^{\gamma_r}M^{1-\gamma_r}\rho_2{}^{1-\gamma_r}m^{-\alpha} \\[4pt] \qquad \leq p \leq 1 - a(\gamma_r)\,m^{-\alpha}, \\[6pt] \dfrac{C_r D}{K}m^{-\alpha} + \delta_4(m), \\[4pt] \quad p \geq 1 - a(\gamma_r)\,m^{-\alpha}, \end{cases} \tag{14}$$

where $a(\gamma_r)$, A, B, *and* D *are constants depending on* γ_r *and* M, *which can be found in [14] and where* ρ_1 *and* ρ_2 *are positive parameters satisfying* $\rho_1 \geq \gamma_r$ *and* $\rho_2 \geq ((1 - \gamma_r)/\gamma_r{}^{\gamma_r}M^{1-\gamma_r})^{1/(2-\gamma_r)}$. *The cluster size* g_c *is any function of* m *satisfying* $g_c = \omega(m^\alpha)$ *and* $g_c \leq \gamma_r m/M$. *The functions* $\delta_i(m)$, $i = 1, 2, 3, 4$ *are vanishing for* $m \to \infty$ *with the following orders* $\delta_1(m) = o(M/m)$, $\delta_2(m) = o(M/m(1-p)^{1/(1-\gamma_r)})$, *and* $\delta_3(m)$, $\delta_4(m) = o(m^{-\alpha})$.

The dominant term in (14) can accurately capture the system performance even in the finite-dimensional case shown by simulations in Figure 6. Further, also in [14], we can show that the achievable throughput-outage tradeoff given by (14) is order optimal. When $Mn \geq m$ (the whole library can be cached in the network), for arbitrarily small outage probability, by using (14), the per-user throughput scales as $T^*(p) = \Theta(M/m)$. This means that the per-user throughput is independent of the number of users (or in other words, the network throughput increases linearly with the number of users, as already indicated above). Furthermore, the throughput grows linearly with M. This can be very attractive since, for example, in order to double the throughput, instead of increasing the bandwidth or power, we can just double the (cheap) storage capacity per user.

Interestingly, our result shown by (14) coincides with the achievable throughput by using the subpacketized caching and coded multicasting algorithms in [13, 41]. However, in realistic channel assumptions, the result is shown in Section 6.4.

6.3. Coded Caching and Multicasting. From the previous analysis of the D2D caching network, one important property
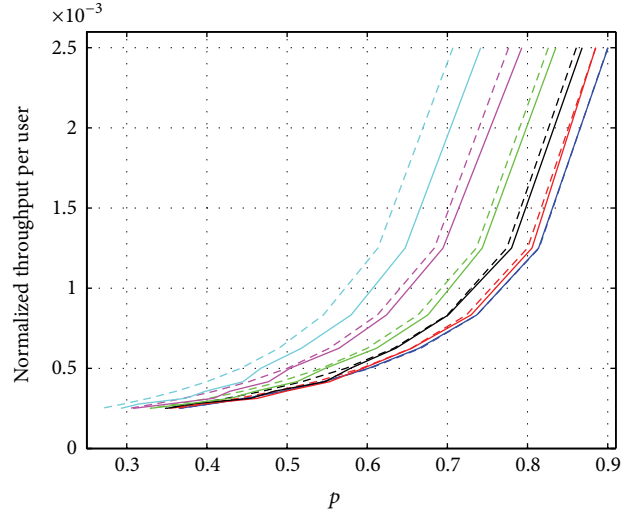

FIGURE 6: Comparison between the normalized theoretical result (solid lines) and normalized simulated result (dashed lines) in terms of the minimum throughput per user versus outage probability. The throughput is normalized by C_r, so that it is independent of the link rate. We assume $m = 1000$, $n = 10000$, $M = 1$, and reuse factor $K = 4$. The parameter γ_r for the Zipf distribution varies from 0.1 to 0.6, which are shown from the right (blue) to the left (cyan). The theoretical curves show the plots of the dominating term in (14) divided by C_r [14].
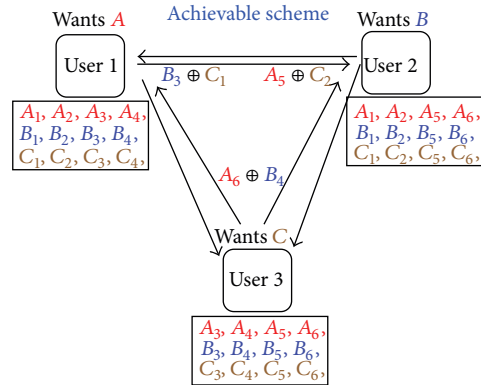

