Alpha Fair Coded Caching

Apostolos Destounis[1], Mari Kobayashi[2], Georgios Paschos [1], Asma Ghorbel[2]
[1] France Research Center, Huawei Technologies Co. Ltd., email: firstname.lastname@huawei.com
[2]Centrale-Supélec, France, email: firstname.lastname@centralesupelec.fr

***Abstract*—The performance of existing *coded caching* schemes is sensitive to the worst channel quality, when applied to wireless channels. In this paper, we address this limitation in the following manner: *in short-term*, we allow transmissions to subsets of users with good channel quality, avoiding users with fades, while *in long-term* we ensure fairness across the different users. Our online delivery scheme combines (i) joint scheduling and power control for the fading broadcast channel, and (ii) congestion control for ensuring the optimal long-term average performance. By restricting the caching operations to *decentralized coded caching* proposed in the literature, we prove that our proposed scheme has near-optimal overall performance with respect to the long-term alpha fairness performance. By tuning the coefficient alpha, the operator can differentiate the user performance in terms of video delivery rates achievable by coded caching. We demonstrate via simulations that our scheme outperforms standard coded caching and unicast opportunistic scheduling, which are identified as special cases of our general framework.**

***Index Terms*—Broadcast channel, coded caching, fairness, Lyapunov optimization.**

I. Introduction

A key challenge for the future wireless networks is the increasing video traffic demand, which reached 70% of total mobile IP traffic in 2015 [1]. Classical downlink systems cannot meet this demand due to limited resource, yielding the per-video throughput vanishing as 1/K with the system dimension K. Recently it was shown that scalable per-video throughput can be achieved if the communications are synergistically designed with caching at the receivers. Indeed, the recent breakthrough of *coded caching* [2] has inspired a rethinking of wireless downlink. By careful selection of sub-file caching and exploitation of the broadcast wireless channel, the transmitted signal is simultaneously useful for decoding at users with different video requests. Although this scheme–theoretically proved to scale well–can potentially resolve the future downlink bottleneck, several limitations hinder its applicability in practical systems [3]. In this work, we take a closer look to the limitations that arise from the fact that *coded caching was originally designed for a symmetric error-free shared link.*

If instead we consider a realistic model for the wireless channel, we observe that a naive application of coded caching faces a *short-term* limitation: since the channel qualities of the users fluctuate over time and our transmissions need to reach all users, the transmissions need to be designed for the worst channel quality. This is in stark contrast with standard downlink techniques, like *opportunistic scheduling* [17], [24], [18], which serve the user with the best instantaneous channel quality. Thus, a first challenge is to discover a way to allow coded caching technique to opportunistically exploit the fading of the wireless channel.

Apart from the fast fading consideration, there is also a *long-term* limitation due to the network topology. The user locations might vary, which leads to consistently poor channel quality for the ill-positioned users. The classical coded caching scheme is designed to deliver equal video shares to all users, which leads to ill-positioned users consuming most of the air time and hence driving the overall system performance to low efficiency. In the literature, this problem has been resolved by the use of fairness among user throughputs [24]. By allowing poorly located users to receive less throughput than others, precious airtime is saved and the overall system performance is greatly increased. Since the sum throughput rate and max-min fairness are typically the two extreme cases, past works have proposed the use of alpha-fairness [15] which allows to select the coefficient α and drive the system to any desirable tradeoff point in between of the two extremes. Previously, the alpha-fair objectives have been studied in the context of (i) multiple user activations [17], (ii) multiple antennas [19] and (iii) broadcast channels [20]. However, here the fairness problem is further complicated by the interplay between scheduling and the coded caching operation. In particular, we wish to shed light into the following questions: *what is the right user grouping and how we should design the codewords to achieve our fairness objective while adapting to changing channel quality?*

To address these questions, we study the content delivery over a realistic block-fading broadcast channel, where the channel quality varies across users and time. In this setting, we design a scheme that decouples transmissions from coding. In the transmission side, we select the multicast user set dynamically depending on the instantaneous channel quality and user urgency captured by queue lengths. In the coding side, we adapt the codeword construction of [6] depending on how fast the transmission side serves each user set. Combining with an appropriate congestion controller, we show that this approach yields our alpha-fair objective. More specifically, our approaches and contributions are summarized below:

1) We impose a novel queueing structure which decomposes the channel scheduling from the codeword construction. Although it is clear that the codeword construction needs to be adaptive to channel variation, our scheme ensures this through our *backpressure* that connects the user queues and the codeword queues. Hence, we are able to show that this decomposition is without loss of optimality.
2) We then provide an online policy consisting of (i) admission control of new files into the system; (ii) combination of files to perform coded caching; (iii) scheduling and power control of codeword transmissions to subset of users

978-3-901882-90-6/17 ©2017 IFIP

on the wireless channel. We prove that the long-term video delivery rate vector achieved by our scheme is a near optimal solution to the alpha-fair optimization problem under the specific coded caching scheme [6].

3) Through numerical examples, we demonstrate the superiority of our approach versus (a) opportunistic scheduling with unicast transmissions and classical network caching (storing a fraction of each video), (b) standard coded caching based on transmitting-to-all.

A. Related work

Since coded caching was first proposed [2] and its potential was recognized by the community, substantial efforts have been devoted to quantify the gain in realistic scenarios, including decentralized placement [6], non-uniform popularities [5], [7], and device-to-device (D2D) networks [4]. A number of recent works replace the original perfect shared link with wireless channels [10], [9], [11]. Commonly in the works with wireless channels, the performance of coded caching is limited by the user in the worst channel condition because the wireless multicast capacity is determined by the worst user [13, Chapter 7.2]. This limitation of coded caching has been recently highlighted in [11], while similar conclusions and some directions are given in [9], [10]. Our work is the first to addresses this aspect by jointly designing the transmissions over the broadcast channel and scheduling appropriate subsets of users.

Most past works deal with *offline* caching in the sense that both cache placement and delivery phases are performed once and do not capture the random and asynchronous nature of video traffic. The papers [8], [12] addressed partly the online nature by studying cache eviction strategies, and delay aspects. In this paper, we explore a different online aspect. Requests for video files arrive in an online fashion, and transmissions are scheduled over time-varying wireless channels.

Online transmission scheduling over wireless channels has been extensively studied in the context of opportunistic scheduling [17] and network utility maximization [14]. Prior works emphasize two fundamental aspects: (a) the balancing of user rates according to fairness and efficiency considerations, and (b) the opportunistic exploitation of the time-varying fading channels. Related to our work are the studies of wireless downlink with broadcast degraded channels; [21] gives a maxweight-type of policy and [22] provides a throughput optimal policy based on a fluid limit analysis. Our work is the first to our knowledge that studies coded caching in this setting. The new element in our study is the joint consideration of user scheduling with codeword construction for the coded caching delivery phase.

