Wireless Video Content Delivery through Coded Distributed Caching

Negin Golrezaei, Karthikeyan Shanmugam, Alexandros G. Dimakis, *Member, IEEE*,
Andreas F. Molisch, *Fellow, IEEE*, and Giuseppe Caire, *Fellow, IEEE*
Dept. of Electrical Eng.
University of Southern California
Los Angeles, CA, USA
emails: {golrezae,kshanmug,dimakis,molisch,caire}@usc.edu

***Abstract*—We suggest a novel approach to handle the ongoing explosive increase in the demand for video content in mobile devices. We envision femtocell-like base stations, which we call helpers, with weak backhaul links but large storage capabilities. These helpers form a wireless distributed caching network that assists the macro base station by handling requests of popular files that have been cached.**

We formalize the wireless distributed caching optimization problem for the case that files are encoded using fountain/MDS codes. We express the problem as a convex optimization. By adding additional variables we reduce it to a linear program. On the practical side, we present a detailed simulation of a university campus scenario covered by a single 3GPP LTE R8 cell and several helper nodes using a simplified 802.11n protocol. We use a real campus trace of video requests and show how distributed caching can increase the number of served users by as much as $600-700\%$.

I. Introduction

Projections indicate that there will be a tremendous increase in mobile video content demand in the next few years [1], which will require a corresponding increase of area spectral efficiency of wireless networks. The shrinking of cell sizes and base stations allows localized communication and hence higher spatial reuse of communication resources [2]. These pico and femtocell networks are usually combined with macrocells into a heterogeneous network and form a promising direction to address the future challenges in mobile content delivery (see e.g. [3] and references therein). In many cases, the bottleneck seems to be the the lack of cost-effective backhaul connectivity of these small base stations to the cellular operator network [3].

In this paper we propose a novel architecture that deals with the backhaul problem by exploiting storage at the femto base stations. These stations, which we henceforth call caching helpers, or simply helpers, form a wireless distributed caching infrastructure. More concretely, the helpers are placed in fixed positions in the cell and are assumed to have (i) large storage capacity, (ii) localized, high-bandwidth wireless communication capabilities with the mobile users, which enable high frequency reuse, and (iii) low-rate backhaul links, which can be wired or wireless. They can cache popular files and serve requests from mobile User Terminals (UTs) by enabling localized communication. Our architecture relies on the observation that *if there is enough content reuse, i.e. many users are requesting the same video content, caching can replace backhaul communication*. This occurs because the most popular files are stored in the cache, and are therefore always available locally to the UTs that are requesting it. Our approach is thus fundamentally different from a heterogeneous network using femto base stations, which do not have caches, and need to obtain any file through their backhaul network when it has been requested locally.

To simplify our exposition we consider a single cell, equipped with a macro base station (BS), serving a large number of UTs with the help of dedicated helpers. If a UT requests a file that is cached in local helpers, the helpers handle the request; the macro BS manages the requests that cannot be handled locally. Clearly, the smaller the percentage of file requests that has to be fulfilled by the macrocell, the larger the number of users that can be served.

This paper builds on our previous work [4] which introduced the idea of adding storage in helpers and discussed the problem of optimizing the distributed caching placement. In this prior work we assume that video files are packetized and these packets were placed in an optimized way into the helpers' caches. We refer to this case as the "uncoded distributed caching problem" and in our prior work [4] we show that finding an optimal placement is computationally intractable (NP-Complete).

In this paper we show a surprising result: if instead of trying to place the packets, we first encode them with a maximum distance separable (MDS) code, the "encoded distributed caching problem" becomes a convex optimization problem that can be solved efficiently.

To illustrate the effectiveness of our architecture, we present a detailed simulation of a university campus scenario covered by a single 3GPP LTE R8 cell and several helpers that use a simplified 802.11n protocol. While real-life traces for mobile video requests are not yet easily available, we use the YouTube request trace data from a study conducted on the University of Massachusetts, Amherst campus in 2008 [5] [6]. We map these requests to mobile users in a fictional university campus and simulate their request trace on our wireless distributed caching system. We find that even very simple caching algorithms can give significant gains and that our optimized coded femtocaching architecture can be used to increase the number of users that can be served by as much as $600-700\%$.