FIGURE 7: Illustration of the example of 3 users, 3 files, and $M = 2$, achieving 1/2 transmissions in terms of file. We divide each file into 6 packets (e.g., A is divided into $A_1, \ldots, A_6$.) User 1 requests A; user 2 requests B; and user 3 requests C. The cached packets are shown in the rectangles under each user. For the delivery phase, user 1 transmits $B_3 \oplus C_1$; user 2 transmits $A_5 \oplus C_2$; and user 3 transmits $A_6 \oplus B_4$. The normalized number of transmissions is $3 \cdot (1/6) = 1/2$, which is also information theoretically optimal for this network [13].

of the proposed scheme is that in both the caching phase and the delivery phase, an uncoded approach is applied. The gain of the throughput is mainly obtained by spatial reuse (TDMA). At this point, a natural question to ask is whether coded multicasting for D2D transmissions can provide an additional gain or whether the coding gain and the spatial reuse gain can accumulate. In [13], we designed a subpacketized caching and a network-coded delivery scheme

for the D2D caching networks. The schemes are best to be explained by the example shown in Figure 7, where we assume no spatial reuse can be used, or only one transmission per time-frequency slot is allowed but the transmission range can cover the whole network. This scheme can be generalized to any n, m, M. Without using spatial reuse, for zero outage, the achievable normalized number of transmissions such that every user can successfully decode is $(m/M)(1 - M/m)$ (We normalize the number of transmissions by the file size, which is assumed to be the same for all the files). Which is surprisingly almost the same as the result shown in [41], where instead of D2D communications, one central server (base station) which has access to all the files multicasts coded packets. In addition, it also has the same scaling law as the throughput by using our previously proposed decentralized caching and uncoded delivery scheme. (Notice that the reciprocal of the number of transmissions is proportional to the throughput under our protocol model assumption.) Moreover, it can be shown that there is no further gain when spatial reuse is also exploited. In other words, the gains of spatial reuse and coding cannot accumulate. Intuitively, if spatial reuse is not allowed, a complicated caching scheme can be designed such that one transmission can be useful for as many users as possible. While if we reduce transmission range and perform our scheme in one cluster as shown in Figure 4(b), then the number of users benefitted by one transmission is reduced but the D2D transmissions can operate simultaneously at a higher rate. Moreover, the complexity of caching subpacketization and coding can also be reduced. Hence, the benefit of coding depends on the actual physical layer throughput (bits/s/Hz) and the caching/coding complexity rather than throughput scaling laws.

6.4. Simulation Results. To see the difference between the performance of the proposed D2D caching network and the state-of-the-art schemes for video streaming, we need to consider the realistic propagation and interference channel model instead of the protocol model. One reason is that as mentioned in Section 6.2, for small outage probability, the throughput of the proposed D2D scheme has the same scaling laws as the coded multicasting scheme in [41]. The state-of-the-art schemes that will be compared with are conventional unicasting, harmonic broadcasting, and coded multicasting, whose details can be found in [15]. In the following, for practical considerations, the proposed uncoded D2D scheme discussed in Section 6.2 is used for simulations.

For simulations, we considered a network of size 600 m $\times$ 600 m, where we relax the grid structure of the users' distribution and let $n = 10000$ users distributed uniformly. The file library has size $m = 300$ (e.g., 300 popular movies and TV shows to be refreshed on a daily basis at off-peak times by the cellular network). The storage capacity in each user is $M = 20$ and the parameter for the Zipf distribution is $\gamma_r = 0.4$ [35]. We considered a regular pattern of buildings of size 50 m $\times$ 50 m, separated by streets of widths 10 m [15], with indoor, outdoor, indoor-to-outdoor, and outdoor-to-indoor pathloss and shadowing models taken from [42], assuming that D2D links operate at 2.4 GHz (WiFi Direct). We assumed a channel bandwidth of 20 MHz in
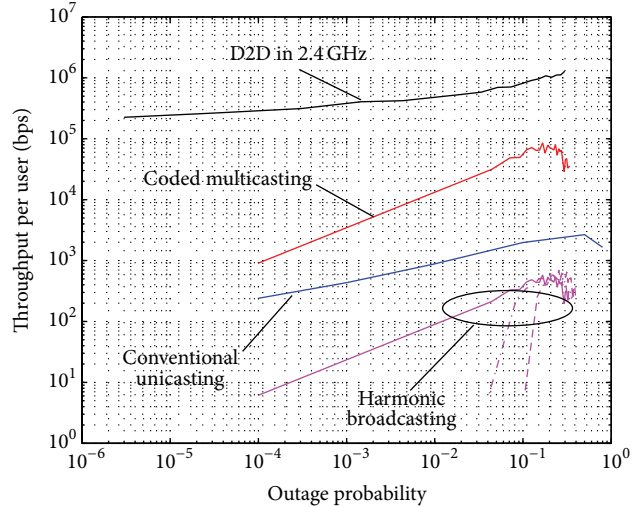