II. System Model and Problem Formulation

We study a wireless downlink consisting of a base station and K users. The users are interested in downloading files over the wireless channel.

A. Fair file delivery

The performance metric is the *time average delivery rate of files* to user k, denoted by $\overline{r}_k$. Hence our objective is expressed with respect to the vector of delivery rates $\boldsymbol{r}$. We are interested in the *fair file delivery* problem:

$$\overline{\boldsymbol{r}}^* = \arg\max_{\overline{\boldsymbol{r}}\in\Lambda} \sum_{k=1}^{K} g(\overline{r}_k), \tag{1}$$

where Λ denotes the set of all feasible delivery rate vectors–clarified in the following subsection–and the utility function corresponds to the *alpha fair* family of concave functions obtained by choosing:

$$g(x) = \begin{cases} \frac{(d+x)^{1-\alpha}}{1-\alpha}, \alpha \neq 1 \\ \log(1+x/d), \alpha = 1 \end{cases} \tag{2}$$

for some arbitrarily small $d > 0$ (used to extend the domain of the functions to $x = 0$). Tuning the value of α changes the shape of the utility function and consequently drives the system performance $\overline{\boldsymbol{r}}^*$ to different points: (i) $\alpha = 0$ yields max sum delivery rate, (ii) $\alpha \to \infty$ yields max-min delivery rate [15], (iii) $\alpha = 1$ yields proportionally fair delivery rate [16]. Choosing $\alpha \in (0, 1)$ leads to a tradeoff between max sum and proportionally fair delivery rates.

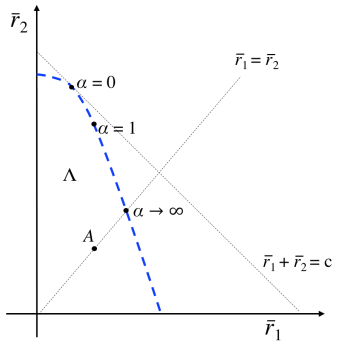


Fig. 1. Illustration of the feasibility region and different performance operating points for $K = 2$ users. Point A corresponds to a naive adaptation of [2] on our channel model, while the rest points are solutions to our fair delivery problem.

The optimization (1) is designed to allow us tweak the performance of the system; we highlight its importance by an example. Suppose that for a 2-user system Λ is given by the convex set shown on figure 1. Different boundary points are obtained as solutions to (1). If we choose $\alpha = 0$, the system is operated at the point that maximizes the sum $\overline{r}_1 + \overline{r}_2$. The choice $\alpha \to \infty$ leads to the maximum r such that $\overline{r}_1 = \overline{r}_2 = r$, while $\alpha = 1$ maximizes the sum of logarithms. The operation point A is obtained when we always broadcast to all users at the weakest user rate and use [2] for coded caching transmissions. Note that this results in a significant loss of efficiency due to the variations of the fading channel, and consequently A lies in the interior of Λ. We may infer that the point $\alpha \to \infty$ is obtained by avoiding transmissions to users with instantaneous poor channel quality but still balancing their throughputs in the long run.

B. Transmission model

To analyze the set of feasible rate vectors Λ we need to zoom in the detailed model of transmissions.

Caching model. There are N equally popular files $W_1, \ldots, W_N$, each F bits long. The files are available to the

base station. User k is equipped with cache memory Z_k of MF bits, where $M \in [0, N]$. Caching placement is performed during off-peak hour, and the goal is to fill the caches up to the memory constraint with selected bits. To this end, we need to select K *caching functions* $\phi_k : \mathbb{F}_2^{NF} \to \mathbb{F}_2^{MF}$ which map the files $W_1, \ldots, W_N$ into the cache contents

$$Z_k \triangleq \phi_k(W_1, \ldots, W_N), \quad \forall k = 1, \ldots, K.$$

The caching functions can be used to cache a few entire files, or a small fraction from each file, or even coded combinations of subfiles [2], [8]. It is important to note that the caching functions are selected once, without knowledge of future requests, and are fixed throughout our system operation.[1]

Downlink channel model. We consider a standard block-fading broadcast channel, such that the channel state remains constant over a slot of T_{slot} channel uses and changes from one slot to another in an i.i.d. manner. The channel output of user k in any channel use of slot t is given by

$$\boldsymbol{y}_k(t) = \sqrt{h_k(t)}\boldsymbol{x}(t) + \boldsymbol{\nu}_k(t), \tag{3}$$

where the channel input $\boldsymbol{x} \in \mathbb{C}^{T_{\text{slot}}}$ is subject to the power constraint $\mathbb{E}[\|\boldsymbol{x}\|^2] \leq PT_{\text{slot}}$; $\boldsymbol{\nu}_k(t) \sim \mathcal{N}_{\mathbb{C}}(0, \mathbf{I}_{T_{\text{slot}}})$ are additive white Gaussian noises with covariance matrix identity of size T_{slot}, assumed independent of each other; $\{h_k(t) \in \mathbb{C}\}$ are channel fading coefficients $\sim \beta_k^2 \text{exp}(1)$ independently distributed across time and users, with β_k denoting the path-loss parameter of user k.

Encoding and transmissions. The transmissions aim to contribute information towards the delivery of a specific vector of file requests $\boldsymbol{d}(t)$, where $d_k(t) \in \{1, \ldots, N\}$ denotes the index of the requested file by user k in slot t. Here N is the video library size, typically in the order of 10K. The requests are generated randomly, and whenever a file is delivered to user k, the next request of this user will be for another randomly selected file.

At each time slot, the base station observes the channel state $\boldsymbol{h}(t) = (h_1(t), \ldots, h_K(t))$ and the request vector up to t, $\boldsymbol{d}^t$, constructs a transmit symbol using the *encoding function* $f_t : \{1, .., N\}^{Kt} \times \mathbb{C}^K \to \mathbb{C}^{T_{\text{slot}}}$.

$$\boldsymbol{x}(t) = f_t\left(\boldsymbol{d}^t, \boldsymbol{h}(t)\right),$$

Finally, it transmits a codeword $\boldsymbol{x}(t)$ for the T_{slot} channel uses over the fading broadcast channel in slot t . The encoding function may be chosen at each slot to contribute information to a selected subset of users $\mathcal{J}(t) \subseteq \{1, \ldots, K\}$. This allows several possibilities, e.g. to send more information to a small set of users with good instantaneous channel qualities, or less information to a large set that includes users with poor quality.