The remainder of the paper is organized as follows: Section II presents the system model, formulates the distributed caching problem, and presents a solution in terms of a linear program. Section III presents simulation results based on the experimental YouTube requests. A summary concludes the paper in Sec. IV.

II. Theoretical analysis

A. Distributed caching placement model and assumptions

We consider a system where video files are requested randomly by the users. The users' requests are redundant, i.e, they may request the same file, at different times, according to some fixed and known popularity distribution. We note that there is a substantial amount of prior work on caching algorithms for web and video content, see e.g. [7], [8], [9] and references therein. To the best of knowledge, all this related work focuses on wired and p2p networks and has a different focus and constraints, compared to the wireless problem we investigate here. One substantial problem involves learning and keeping track of the popularity distribution but in this paper we do not address this problem and focus on creating the best distributed cache for a given request pattern.

We formulate the *wireless distributed caching problem: for a given popularity distribution, storage capacity in the helpers and wireless communication model for the BS to UTs downlink and helpers to UTs links, how should the files be placed in the helpers such that the average sum delay of all users is minimized?* Since users experience shorter delay when they locally download from helpers in their neighborhoods instead of the BS, minimizing the average delay for a given user is equivalent to maximizing the probability of finding the desired content in the helpers within reach. If each UT can communicate to only one helper, the optimal caching policy is simple: each helper should cache the most popular files. When a user has connection to multiple helpers, however, the caching policy becomes non-trivial.

In our formulation, we make the following assumptions:

- The optimal content placement is determined centrally by the BS according to the optimization algorithms discussed later.
- The popularity distribution of the files changes slowly. Typical examples include popular news, containing short videos, which are updated every 2-3 hours, new movies, which are posted every week, new music videos, which are posted (or change popularity) about every month. Invoking a time-scale decomposition, this has two important consequences: (i) the popularity distribution of the files is effectively fixed; furthermore it can be learned by the system, and thus be assumed known for our further considerations. (ii) once the optimal content placement is determined, the operation of actually populating the helpers' caches can take place using weak backhaul links, since the cost of refreshing the helpers' content can be safely neglected.
- All files are assumed to have the same size. This assumption is mainly used for notational convenience, and could be easily lifted. We also assume that each of the original files is encoded using MDS or near-MDS fountain codes [10]. The property we require from our coding scheme is the following: after the original file is separated into k packets, the code produces a stream of coded packets that have the property that *any* k suffice to recover the original k. Fountain codes [11] can be used instead, and offer lower decoding complexity, the potential to dynamically change the number of encoded packets but provide a probabilistic guarantee and require some overhead, i.e. $(1+\epsilon)k$ must be received to recover the original file. We stress that we do not code across different files but only within each file. The benefit that coding provides is that the identity of the coded packets does not matter. Rather, it is *how many* parity symbols of a given file are retrieved from the helpers in the reach of each user. This is the key property that makes the caching problem tractable.
- We assume that the connectivity between UTs and helpers does not change during the transmission of a video file. This requires our users to be fairly static compared to the download time.

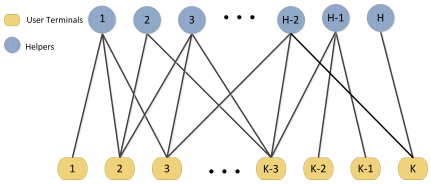

Fig. 1. Example of a connectivity bipartite graph indicating how UTs are connected to helpers

We assume that there are H helpers, K user terminals, and a library of N files, denoted by $\mathcal{F}$. The size of each file is B symbols and the encoded files can be recovered if a user can access to at least to B encoded symbols. The popularity distribution of the files conditioned on the event that a user makes a request is denoted by P_n, for $n = 1, \ldots, N$. The connectivity between users and helpers can be represented in a bipartite graph; one example is shown in figure 1. If there is an edge between helper h and UT k, it means that UT k can communicate reliably with helper h. In practice, the connectivity graph is determined by the location of users and helpers and transmission radius of the helpers.