FIGURE 8: Simulation results for the throughput-outage tradeoff for different schemes under the realistic indoor/outdoor propagation environment (for details, see [15]). For harmonic broadcasting with only the m' most popular files, solid line: $m' = 300$; dash-dot line: $m' = 280$; dash line: $m' = 250$. We have $n = 10000$, $m = 300$, $M = 20$, and $\gamma_r = 0.4$. From [15].

order to provide throughput in bit/s. All the details of the simulation parameters, including the pathloss and shadowing models, can be found in [15]. The simulation results of the throughput-outage tradeoff for different schemes are given in Figure 8. We observe that in this realistic propagation scenario the D2D single-hop caching network can provide both large throughput, sufficient for streaming video at standard definition quality, and low outage probability. Also, the D2D caching scheme significantly outperforms the other schemes in the regime of low outage probability. This performance gain is particularly impressive with respect to conventional unicasting and harmonic broadcasting from the base station, which are representative of the current technology. We also note the distinct performance advantages compared to coded multicasting, despite the fact that the two schemes have the same scaling laws. The main reason for this development is that the capacity of multicasting is limited by the "weakest link" between BS and the various MSs, while, for the D2D transmission scheme, short distance transmission (which usually has high SNR, shallow fading, and thus high capacity) determines the overall performance.

It is also worthwhile to notice that the scheduling scheme used in the simulations is based on the clustering structure and the interference avoidance (TDMA) discussed in Section 6.2 without using any advanced interference management scheme such as FlashLinQ [43] and ITLinQ [44], which may provide an even higher gain in terms of throughput for the D2D caching networks.

7. Conclusions

As user demand for video data continues to increase sharply in cellular networks, new approaches are needed to dramatically expand network capacity. This paper has provided

an overview of an approach explored by the University of Southern California as part of the industry-sponsored research program, video aware wireless networks (VAWN). The approach exploits a key feature of wireless video, namely, the high degree of (asynchronous) content reuse across users. To exploit this feature, we propose replacing expensive back-haul infrastructure with inexpensive caching capabilities. This can be realized in two ways: the use of femtocaching or dedicated helper nodes that cache popular files and serve nearby user requests, and the use of user devices themselves to cache and exchange files using device-to-device (D2D) communications. Simulations with realistic settings show that even for relatively low-density deployment of helper stations, throughput can be increased by a factor of five. D2D networks allow in many situations a throughput increase that is linear with the number of users (thus making the per-user throughput independent of the number of users). Simulations in realistic propagation channels, storage capacity settings, video popularity distributions, and user densities show that (for constant outage) the throughput can be two orders of magnitude or more higher than the state-of-the-art multicast systems.

A key issue in our caching approach is that of file placement. In the helper node approach, we show that the problem of minimizing average file downloading time in the uncoded-placement case (video-encoded files are cached directly on help nodes) is NP-complete but can be reformulated and is solvable as a monotone submodular function over matroid constraints. For the coded case (coded chunks of files are placed on different helper stations), optimum cache placement can be formulated and is solvable as a convex optimization problem. Also for the D2D approach, the question of which files to cache is key. Two approaches are deterministic caching in which a BS instructs devices which files to cache (i.e., the most popular and in a disjoint manner) and random caching in which each device randomly caches a set of files according to a probability mass function. It is remarkable that the simple random caching is not only optimum from a scaling law point of view, but also in numerical simulations provides throughputs that are close to the deterministic caching (which is ideal but difficult to realize for time-varying topologies).