Decoding. At slot t, each user k observes the local cache contents Z_k and the sequence of channel outputs so far $y_k(\tau), \quad \tau = 1, \ldots, t$ and employs a *decoding function* ξ_k to determine the decoded files. Let $D_k(t)$ denote the number of files decoded by user k after t slots. The decoding function ξ_k is a mapping

$$\xi_k : \mathbb{C}^{T_{slot}t} \times \mathbb{F}_2^{FM} \times \mathbb{C}^{Kt} \times \{1, .., N\}^{Kt} \to \mathbb{F}_2^{FD_k(t)}.$$

The decoded files of user k at slot t are given by $\xi_k(y_k^{T_{\text{slot}}t}, Z_k, \boldsymbol{h}^t, \boldsymbol{d}^t)$, and depend on the channel outputs and states up to t, the local cache contents, and the requested files of all users up to t. We say that there is no decoding error at time t if the all decoded files belong to the set of the requested files up to t, i.e $\xi_k(t) \subseteq d_k^t$. A file is incorrectly decoded if it does not belongs to the set of requested files. The number of incorrectly decoded files are then given by $|\cup_t \{\xi_k(t)\} \setminus d_k^t|$ and the number of correctly decoded files at time t is:

$$C_k(t) = D_k(t) - |\cup_t \{\xi_k(t)\} \setminus d_k^t|$$

Definition 1 (Feasible rate)**.** *A rate vector $\overline{\boldsymbol{r}} = (\overline{r}_1, \ldots, \overline{r}_K)$, measured in file/slot, is said to be feasible $\overline{\boldsymbol{r}} \in \Lambda$ if there exist functions $([\phi_k], [f_t], [\xi_k])$ such that:*

$$\overline{r}_k = \limsup_{t \to \infty} \frac{C_k(t)}{t},$$

In contrast to past works which study the performance of one-shot coded caching [2], [6], [8], our rate metric measures the ability of the system to continuously deliver files to users.

C. Code-constrained rate region

Finding the optimal policy is very complex. In this paper, we restrict the problem to specific class of policies given by the following mild assumptions:

Definition 2 (Admissible class policies Π^{CC})**.** *The admissible policies have the following characteristics:*

1) *The caching placement and delivery follow the decentralized scheme [6].*
2) *The users request distinct files, i.e., the ids of the requested files of any two users are different.*

Since we restrict our action space, the delivery rate feasibility region, Λ^{CC}, of the class of policies Π^{CC} is smaller than the one for the original problem Λ. However, these restrictions allow us to come up with a concrete solution approach. Note that the optimal cache and transmission design policy is already a very hard problem even in the simple case of broadcast transmissions with a fixed common rate, and the method in [2], [6] are practical approaches with good performance. In addition, looking at demand IDs when combining files would be very complex and, because of the big library sizes, is not expected to bring substantial gains (it is improbable that two users will make request for the same file in close time instances).

III. Offline Coded Caching

In this section we briefly review decentralized coded caching, first proposed in [6], and used by all admissible policies Π^{CC}. We set $m = \frac{M}{N}$ the normalized memory size. Under the memory constraint of MF bits, each user k independently caches a subset of mF bits of file i, chosen uniformly at random for $i = 1, \ldots, N$. By letting $W_{i|\mathcal{J}}$ denote the sub-file of W_i

[1] A reasonable extension is to enable infrequent updates of the caching placement phase.

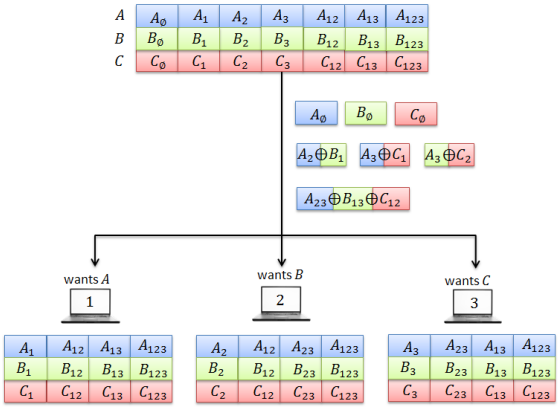


Fig. 2. Decentralized coded caching for $K = 3$

stored exclusively in the cache memories of the user set $\mathcal{J}$, the cache memory Z_k of user k after decentralized placement is given by

$$Z_k = \{W_{i\,|\,\mathcal{J}} : \ \forall \mathcal{J} \subseteq [K], \forall \mathcal{J} \ni k, \forall i = 1, \ldots, N\}. \quad (4)$$

Once the requests of all users are revealed, the offline scheme proceeds to the delivery of the requested files (delivery phase). Assuming that user k requests file k, i.e. $d_k = k$, the server generates and conveys the following codeword simultaneously useful to the subset of users $\mathcal{J}$:

$$V_{\mathcal{J}} = \oplus_{k \in \mathcal{J}} W_{k|\mathcal{J}\setminus\{k\}}, \quad (5)$$

where $\oplus$ denotes the bit-wise XOR operation. The main idea here is to create a codeword useful to a subset of users by exploiting the receiver side information established during the placement phase. It is worth noticing that the *coded* delivery with XORs significantly reduces the number of transmissions. Compared to uncoded delivery, where the sub-files are sent sequentially and the number of transmissions are equal to $|\mathcal{J}| \times |W_{k|\mathcal{J}\setminus\{k\}}|$, the coded delivery requires the transmission of $|W_{k|\mathcal{J}\setminus\{k\}}|$, yielding a reduction of a factor $|\mathcal{J}|$. In a practical case of $N > K$, it has been proved that decentralized coded caching achieves the total number of transmissions, measured in the number of files, given by [6]

$$T_{\text{tot}}(K, m) = \frac{1}{m}(1-m)\left\{1 - (1-m)^K\right\}. \quad (6)$$

Example 1. *For the case of $K = 3$ users in Fig.2, let us assume that user 1, 2, 3, requests file A, B, C, respectively. After the placement phase, a given file A will be partitioned into 8 subfiles. Codewords to be sent are the following*

- *$A_\emptyset$, $B_\emptyset$ and $C_\emptyset$ to user 1, 2 and 3 respectively.*
- *$A_2 \oplus B_1$ is intended to users $\{1, 2\}$. Once received, user 1 decodes A_2 by combining the received codeword with B_1 given in its cache. Similarly user 2 decodes B_1. The same approach holds for codeword $B_3 \oplus C_2$ to users $\{2, 3\}$ and codeword $A_3 \oplus C_1$ to users $\{1, 3\}$*
- *$A_{23} \oplus B_{13} \oplus C_{12}$ is intended users $1, 2, 3$. User 1 can decode A_{23} by combining the received codeword with $\{B_{13}, C_{12}\}$ given in its cache. The same approach is used for user 2, 3 to decode B_{13}, C_{12} respectively.*