When a user requests some file n, it first asks its local helpers, i.e., helpers in the neighborhood of the user in the user-helper connectivity graph. The set of available transmitters for user k, denoted by $\mathcal{N}(k)$, includes its local helpers and the BS. We sort transmitters in $\mathcal{N}(k)$ such that the first helper in the set has the highest rate for user k. We assume that rate of the helpers are always higher than the rate of the BS. In other words, we ignore the helpers that have a lower rate than the BS for user k in $\mathcal{N}(k)$. Consequently, the BS is always the last transmitter in $\mathcal{N}(k)$. Upon receiving the request from user k for file n, the first helper in $\mathcal{N}(k)$ starts transmitting all cached parity symbols of file n. If these parity symbols are enough for decoding the file, the user will start decoding; otherwise, the next helper in $\mathcal{N}(k)$ transmits more parity symbols. This process continues until the user has received enough parity symbols. If the parity symbols of all local helpers are insufficient, the BS transmits the required extra information.

B. Distributed caching

We want to find the optimum way of placing fountain/MDS-encoded files in the helpers' caches to minimize the total average delay of all users.

Assume that the set of available transmitters for user k is equal to $\mathcal{N}(k) = \{h_1^k, h_2^k, \ldots, h_{|\mathcal{N}(k)|-1}^k, BS\}$. The reciprocal of the average data rates for the link between user k and its local helper h (including BS as the "helper" number $h = |\mathcal{N}(k)|$) are denoted by $\Omega_k = \{\omega_1^k, \omega_2^k, \ldots, \omega_{|\mathcal{N}(k)|-1}^k, \omega_{|\mathcal{N}(k)|}^k\}$. As a matter of fact, the instantaneous rate of these links fluctuates because of fading and scheduling, so that the instantaneous rate might be zero on some slots. Nevertheless, assuming that each user is a "drop in the ocean", i.e., that the mutual influence of the users on each other is negligible, we may consider each link as a channel of variable capacity, and given average rate that depends only on the link itself, i.e., basically on the distance between the helper and the user. At this point, assuming that the file size is large enough such that long-term time averages are meaningful, the experienced delay for user k for downloading B bits from helper h is given by $B \times \omega_h^k$ where the ω coefficients have the dimension of sec/bit, i.e., they are reciprocal of rates. Regarding ω_{BS}^k, it can be observed and shown analytically that under the standard proportional fairness downlink for high-speed data (HSDPA/Ev-Do) in 3GPP, also used in LTE R8, the delay of downloading from the BS depends on the overall number of active users in the system, on the individual user position in the cell and it is linear with the file size [12].

With fountain/MDS codes, user k should receive B coded symbols of file n in the union of the caches of transmitters in $\mathcal{N}(k)$. If the first $j \leq |\mathcal{N}(k)|$ transmitters in set $\mathcal{N}(k)$ are enough for the recovering some file n while user k cannot recover the file by just receiving the parity symbols of the first $j-1$ transmitters, after normalizing the number of coded symbols to the common file size B, the delay for user k because of file n is equal to:

$$\bar{D}_k^{n,j} = \sum_{h \in A_k^j} \omega_h^k \rho_{hn} + (1 - \sum_{h \in A_k^j} \rho_{hn}) \omega_j^k \tag{1}$$

where $A_k^j = \{h_1^k, \ldots, h_{j-1}^k\}$ and ρ_{hn} is the amount of coded symbols for file n in the cache of helper h. Clearly ρ_{BSn} is equal to 1 for all n. $(1 - \sum_{h \in A_k^j} \rho_{hn})$ is the amount of parity symbols that should be transmitted by the jth transmitter in $\mathcal{N}(k)$. The above expression can be used if $\sum_{h \in A_k^j} \rho_{hn} < 1$ and $\sum_{h \in A_k^j} \rho_{hn} \geq 1$. The matrix whose element in the h-th row and n-th column is $\boldsymbol{\rho} = [\rho_{hn}]$, and it is referred to as the placement matrix. Because of the normalization, all the elements of placement matrix are in $[0, 1]$.