An important area of future work is that of predicting user requests. The effectiveness of caching schemes depends not only on the degree of content reuse, but also on our ability to understand and predict request behavior across clusters of users. Furthermore, the approach is predicated on a "time-scale decomposition," namely, that request distributions change much more slowly (over days or weeks) than the time it takes to stream a video (minutes to a couple of hours). For femtocaching, it is noteworthy that the type of users (and thus the requests) within range of a helper station might change over the course of a day; more research on how such spatiotemporal aspects can be predicted and accommodated is required. Similarly, the impact of social networks on user preferences could be exploited. It is noteworthy that not exploiting the space-time correlation of the demands yields only loss of *potential further performance gains* over those already demonstrated here. In short, if we have a correlated

demand process with Zipf first-order statistics, we could further gain by taking into account the correlation structure.

In the D2D sphere, research on new approaches for incentivizing users to participate in cooperative caching schemes is needed. Both femtocaching and D2D caching schemes would benefit from research into multihop cache retrieval schemes and PHY schemes that better exploit advances in wireless communication technology (e.g., multiuser MIMO). In the D2D area, we are/will be investigating how to optimize neighbor discovery, estimating channel conditions and then using the information to make scheduling optimizations, and transmission schemes closely tuned to existing communications standards like WiFi Direct.
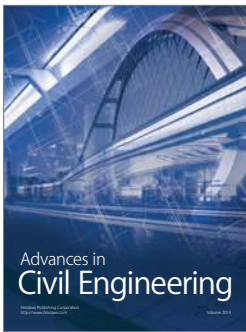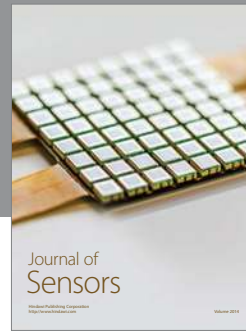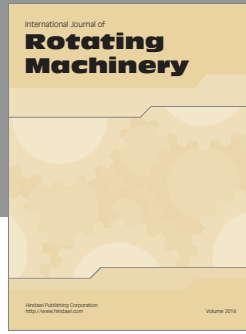
Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] "Cisco visual networking index: global mobile data traffic forecast update, 2012–2017," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI-Forecast_QA.html.

[2] "Conviva viewer experience report," http://www.conviva.com/vxr/.

[3] F. Dobrian, V. Sekar, A. Awan et al., "Understanding the impact of video quality on user engagement," in *Proceedings of the ACM SIGCOMM Conference (SIGCOMM '11)*, pp. 362–373, August 2011.

[4] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Understanding the impact of network dynamics on mobile video user engagement," *SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 367–379, 2014.

[5] https://software.intel.com/en-us/articles/video-aware-wireless-networks.

[6] N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Wireless video content delivery through distributed caching and peer-to-peer gossiping," in *Proceedings of the 45th Asilomar Conference on Signals, Systems and Computers (ASILOMAR '11)*, pp. 1177–1180, Pacific Grove, Calif, USA, November 2011.

[7] N. Golrezaei, A. Molisch, A. G. Dimakis, and G. Caire, "Femto-caching and device-to-device collaboration: a new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, 2013.

[8] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Scaling behavior for device-to-device communications with distributed caching," *IEEE Transactions on Information Theory*, vol. 60, no. 7, pp. 4286–4298, 2014.

[9] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[10] J. Kim, F. Meng, P. Chen et al., "Adaptive video streaming for device-to-device mobile platforms," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom '13)*, pp. 127–129, October 2013.

[11] D. Bethanabhotla, G. Caire, and M. J. Neely, "Adaptive video streaming for wireless networks with multiple users and helpers," http://arxiv.org/abs/1304.8083.

[12] D. Bethanabhotla, G. Caire, and M. J. Neely, "Adaptive video streaming in MU-MIMO networks," Submitted to ISIT 2014, http://arxiv.org/abs/1401.6476.

[13] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of distributed caching in D2D wireless networks," in *Proceedings of the IEEE Information Theory Workshop (ITW '13)*, September 2013.

[14] M. Ji, G. Caire, and A. F. Molisch, "The throughput-outage tradeoff of wireless one-hop caching networks," http://arxiv.org/abs/1312.2637.

[15] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: basic principles and system performance," http://arxiv.org/abs/1305.5216.

[16] J. Foerster and M. Gong, "Industrial column: special Issue on Video Aware Wireless Networks," invited special issue for IEEE Multimedia Communications Technical Committee (MMTC) E-Letters, September 2013.

[17] L. K. Choi, Y. Liao, and A. C. Bovik, "Video QoE models for the compute continuum," Invited Paper for IEEE Multimedia Communications Technical Committee (MMTC) E-Letters, 2013.