IV. Broadcasting Private and Common Messages

In this section, we address the question on how the transmitter shall convey private and multiple common messages, each intended to a subset of users, while opportunistically exploiting the underlying wireless channel. In example 1, $A_\emptyset$, $B_\emptyset$ and $C_\emptyset$ refer to the private messages and the XOR packets refer to common messages. We start by remarking that the channel in (3) for a given channel realization $\boldsymbol{h}$ corresponds to the Gaussian degraded broadcast channel. Without loss of generality, let us assume $h_1 \geq \cdots \geq h_K$ so that the following Markov chain holds.

$$X \leftrightarrow Y_1 \leftrightarrow \cdots \leftrightarrow Y_K.$$

The capacity region of the degraded broadcast channel for K private messages and a common message is well-known [13]. In this section, we consider a more general setup where the transmitter wishes to convey $2^K - 1$ mutually independent messages, denoted by $\{M_{\mathcal{J}}\}$, where $M_{\mathcal{J}}$ denotes the message intended to the users in subset $\mathcal{J} \subseteq \{1, \ldots, K\}$. Each user k must decode all messages $\{M_{\mathcal{J}}\}$ for $\mathcal{J} \ni k$. By letting $R_{\mathcal{J}}$ denote the multicast rate of the message $M_{\mathcal{J}}$, we say that the rate-tuple $\boldsymbol{R} \in \mathbb{R}_+^{2^K-1}$ is achievable if there exists encoding and decoding functions which ensure the reliability and the rate condition. The capacity region is defined as the supremum of the achievable rate-tuple, where the rate is measured in bit/channel use. Then we prove [25] the following result.

Theorem 1. *The capacity region $\Gamma(\boldsymbol{h})$ of a K-user degraded Gaussian broadcast channel with fading gains $h_1 \geq \cdots \geq h_K$ and $2^K - 1$ independent messages $\{M_{\mathcal{J}}\}$ is given by*

$$R_1 \leq \log(1 + h_1 \alpha_1 P) \quad (7)$$

$$\sum_{\mathcal{J} \subseteq \{1,\ldots,k\}: k \in \mathcal{J}} R_{\mathcal{J}} \leq \log \frac{1 + h_k \sum_{j=1}^{k} \alpha_j P}{1 + h_k \sum_{j=1}^{k-1} \alpha_j P} \quad k = 2, \ldots, K \quad (8)$$

for non-negative variables $\{\alpha_k\}$ such that $\sum_{k=1}^{K} \alpha_k \leq 1$.

The achievability builds on superposition coding at the transmitter and successive interference cancellation at receivers. For $K = 3$, the transmit signal is simply given by

$$x = x_1 + x_2 + x_3 + x_{12} + x_{13} + x_{123}$$

where $\{x_{\mathcal{J}}\}$ are mutually independent Gaussian distributed random variables satisfying the power constraint and $x_{\mathcal{J}}$ denotes the signal corresponding to the message $M_{\mathcal{J}}$ intended to the subset $\mathcal{J} \subseteq \{1, 2, 3\}$. User 3 (the weakest user) decodes $\tilde{M}_3 = \{M_3, M_{13}, M_{23}, M_{123}\}$ by treating all the other messages as noise. User 2 decodes first the messages $\tilde{M}_3$ and then jointly decodes $\tilde{M}_2 = \{M_2, M_{12}\}$. Finally, user 1 (the strongest user) successively decodes $\tilde{M}_3, \tilde{M}_2$ and, finally, M_1.

Later in our online coded caching scheme we will need the capacity region $\Gamma(\boldsymbol{h})$, and more specifically, we will need to characterize its boundary. To this end, it suffices to consider the weighted sum rate maximization:

$$\max_{\boldsymbol{r} \in \Gamma(\boldsymbol{h})} \sum_{\mathcal{J}: \mathcal{J} \subseteq \{1,\ldots,K\}} \theta_{\mathcal{J}} r_{\mathcal{J}}. \quad (9)$$

We first simplify the problem using the following theorem.

Theorem 2. *The weighted sum rate maximization with* 2^K-1 *variables in* (9) *reduces to a simpler problem with* K *variables, given by*

$$f(\boldsymbol{\alpha}) = \sum_{k=1}^{K} \tilde{\theta}_k \log \frac{1+h_k \sum_{j=1}^{k} \alpha_j P}{1+h_k \sum_{j=1}^{k-1} \alpha_j P}. \tag{10}$$

where $\tilde{\theta}_k$ *denotes the largest weight for user* k

$$\tilde{\theta}_k = \max_{\mathcal{K}: k \in \mathcal{K} \subseteq \{1,\dots,k\}} \theta_{\mathcal{K}}.$$

Proof. The proof builds on the simple structure of the capacity region. We first remark that for a given power allocation of other users, user k sees 2^{k-1} messages $\{W_{\mathcal{J}}\}$ for all $\mathcal{J}$ such that $k \in \mathcal{J} \subseteq \{1,\dots,k\}$ with the equal channel gain. For a given set of $\{\alpha_j\}_{j=1}^{k-1}$, the capacity region of these messages is a simple hyperplane characterized by 2^{k-1} vertices $C_k \boldsymbol{e}_i$ for $i = 1,\dots,2^{k-1}$, where C_k is the sum rate of user k in the RHS of (8) and $\boldsymbol{e}_i$ is a vector with one for the i-th entry and zero for the others. Therefore, the weighted sum rate seen is maximized for user k by selecting the vertex corresponding to the largest weight, denoted by $\tilde{\theta}$. This holds for any k. □

We provide an efficient algorithm to solve this power allocation problem as a special case of the parallel Gaussian broadcast channel studied in [23, Theorem 3.2]. Following [23], we define the rate utility function for user k given by

$$u_k(z) = \frac{\tilde{\theta}_k}{1/h_k + z} - \lambda \tag{11}$$

where λ is a Lagrangian multiplier. The optimal solution corresponds to selecting the user with the maximum rate utility at each z and the resulting power allocation for user k is

$$\alpha_k^* = \left\{ z : [\max_j u_j(z)]_+ = u_k(z) \right\} / P \tag{12}$$

with λ satisfying

$$P = \left[\max_k \frac{\tilde{\theta}_k}{\lambda} - \frac{1}{h_k} \right]_+ . \tag{13}$$

V. Proposed Online Delivery Scheme

This section presents first the queued delivery network and its feasible rate region of arrival rates, then describes the proposed control policy provided in Algorithm 1. Our delivery network consists of K user queues $\{S_k\}$, 2^K-1 codeword queues $\{Q_{\mathcal{J}}\}$, and K virtual queues $\{U_k\}$.

A. Solution plan

The delivery plase operates after applying the decentralized placement phase [6]. At each time slot t, the controller *admits* $a_k(t)$ files to be delivered to user k, and hence $a_k(t)$ is a control variable.[2] As our model dictates, the succession of requested files for user k is determined uniformly at random.

[2]We note that random file arrivals can be directly captured with the addition of an extra queue [14], which we avoid to simplify exposition.

Algorithm 1 Proposed delivery scheme

PLACEMENT:

$Z_k = \{W_{i\,|\,\mathcal{J}} : \ \forall \mathcal{J} \subseteq [K], \forall \mathcal{J} \ni k, \forall i = 1,\dots,N\}$

DELIVERY:

for $t = 1, 2 \dots$ **do**

$\gamma_k(t) = \arg \max_{0 \le x \le \gamma_{k,max}} [V g_k(x) - U_k(t) x]$

$a_k(t) = \gamma_{k,max} \mathbf{1}\{U_k(t) \ge S_k(t)\}$

$\sigma_{\mathcal{J}}(t) = \sigma_{max} \mathbf{1}\left\{ \sum_{k \in \mathcal{J}} S_k(t) > \sum_{\mathcal{I}: \mathcal{I} \subseteq \mathcal{J}} \frac{b_{\mathcal{J},\mathcal{I}}}{F^2} Q_{\mathcal{I}}(t) \right\}$

$\boldsymbol{\mu}(t) = \arg \max_{\boldsymbol{r} \in \Gamma(\boldsymbol{h}(t))} \sum_{\mathcal{J} \subseteq \{1,\dots,K\}} Q_{\mathcal{J}}(t) r_{\mathcal{J}}$

$U_k(t+1) = [U_k(t) - a_k(t)]^+ + \gamma_k(t)$

$S_k(t+1) = \left[S_k(t) - \sum_{\mathcal{J}: k \in \mathcal{J}} \sigma_{\mathcal{J}}(t)\right]^+ + a_k(t)$

$$Q_{\mathcal{I}}(t+1) = [Q_{\mathcal{I}}(t) - T_{\text{slot}} \mu_{\mathcal{I}}(t)]^+ + \sum_{\mathcal{J}: \mathcal{I} \subseteq \mathcal{J}} b_{\mathcal{J},\mathcal{I}} \sigma_{\mathcal{J}}(t)$$

end for

Queueing model. The base station organizes the information into the following types of queues:

1) **User queues** to store admitted files, one for each user. The buffer size of queue k is denoted by $S_k(t)$ and expressed in number of files.
2) **Codeword queues** to store codewords to be multicast. These codewords are generated by applying coded caching [6] that provides multicasting opportunities. There is one codeword queue for each subset of users $\mathcal{J} \subseteq \{1,\dots,K\}$. The size of codeword queue $\mathcal{J}$ is denoted by $Q_{\mathcal{J}}(t)$ and expressed in bits.

A queueing policy π performs the following operations: (i) decides how many files to admit into the user queues $S_k(t)$ in the form of $(a_k(t))$ variables, (ii) then it decides how to combine together files from different user queues to be encoded into the form of multiple codewords which represent the required broadcast transmissions for the reception of this file–these codewords are stored in the appropriate codeword queues $Q_{\mathcal{J}}(t)$, (iii) and last it decides the encoding function f_t. (ii) and (iii) are further clarified in the next section.

Definition 3 (Stability)**.** *A queue* $S(t)$ *is said to be (strongly) stable if*

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[S(t)\right] < \infty.$$

A queueing system is said to be stable if all its queues are stable. Moreover, the stability region of a system is the set of all arrival rates such that the system is stable.

The above definition implies that the average delay of each job in the queue is finite. In our problem, if we develop a policy that keeps *user queues* $\boldsymbol{S}(t)$ stable, then all admitted files will, at some point, be combined into codewords. If in addition *codeword queues* $\boldsymbol{Q}(t)$ are stable, then all generated codewords will reach their destinations, meaning that all receivers will be able to decode the admitted files that they requested.

Lemma 3. *The region of all feasible delivery rates Λ is the same as the stability region of the system (i.e. the set of all demand arrival rates for which there exists a policy that stabilizes the queueing system).*

Let $\overline{a}_k = \limsup_{t\to\infty} \frac{1}{t}\sum_{t=0}^{t-1} \mathbb{E}\left[a_k(t)\right]$, denote the time average number of admitted files for user k. Lemma 3 implies the following Corollary.

Corollary 4. *Solving* (1) *is equivalent to finding a policy π such that*

$$\overline{\boldsymbol{a}}^{\pi} = \text{argmax} \sum_{k=1}^{K} g_k(\overline{a}_k) \quad (14)$$

$$\text{s.t.} \quad \text{the system is stable.}$$

B. Feasible Region

Contrary to the offline coded caching in [6], we propose an online delivery scheme consisting of the following three blocks. Each block is operated at each slot.

1) **Admission control:** At the beginning of each slot, the controller decides how many requests for each user, $a_k(t)$ should be pulled into the system from the infinite reservoir.
2) **Routing:** The cumulative accepted files for user k are stored in the admitted demand queue whose size is given by $S_k(t)$ for $k = 1, \ldots, K$. The server decides the combinations of files to perform coded caching. The decision at slot t for a subset of users $\mathcal{J} \subseteq \{1, .., K\}$, denoted by $\sigma_{\mathcal{J}}(t) \in \{0, 1, \ldots, \sigma_{\max}\}$, refers to the number of combined requests for this subset of users. It is worth noticing that offline coded caching lets $\sigma_{\mathcal{J}} = 1$ for $\mathcal{J} = \{1, \ldots, K\}$ and zero for all the other subsets. The size of the queue S_k evolves as:

$$S_k(t+1) = \left[S_k(t) - \sum_{\mathcal{J}:k\in\mathcal{J}} \sigma_{\mathcal{J}}(t)\right]^+ + a_k(t) \quad (15)$$

If $\sigma_{\mathcal{J}}(t) > 0$, the server creates codewords by applying offline coded caching explained in Section III for this subset of users as a function of the cache contents $\{Z_j : j \in \mathcal{J}\}$.
3) **Scheduling:** The codewords intended to the subset $\mathcal{J}$ of users are stored in codeword queue whose size is given by $Q_{\mathcal{I}}(t)$ for $\mathcal{I} \subseteq \{1, \ldots, K\}$. Given the instantaneous channel realization $\boldsymbol{h}(t)$ and the queue state $\{Q_{\mathcal{I}}(t)\}$, the server performs scheduling and rate allocation. Namely, at slot t, it determines the number $\mu_{\mathcal{I}}(t)$ of bits per channel use to be transmitted for the users in subset $\mathcal{I}$. By letting $b_{\mathcal{J},\mathcal{I}}$ denote the number of bits generated for codeword queue $\mathcal{I} \subseteq \mathcal{J}$ when offline coded caching is performed to the users in $\mathcal{J}$, codeword queue $\mathcal{I}$ evolves as

$$Q_{\mathcal{I}}(t+1) = \left[Q_{\mathcal{I}}(t) - T_{\text{slot}}\mu_{\mathcal{I}}(t)\right]^+ + \sum_{\mathcal{J}:\mathcal{I}\subseteq\mathcal{J}} b_{\mathcal{J},\mathcal{I}}\sigma_{\mathcal{J}}(t)$$

where $b_{\mathcal{J},\mathcal{I}} = m^{|\mathcal{I}|}(1-m)^{|\mathcal{J}|-|\mathcal{I}|-1}$.

Note that our scheme benefits from coded caching via the codeword queues that contain the XOR packets.

In order determine our proposed policy, namely the set of decisions $\{\boldsymbol{a}(t), \boldsymbol{\sigma}(t), \boldsymbol{\mu}(t)\}$ at each slot t, we first characterize the feasible region Λ as a set of arrival rates $\boldsymbol{a}$. We let $\pi_{\mathbf{h}}$ denote the probability that the channel state at slot t is $\boldsymbol{h} \in \mathcal{H}$ where $\mathcal{H}$ is the set of all possible channel states. We let $\Gamma(\boldsymbol{h})$ denote the capacity region for a fixed channel state $\mathbf{h}$. Then we have the following

Theorem 5 (Feasibility region Λ^{CC})**.** *A demand rate vector is feasible, i.e. $\bar{\boldsymbol{a}} \in \Lambda^{CC}$, if and only if there exist $\boldsymbol{\mu} \in \sum_{\mathbf{h}\in\mathcal{H}} \pi_{\mathbf{h}}\Gamma(\mathbf{h})$, $\bar{\sigma}_{\mathcal{I}} \in [0, \sigma_{max}], \forall \mathcal{I} \subseteq \{1, \ldots, K\}$ such that:*

$$\sum_{\mathcal{J}:k\in\mathcal{J}} \bar{\sigma}_{\mathcal{J}} \geq \bar{a}_k, \forall k = 1, \ldots, K \quad (16)$$

$$T_{slot}\mu_{\mathcal{I}} \geq \sum_{\mathcal{J}:\mathcal{I}\subseteq\mathcal{J}} b_{\mathcal{J},\mathcal{I}}\bar{\sigma}_{\mathcal{J}}, \forall \mathcal{I} \in 2^{\mathcal{K}}. \quad (17)$$

Constraint (16) says that the service rate at which admitted demands are combined to form codewords is greater than the arrival rate, while (17) implies that the long-term average transmission rate $\overline{\mu}_{\mathcal{I}}$ for the subset $\mathcal{I}$ of users should be higher than the rate at which bits of generated codewords for this group arrive. In terms of the queueing system defined, these constraints impose that the service rates of each queue should be greater than their arrival rates, thus rendering them stable.

Theorem 5 implies that the set of feasible average delivery rates is a convex set.

C. Admission Control and Routing

In order to perform the utility maximization (14), we need to introduce one more set of queues. These queues are virtual, in the sense that they do not hold actual file demands or bits, but are merely counters to drive the control policy. Each user k is associated with a queue $U_k(t)$ which evolves as follows:

$$U_k(t+1) = \left[U_k(t) - a_k(t)\right]^+ + \gamma_k(t) \quad (18)$$

where $\gamma_k(t)$ represents the arrival process to the virtual queue and is given by

$$\gamma_k(t) = \arg\max_{0\leq x\leq\gamma_{k,max}} \left[Vg_k(x) - U_k(t)x\right] \quad (19)$$

In the above, $V > 0$ is a parameter that controls the utility-delay tradeoff achieved by the algorithm (see Theorem 6).

The general intuition here is as follows: Observe that the number $a_k(t)$ of admitted demands is the service rate for the virtual queues $U_k(t)$. The control algorithm actually seeks to optimize the time average of the virtual arrivals $\gamma_k(t)$. However, since $U_k(t)$ is stable, its service rate, which is the actual admission rate, will be greater than the rate of the virtual arrivals, therefore giving the same optimizer. Stability of all other queues will guarantee that admitted files will be actually delivered to the users.

We present our on-off policy for admission control and routing. For every user k, admission control chooses $a_k(t)$ demands given by

$$a_k(t) = \gamma_{k,max}\mathbf{1}\{U_k(t) \geq S_k(t)\} \quad (20)$$

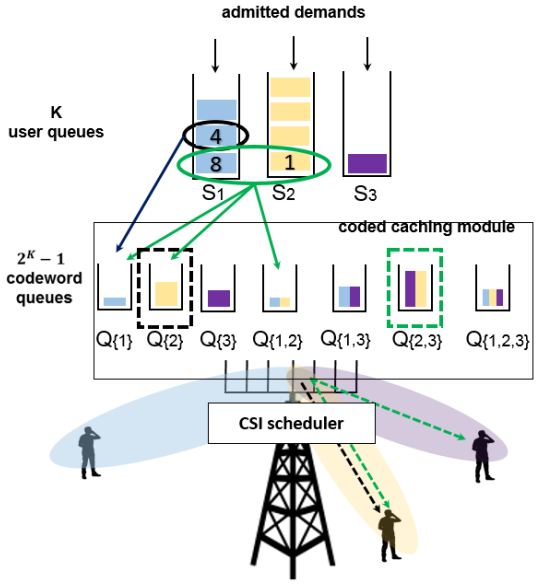


Fig. 3. An example of the queueing model for a system with $K = 3$ users. Dashed lines represent wireless transmissions, solid circles files to be combined and solid arrows codewords generated.

For every subset $\mathcal{J} \subseteq \{1, \dots, K\}$, routing combines $\sigma_{\mathcal{J}}(t)$ demands of users in $\mathcal{J}$ given by

$$\sigma_{\mathcal{J}}(t) = \sigma_{max} \mathbf{1}\left\{ \sum_{k \in \mathcal{J}} S_k(t) > \sum_{\mathcal{I}: \mathcal{I} \subseteq \mathcal{J}} \frac{b_{\mathcal{J},\mathcal{I}}}{F^2} Q_{\mathcal{I}}(t) \right\}. \quad (21)$$

D. Scheduling and Transmission

In order to stabilize all *codeword queues*, the scheduling and resource allocation explicitly solve the following weighted sum rate maximization at each slot t where the weight of the subset $\mathcal{J}$ corresponds to the queue length of $Q_{\mathcal{J}}$

$$\boldsymbol{\mu}(t) = \arg \max_{\boldsymbol{r} \in \Gamma(\boldsymbol{h}(t))} \sum_{\mathcal{J} \subseteq \{1,\dots,K\}} Q_{\mathcal{J}}(t) r_{\mathcal{J}}. \quad (22)$$

We propose to apply the power allocation algorithm in Section IV to solve the above problem by sorting users in a decreasing order of channel gains and treating $Q_{\mathcal{J}}(t)$ as $\theta_{\mathcal{J}}$. In adition, we assume that the number of channel uses in one coherence block is large enough such that the decoding error from choosing channel codes with rate $\boldsymbol{\mu}(t)$ is very small. In this case, no feedback from the receivers is given.

E. Example

We conclude this section by providing an example of our proposed online delivery network for $K = 3$ users as illustrated in Fig. 3. At slot t the server decides to combine W_1 requested by user 2 with W_8 requested by user 2 and to process W_4 requested by user 1 uncoded. Therefore $\sigma_{\{1,2\}}(t) = \sigma_{\{1\}}(t) = 1$ and $\sigma_{\mathcal{J}}(t) = 0$ otherwise. Given this codeword construction, codeword queues have inputs as described in Table I. In addition, data from queues $Q_{\{2\}}(t), Q_{\{2.3\}}(t)$ are transmitted.

VI. Performance Analysis

In thi section, we present the main result of the paper, that our proposed online algorithm leads to close to optimal performance for all policies in the class Π^{CC}:

TABLE I
CODEWORD QUEUES INPUTS.

Queue	Input
$\mathcal{Q}_{\{1\}}$	$W_{8\|\emptyset}$; $W_{8\|\{3\}}$ $W_{4\|\emptyset}$; $W_{4\|\{2\}}$; $W_{4\|\{3\}}$; $W_{4\|\{2,3\}}$
$\mathcal{Q}_{\{2\}}$	$W_{1\|\emptyset}$; $W_{1\|\{3\}}$
$\mathcal{Q}_{\{1,2\}}$	$W_{1\|\{1\}} \oplus W_{8\|\{2\}}$; $W_{1\|\{1,3\}} \oplus W_{8\|\{2,3\}}$

Theorem 6. *Let $\bar{r}_k^{\pi}$ the mean time-average delivery rate for user k achieved by the proposed policy. Then*

$$\sum_{k=1}^{K} g_k(\bar{r}_k^{\pi}) \geq \max_{\bar{\boldsymbol{r}} \in \Lambda^{CC}} \sum_{k=1}^{K} g_k(\bar{r}_k) - O\left(\frac{1}{V}\right)$$

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left\{\hat{Q}(t)\right\} = O(V),$$

where $\hat{Q}(t)$ is the sum of all queue lengths at the beginning of time slot t, thus a measure of the mean delay of file delivery.

The above theorem states that, by tuning the constant V, the utility resulting from our online policy can be arbitrarily close to the optimal one, where there is a tradeoff between the guaranteed optimality gap $O(1/V)$ and the upper bound on the total buffer length $O(V)$.

For proving the Theorem, we use the Lyapunov function

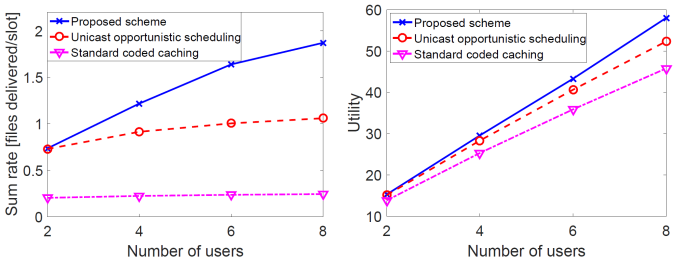
$$L(t) = \frac{1}{2}\left(\sum_{k=1}^{K} U_k^2(t) + S_k^2(t) + \sum_{\mathcal{I} \in 2^{\mathcal{K}}} \frac{1}{F^2} Q_{\mathcal{I}}^2(t) \right)$$

and specifically the related drift-plus-penalty quantity, defined as: $\mathbb{E}\{L(t+1) - L(t) | \mathbf{S}(t), \mathbf{Q}(t), \mathbf{U}(t)\} - V\mathbb{E}\left\{\sum_{k=1}^{K} g(\gamma_k(t)) | \mathbf{S}(t), \mathbf{Q}(t), \mathbf{U}(t)\right\}$. The proposed algorithm is such that it minimizes (a bound on) this quantity. The main idea is to use this fact in order to compare the evolution of the drift-plus-penalty under our policy and two "static" policies, that is policies that take random actions (admissions, demand combinations and wireless transmissions), drawn from a specific distribution, based only on the channel realizations (and knowledge of the channel statistics). We can prove from Theorem 4 that these policies can attain every feasible delivery rate. The first static policy is one such that it achieves the stability of the system for an arrival rate vector $\boldsymbol{a}'$ such that $\boldsymbol{a}' + \delta \in \partial\Lambda^{CC}$. Comparing with our policy, we deduce strong stability of all queues and the bounds on the queue lengths by using a Foster-Lyapunov type of criterion. In order to prove near-optimality, we consider a static policy that admits file requests at rates $\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \sum_k g_k(a_k)$ and keeps the queues stable in a weaker sense (since the arrival rate is now in the boundary Λ^{CC}). By comparing the drift-plus-penalty quantities and using telescopic sums and Jensen's inequality on the time average utilities, we obtain the near-optimality of our proposed policy.

The full proof is in out technical report [25].

VII. Numerical Examples

In this section, we compare our online proposed delivery scheme (section V) with the following two other schemes, all building on decentralized cache placement in (4).



(a) Sum rate ($\alpha = 0$) (b) Proportional fair utility ($\alpha = 1$)

Fig. 4. Performance results vs number of users for $\alpha = 0$ and $\alpha = 1$

Unicast opportunistic scheduling: for any request, the server sends the remaining $(1-m)F$ bits to the corresponding user without combining any files (we only exploit the local caching gain). In slot t the serve sends with full power to user

$$k^*(t) = \arg\max_k \frac{\log\left(1 + h_k(t)P\right)}{T_k(t)^\alpha},$$

where $T_k(t) = \frac{\sum_{1\leq\tau\leq t-1}\mu_k(\tau)}{(t-1)}$ is the empirical average rate for user k up to slot t.

Standard coded caching: we use decentralized coded caching among all K users. For the delivery, non-opportunistic TDMA transmission is used. The server sends sequentially codewords $V_{\mathcal{J}}$ to the subset of users $\mathcal{J}$ at the weakest user rate among $\mathcal{J}$:

$$\mu_{\mathcal{J}}(t) = \log\left(1 + P\min_{k\in\mathcal{J}}(h_k(t))\right).$$

Once the server has sent codewords $\{V_{\mathcal{J}}\}_{\emptyset\neq\mathcal{J}\subseteq\{1,..,K\}}$, every user is able to decode one file. Then the process is repeated for all the demands.

We consider the system with normalized memory of $m = 0.6$, power constraint $P = 10dB$, file size $F = 10^3$ bits and number of channel uses per slot $T_{\text{slot}} = 10^2$. We divide users into two classes of $K/2$ users each: strong users with $\beta_k = 1$ and weak users with $\beta_k = 0.2$. We compare the three algorithms for the cases where the objective of the system is sum rate maximization ($\alpha = 0$) and proportional fairness ($\alpha = 1$). The results are depicted in Fig. 4a and 4b, respectively.

Regarding the sum rate objective, standard coded caching performs very poorly, indicative of the adverse effect of users with bad channel quality. It is notable that our proposed scheme outperforms the unicast opportunistic scheme, which maximizes the sum rate if only private information packets are to be conveyed. The relative merit of our scheme increases as the number of users grows. This can be attributed to the fact that our scheme can exploit any available multicast opportunities. *Our result here implies that, in realistic wireless systems, coded caching can indeed provide a significant throughput increase when an appropriate joint design of routing and opportunistic transmission is used.*

Regarding the proportional fair objective, we can see that the average sum utility increases with a system dimension for three schemes although our proposed scheme provides a gain compared to the two others.

VIII. Conclusions

We studied the content delivery system over the block fading broadcast channel where users may have asymmetric fading statistics. Building on *decentralized coding caching*, we proposed a novel online delivery algorithm to ensure alpha fairness in the long term delivery rates. Our results demonstrate that an appropriate joint design of routing and opportunistic transmission enables to overcome the detrimental effect of naive coded caching limited by the worst channel users and hence significantly increase the system performance.

References

[1] "White paper: Cisco VNI Forecast and Methodology, 2015-2020", Tech. Report, 2015.

[2] M. Maddah-Ali and U. Niesen, "Fundamental Limits of Caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[3] G. S. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: technical misconceptions and business barriers", IEEE Communications Magazine, 2016.

[4] M. Ji, G. Caire, A. Molisch, "Fundamental Limits of Distributed Caching in D2D Wireless Networks" , arXiv/1304.5856, 2013.

[5] M. Ji, A. Tulino, J. Llorca, and G. Caire, "Order-Optimal Rate of Caching and Coded Multicasting with Random Demands", arXiv:1502.03124, 2015.

[6] M. Maddah-Ali and U. Niesen, "Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff", *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1029–1040, 2015.

[7] M. Maddah-Ali and U. Niesen, "Coded Caching with Nonuniform Demands",in *IEEE INFOCOM Workshops*, 2014.

[8] R. Pedarsani, M. Maddah-Ali, and U. Niesen, "Online Coded Caching," in IEEE/ACM Trans. Netw., vol. 24, no. 2, pp. 836-845, 2016.

[9] S. S. Bidokhti, M. Wigger, and R. Timo, "Noisy Broadcast Networks with Receiver Caching", arXiv preprint arXiv:1605.02317, 2016.

[10] J. Zhang, and P. Elia, "Wireless Coded Caching: a Topological Perspective". arXiv:1606.08253, 2016.

[11] K-H. Ngo, S. Yang, and M. Kobayashi, "Cache-Aided Content Delivery in MIMO Channels", in *Proc. Allerton*, IL, USA, 2016.

[12] U. Niesen M. Maddah-Ali "Coded Caching for Delay-Sensitive Content", in IEEE ICC, pp. 5559-5564, 2015.

[13] A. El Gamal and Y. H. Kim, "Network Information Theory", Cambridge university press, 2011.

[14] M. Neely, "Stochastic Network Optimization with Application to Communication and Queueing Systems", Morgan & Claypool, 2010.

[15] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control", IEEE/ACM Trans. Netw. , Vol. 8, No. 5, Oct. 2000.

[16] F. Kelly, "Charging and Rate Control for Elastic Traffic", European Transactions on Telecommunications, 1997.

[17] A. L. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation." Operations Research Vol. 53 No. 1, 2005.

[18] R. Knopp and P. A. Humblet, "Information capacity and power control in single-cell multiuser communications," in IEEE ICC, Seattle, WA, 1995.

[19] H. Shirani-Mehr, G. Caire and M. J. Neely, "MIMO Downlink Scheduling with Non-Perfect Channel State Knowledge," in IEEE Trans. Commun. , vol. 58, no. 7, pp. 2055-2066, July 2010.

[20] G. Caire, R. R. Muller and R. Knopp, "Hard Fairness Versus Proportional Fairness in Wireless Communications: The Single-Cell Case," in IEEE Trans. Inf. Theory, vol. 53, no. 4, pp. 1366-1385, April 2007.

[21] K. Seong, R. Narasimhan, and J. Cioffi, "Queue Proportional Scheduling via Geometric Programming in Fading Broadcast Channels", in IEEE JSAC, vol. 24, no. 8, Aug 2006.

[22] A. Eryilmaz, and R. Srikant, and J. R. Perkins, "Throughput-optimal Scheduling for Broadcast Channels", in Proc. ITCom, Denver, CO, August 2001.

[23] D. Tse, "Optimal Power Allocation over Parallel Gaussian Broadcast Channels", *unpublished*, 1999.

[24] M. Neely, E. Modiano, and C.-P. Li, "Fairness and Optimal Stochastic Control for Heterogeneous Networks", IEEE/ACM Trans. Netw., 2005.

[25] "Alpha Fair Coded Caching: Technical Report", https://www.dropbox.com/s/x2qeyi0bojp79h6/AlphaFairCodedCachingTR.pdf?dl=0, 2017