The delay $\bar{D}_k^n$ incurred by user k because of downloading file n is a piecewise-defined affine function of the elements of the placement matrix ρ, defined by :

$$\bar{D}_k^n = \begin{cases} \bar{D}_k^{n,1} & \rho_{h_1^k n} \geq 1 \\ \vdots & \\ \bar{D}_k^{n,j} & \sum_{h \in A_k^j} \rho_{hn} < 1, \ \sum_{h \in A_k^{j+1}} \rho_{hn} \geq 1 \\ \vdots & \\ \bar{D}_k^{n,|N(k)|} & \sum_{h \in N(k) \setminus BS} \rho_{hn} < 1 \end{cases}$$

We have the following result:

Lemma 1: $\bar{D}_k^n$ is a convex function of $\boldsymbol{\rho}$.

Proof: A function that is pointwise maximum of affine functions is a convex function and it is called convex piecewise linear function [13]. Now we show that:

$$\bar{D}_k^n = \max_{j \in \{1,2,\ldots,|\mathcal{N}(k)|\}} \bar{D}_k^{n,j} \tag{2}$$

This means that if for some given j, $\sum_{h \in A_k^j} \rho_{hn} < 1$ and $\sum_{h \in A_k^j} \rho_{hn} \geq 1$, and therefore $\bar{D}_k^n = \bar{D}_k^{n,j}$, we have $\bar{D}_k^{n,j} > \bar{D}_k^{n,i} \forall\ i \neq j$. Thus, from (1), we should show that:

$$\begin{aligned} &\sum_{h \in A_k^j} \omega_h^k \rho_{hn} + (1 - \sum_{h \in A_k^j} \rho_{hn}) \omega_j^k > \\ &\sum_{h \in A_k^i} \omega_h^k \rho_{hn} + (1 - \sum_{h \in A_k^i} \rho_{hn}) \omega_i^k \end{aligned} \tag{3}$$

where the first and second expression in the above equation are respectively $\bar{D}_k^{n,j}$ and $\bar{D}_k^{n,i}$. We only discuss the case $i > j$ since the proof is similar for the case $i < j$. After some manipulations, the above expression for $i > j$ can be written as:

$$(\omega_i^k - \omega_j^k)(\sum_{h \in A_k^{j+1}} \rho_{hn} - 1) + \sum_{h \in A_k^i \setminus A_k^{j+1}} (\omega_i^k - \omega_h^k) \rho_{hn} > 0$$

Since $\sum_{h \in A_k^{j+1}} \rho_{hn} \geq 1$ and $\omega_i^k > \omega_h^k$ for $h \in A_k^i$, the above expression is always true. ■

The average delay of user k can be written as:

$$\bar{D}_k = \sum_{n=1}^{N} P_n \bar{D}_k^n \tag{4}$$

where P_n is the probability of requesting file n. From (3) and (2), the expected delay of all users is:

$$\bar{D} = \sum_{k=1}^{K} \sum_{n=1}^{N} P_n \max_{j \in \{1,2,\ldots,|\mathcal{N}(k)|\}} \bar{D}_k^{n,j}$$

Thus, the placement optimization problem takes on the form:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{K} \sum_{n=1}^{N} P_n \max_{j \in \{1,2,\ldots,|\mathcal{N}(k)|\}} \bar{D}_k^{n,j} \\ \text{subject to} \quad & \sum_{n=1}^{N} \rho_{hn} \leq M, \quad \forall\ h \\ & 0 \leq \rho_{hn} \leq 1, \quad \forall h, n \end{aligned} \tag{5}$$

where the optimization is with respect to $\boldsymbol{\rho}$. The first constraint indicates the cache capacity of each helper is equal to M files. The second constraint shows the amount of parity symbols of file n cached by a helper h is normalized to the size of files. The objective function in (4) is convex function since it is positive weighted sum of convex piecewise linear functions [13]. Hence, we have minimization of convex function over linear constraints which can be solved using convex optimization.

We also can solve the placement optimization problem by converting it to the linear program which easily can be solved using standard techniques. To do this we introduce new variables and new constraints as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N} P_n t_k^n \\ \text{subject to} \quad & D_k^{n,j} + s_k^{n,j} = t_k^n \\ & s_k^{n,j} \geq 0 \quad \forall\, k, n \,\text{and}\, j \in \{1, 2, \ldots, |\mathcal{N}(k)|\} \\ & \sum_{n=1}^{N} \rho_{hn} \leq M, \quad \forall\, h \\ & 0 \leq \rho_{hn} \leq 1, \quad \forall h, n \end{aligned} \tag{6}$$

where $t_k^n = \max_{j\in\{1,2,\ldots,|\mathcal{N}(k)|\}} D_k^{n,j}$ and the optimization is with respect to $\boldsymbol{\rho}$, t_k^ns and $s_k^{n,j}$s.

III. Experimental evaluation

A. Simulation System Parameters

In this section, evaluation of the proposed placement algorithm through simulation in a cell based on 3GPP LTE Release 8 is presented. We assume a macro BS operating with a conventional scheduling policy and helpers with some storage capacity serving users using WiFi-like links. We simulate the system with realistic user request pattern and average delays for downloading are obtained to evaluate the placement algorithm.

We assume a circular cell with no inter-cell interference from other cells. The cell has a radius of $400m$ and users are randomly and independently distributed in the cell with uniform probability distribution. The assumed cell radius is typical in an urban macro cell [14]. The pathloss function (in dB) is taken to be:

$$PL(d(u,v)) = \left\{ \begin{array}{c} 38 + 20 * \log_{10}(d(u,v)) \quad , d(u,v) < 40 \\ 38 + 20 * \log_{10}(40) + \ldots \\ 35 * \log_{10}\left(d(u,v)/40\right) \quad , d(u,v) > 40 \end{array} \right\} \tag{7}$$

where $d(u,v)$ denotes the distance, in meters, between the transmitter (at u) and the receiver (at v) where $u, v \in \mathbb{R}^2$. Let $g(u,v)$ be the received SNR and G_0 be the average transmit power. Since we don't consider any inter-cell interference, for realistic rates we fix G_0 such that received SNR at the cell edge is equal to 0 dB (usually typical SINR at cell edge is -1dB to -4 with frequency reuse 1 [2]).

We assume that the wireless channel is frequency selective and make a standard block-fading approximation of the small-scale Rayleigh fading, such that coherence time is $N_t \times S$ OFDM symbols) and coherence bandwidth is N_f sub-carriers.

TABLE I
Simulation parameters based on LTE specifications.

Parameter of the System	Value Assigned
Usable System badnwidth	18 MHz
Sub-carrier Bandwidth	15 KHz
Parameter N_t	7 OFDM symbs. $= 0.5ms$
Parameter N_f	12 Sub-carriers $= 180KHz$
Smallest Resource allocation slot	$N_t = 7$ OFDM symbs. $\times$ $N_f = 12$ sub-carriers.
Coherence time	$100ms$ = 1400 OFDM symbs. $= S \times N_t$ OFDM symbs. across time
Parameter S	$\frac{100}{0.5} = 200$ slots (N_t OFDM symbs. each)
Actual resource allocation block	$N_t \times N_f \times S = 16800$ OFDM symbs.
Frequency blocks for allocation	100

Fading coefficients are i.i.d. from block to block, and independent across users. The time-frequency structure and the resource allocation slot of the downlink channel in an OFDM TDMA sytem inspired by 3GPP LTE (Rel 8) [15] is described in Table I.

The assumed signal model between the BS (located at the origin) and user k, located at u_k, over time-frequency slot (t, f), with $t = 0, 1, 2, \ldots$ and $f \in \{1, \ldots, F\}$, is given by

$$\mathbf{y}_k(t,f) = \sqrt{g(u_k,0)} H_{k,0}(t,f)\mathbf{x}_0(t,f) + \mathbf{z}_k(t,f), \tag{8}$$

where $H_{k,0}(t,f) \sim \mathcal{CN}(0,1)$ is the Rayleigh fading coefficient on slot (t,f), and $\mathbf{x}_0(t,f) \in \mathbb{C}^T$, $\mathbf{y}_k(t,f) \in \mathbb{C}^T$ and $\mathbf{z}_k(t,f) \in \mathbb{C}^T$ denote the transmit, received and noise vectors, with $\mathbf{z}_k(t,f) \sim \mathcal{CN}(0, \mathbf{I})$, independent across time-frequency. The achievable rate of user k on slot (t,f) is given by

$$R_k(t,f) = \log_2\left(1 + g(u_k,0)|H_{k,0}(t,f)|^2\right). \tag{9}$$

B. Scheduling Policy used

A UT is said to be "idle" if it has no active download and "active" otherwise. Active UTs cannot place more than one request at a time. If they do, all the previous requests are dropped.

We assume *Proportional Fairness Scheduling* (PFS), currently used in Ev-Do and HSDPA high-data rate downlink schemes in 3G [15] with the TDMA constraint. In our system, PFS policy is applied to active users only. Let $K_{\text{on}}(t)$ denote active users at time t. When a user becomes idle, it is eliminated from the scheduling policy. For each resource slot (t,f), choose the an active user $k(t,f)$ subject to the TDMA constraint, that maximizes

$$\sum_{k=1}^{K_{\text{on}}(t)} Q_k(t) \sum_{f=1}^{F} R_k(t,f), \tag{10}$$

where $\{Q_k(t)\}$ are the scheduling weights obtained dynamically by an update scheme based on virtual queues to optimize the PFS utility function. $R_k(t,f)$ is defined in (8). The reader is referred to [16] [17] for more details. The only difference from an infinite backlogged scenario is that the max-weight scheduling is done only for active users.

C. Trace based User Requests

We target a scenario where the BS is overloaded with YouTube (short videos) like video requests and consider the average download delay as the metric of interest. We use the YouTube request trace data from a study conducted on the University of Massachusetts' Amherst campus in 2008 [5] [6]. The study records YouTube requests arising from the wired campus network for several days. We expect wireless video requests to have a different behavior but consider this a reasonable first step in simulating our architecture. We fix all requests to be of size 30 MB and a few minutes ($\approx$ 3 min) playback time. We expect that our conclusions will also hold (in a scaled form) for larger file sizes, playback times and data rates.

All traces used in simulations contain unique users (based on unique IP addresses from trace data). For every user, a time series involving the exact time of request (in seconds) is created and the corresponding video file number (can be distinguished from the trace data) is created. The user requests are simulated exactly as per these traces. We use the trace data [19] for the day $02/19/08$. We choose the busiest four hours of the day, which accounts for 848 unique users and about 4600 requests, and form *trace1*. We derive another trace (named *trace2*) by superposing the two busiest four hours and creating a merged trace containing 1719 (user lists from each four hour period are added up) users and about 9600 requests. We assume we know *a priori* the popularity distribution (number of views vs rank of videos in terms of views) from the data of the entire day. This is used for assigning popularity to the files requested. See Sec II for a discussion of this assumption. The scheduling and request handling take place as per the system specifications in the previous subsections. We compare the performance of three systems.

1) *Baseline* system- Only the BS serves the users.
2) *Popular Helper* system- Helpers and the BS serve the users. Helpers store beforehand the most popular files allowed by their storage capacity.
3) *Coded Helper* system - Helpers are allocated files according to the optimization problem in (4).

The helpers are placed uniformly in a square grid spanning the cellular region. We assume a simple model for the helper-user communication. Every helper has a range of $100m$. This means that the transmission rate decreases from 30 Mbps at $10m$ to 3 Mbps at $100m$. Although simplified, the model can be justified by facts about the latest WiFi standard. The rate decay follows the curve for the case of 2x2 MIMO and 20 MHz bandwidth in Fig 5.9 of [18] but scaled down by a factor of 4 accounting for protocol overhead and the fact that this rate is guaranteed irrespective of other users who may request from the same helper concurrently. For trace1, we also reduce the rate of the BS by a factor of two due to protocol (pilot and signaling) overhead. For trace2 the BS is heavily overloaded and the Baseline system was only providing sufficient quality of service to very few users. For this reason we double the bandwidth of the BS for trace2 to test if helper benefits would remain substantial.

When a user requests a video file, it is recovered from nearby helpers as much as possible. The rest of the file is obtained from the BS. The user reciprocal average rate for the coded algorithm, i.e., ω_k's in (4) are the time average download delay for user k for undropped requests obtained from the simulation done on the Baseline assuming an analytical request model for user requests. In this model, the wait time between two requests is a random variable with a geometric distribution with parameter p. The expected wait time is $1/p$. p can be chosen such that expected wait times are about $4-7$ minutes considering video playback to be 3 min.

To compare the baseline and the helper systems, for every user, we define the Quality of Service (QOS) as the average download delay. A user is *satisfied* when his/her average download delay is below a given QOS (in seconds) threshold. In all simulations, M represents the number of 30 MB files that a helper can cache. In all plots, storage capacity in GB is mentioned and M can be inferred from it. QOS is chosen to be 200 seconds to measure satisfied users. Such a QOS is reasonable for a playback time of 3 minutes for the videos. The number of helpers is 32 in all figures.

In all simulations, Baseline, Popular Helper and Coded Helper systems are compared. Some common conclusions are that the Coded Helper system is strictly better than the Popular Helper System which is better than the Baseline and increasing the storage capacity of helpers increases the number of satisfied users. Now we analyze the results in detail.

Figures 2 and 3 show the number of satisfied users versus the storage capacity of helpers. Figure 2 uses trace1 while figure 3 uses trace2. The number of satisfied users (in figure 2) increases more than $600-700\%$ in both helper systems. We can see even more improvement in trace2 when the number of requests is twice the number of requests in trace1. This indicates the significant advantages of helpers in case of a large number of video requests.

Figures 4 and 5 show the percentage of successful requests versus the storage capacity of each helper for trace1 and trace2. It can be seen that even for large storage capacity the percentage of successful requests does not reach 100%. The reason is that, in our trace data, most users become active for a short time and within that short time they request several video files consecutively. They frequently request another file before the previous one is downloaded completely. We count all incomplete downloads as dropped requests and many of them are due to such rapid request changes. However, as can be seen from the plots, helpers substantially reduce the percentage of dropped requests.

IV. Summary and Conclusions

In this paper we introduced a new method for increasing the throughput of wireless video delivery networks. The key idea is the use of a distributed cache, i.e., helper stations that store the most popular video files, and transmit them, upon request, via short-range wireless links to the user terminals. The caches are low-cost because storage capacity has become exceptionally cheap (according to recent prices, two terabytes cost approximately 100 dollars), while the loading of the files

to the caches can occur through a low-rate (and thus cheap and robust) backhaul links at low demand times. By encoding the files with codes that allow distributed storage, robustness and efficient storage is enhanced. We then formulated and solved the problem of how files should be encoded and assigned to which helpers. Our experimental evaluation uses a realistic LTE-based cellular simulator and a real trace of YouTube requests and demonstrates performance improvements on the order of $600-700\%$ more users at reasonable QoS levels. Our conclusion is that a wireless distributed helper system is a promising way of alleviating the bottlenecks in wireless video delivery.

REFERENCES

[1] Cisco. *http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html*.

[2] A.F. Molisch, *Wireless communications*. 2nd ed., IEEE Press - Wiley, 2011.

[3] V. Chandrasekhar, J. G. Andrews, and A. Gatherer,"Femtocell networks: a survey," *IEEE Commun. Mag.*, 46(9):59-67, Sept. 2008.

[4] N. Golrezaei, K. Shanmugam, A.G. Dimakis, A.F. Molisch and G. Caire, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers," submitted for publication, available on http://arxiv.org/pdf/1109.4179v1.

[5] M. Zink, K. Suh, Y. Gu, and J. Kurose. Watch global, "cache local: YouTube network traffic at a campus networkmeasurements and implications," *Proc. 15th SPIEACM Multimedia Computing and Networking*, 2008.

[6] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network-measurements, models, and implications," *Computer Networks*, 53(4): p.501-514, 2009.

[7] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," *IEEE Proc. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies,* v.1, p.126-134, 1999.

[8] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," *Proc. the twenty-ninth annual ACM symposium on Theory of computing*, p.654-663, 1997.

[9] M. Rabinovich and O. Spatscheck, "Web caching and replication," *SIGMOD Record*, 32(4):107, 2003.

[10] R.M. Roth, *Introduction to coding theory*. Cambridge University Press, 2005.

[11] M. Luby, LT-codes, in Proceedings of the ACM Symposium on Foundations of Computer Science (FOCS), 2002.

[12] S. Borst, "User-level performance of channel-aware scheduling algorithms in wireless data networks," *IEEE Proc. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, v. 1, p. 321-331, 2003.

[13] S,P. Boyd and L. Vandenberghe *Convex optimization*. Cambridge University Press, 2004.

[14] International telecommunication union. *http://www.itu.int/pub/R-REP-M.2135-2008*.

[15] H. Holma and A. Toskala, *LTE for UMTS: OFDMA and SCFDMA based radio access.* John Wiley & Sons Inc, 2009.

[16] L. Georgiadis, M. Neely, M.J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," Now Pub, 2006.

[17] M.J. Neely, "Stochastic Network Optimization with Application to Communication and Queueing Systems," *Synthesis Lectures on Communication Networks*, 3(1):1-211, 2010.

[18] E.Perahia and R.Stacey, *Next generation wireless LANs: throughput, robustness, and reliability in 802.11 n.* Cambridge University Press, 2008.

[19] *http://traces.cs.umass.edu/index.php/Network/Network*.

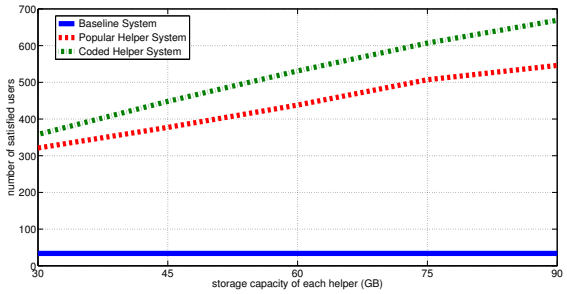

Fig. 2. The number of satisfied users versus the storage capacity of each helper for trace1, number of helpers=32, QOS=200 seconds.

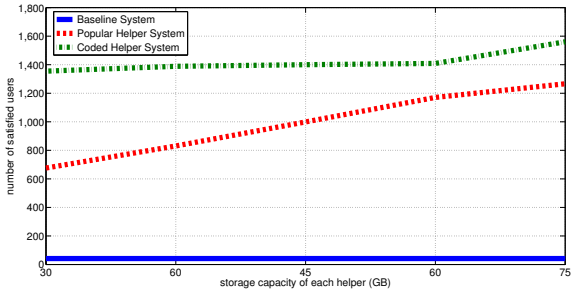

Fig. 3. The number of satisfied users versus the storage capacity of each helper for trace2, number of helpers=32, QOS=200 seconds.

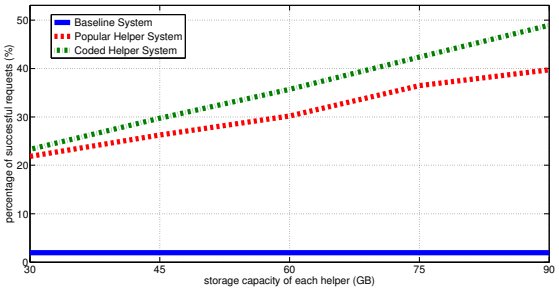

Fig. 4. The parentage of successful requests versus the storage capacity of each helper for trace1, number of helpers=32, QOS=200 seconds.

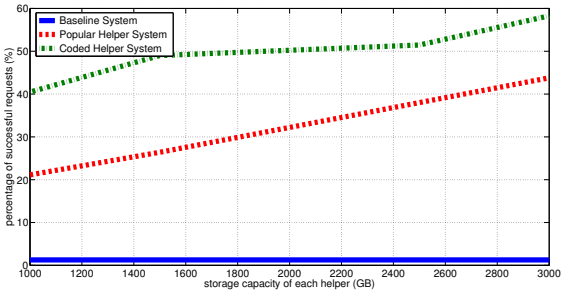

Fig. 5. The parentage of successful requests versus the storage capacity of each helper for trace2, number of helpers=32, QOS=200 seconds.