[18] Y. Liao, A. Younkin, J. Foerster, and P. Corriveau, "Achieving high QoE across the compute continuum: how compression, content, and devices interact," in *Proceedings of the 7th International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, Scottsdale, Ariz, USA, 2013.

[19] M. A. Saad, A. C. Bovik, and C. Charrier, "Blind prediction of natural video quality," *IEEE Transactions on Image Processing*, vol. 23, no. 3, pp. 1352–1365, 2014.

[20] ISO, "Information technology—dynamic adaptive streaming over http (dash)—part 1: media presentation description and segment formats," ISO/IEC 23009-1:2014, 2014.

[21] "Transparent end-to-end packet switched streaming service (pss); progressive download and dynamic adaptive streaming over http (3gp-dash), version 12.3.0," Tech. Rep. G. T. 26.247, 2014.

[22] ISO, "Information technology—dynamic adaptive streaming over http (dash)—part 2: conformance and reference software," ISO/IEC ISO/IEC 23009-2:2014, 2014.

[23] ISO/IEC DTR 23009-3, "Information technology—dynamic adaptive streaming over HTTP (DASH)—part 3: implementation guidelines".

[24] "Information technology—dynamic adaptive streaming over http (dash)—part 4: segment encryption and authentication," Tech. Rep. ISO/IEC 23009-4, 2013.

[25] I. Sodagar, "The MPEG-dash standard for multimedia streaming over the internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[26] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems (MMSys '11)*, pp. 133–144, ACM, 2011.

[27] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 20–27, 2012.

[28] V. Ramamurthi and O. Oyman, "Video-QoE aware radio resource allocation for HTTP adaptive streaming," in *Proceedings of the IEEE ICC 2014-Communication QoS, Reliability and Modeling Symposium*, 2014.

[29] O. Oyman, U. Kumar, V. Ramamurthi, M. Rehan, and M. Morsi, "Dynamic adaptive streaming over http: standards and technology," Invited Paper for IEEE Multimedia Communications Technical Committee (MMTC) E-Letters, 2013.

[30] O. Oyman, "Dynamic adaptive streaming over http (dash) standardization at mpeg and 3gpp," in *Proceedings of the IEEE Multimedia Communications Technical Committee (MMTC '13)*, E-Letters, 2013.

[31] "Optimizing dash delivery over wireless networks," in *Proceedings of the IEEE Multimedia Communications Technical Committee (MMTC '13)*, E-Letters, March 2013.

[32] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahnt, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference (IMC '07)*, pp. 1–14, October 2007.

[33] http://traces.cs.umass.edu/index.php/network/network.

[34] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1447–1460, 2008.

[35] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societie (INFOCOM '99)*, vol. 1, pp. 126–134, March 1999.

[36] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[37] S. Pawar, N. Noorshams, S. El Rouayheb, and K. Ramchandran, "DRESS codes for the storage cloud: Simple randomized constructions," in *Proceedings of the IEEE International Symposium on Information Theory Proceedings (ISIT '11)*, pp. 2338–2342, IEEE, August 2011.

[38] A. Zambelli, *IIS Smooth Streaming Technical Overview*, Microsoft Corporation, 2009.

[39] N. Golrezaei, A. F. Molisch, and A. G. Dimakis, "Base-station assisted device-to-device communications for high-throughput wireless video networks," in *Proceedings of the IEEE International Conference on Communications (ICC '12)*, pp. 7077–7081, June 2012.

[40] A. Molisch, *Wireless Communications*, John Wiley & Sons, 2011.

[41] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," http://arxiv.org/abs/1209.5807.

[42] WINNER-II, "D1. 1.2, WINNER II channel models," 2007.

[43] X. Wu, S. Tavildar, S. Shakkottai, and ETAL, "FlashLinQ: A synchronous distributed scheduler for peer-to-peer ad hoc networks," in *Proceedings of the 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton '10)*, pp. 514–521, October 2010.

[44] N. Naderializadeh and A. S. Avestimehr, "ITLinQ: a new approach for spectrum sharing in device-to-device communication systems," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1139–1151, 2014.

Journal of
Engineering

The Scientific
World Journal

International Journal of
Rotating
Machinery

Journal of
Sensors

International Journal of
Distributed
Sensor Networks

Advances in
Civil Engineering

Journal of
Control Science
and Engineering

Journal of
Robotics

Submit your manuscripts at
http://www.hindawi.com

Journal of
Electrical and Computer
Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and
Observation

Modelling &
Simulation
in Engineering

International Journal of
Aerospace
Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and
Propagation

Active and Passive
Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration