A novel centralized coded caching scheme with coded prefetching

Jesús Gómez-Vilardebó, *Senior Member, IEEE*

***Abstract*—For the caching problem, when the number of files is no larger than that of users, the best known rate-memory region is achieved by memory sharing between the rate-memory pairs obtained by three schemes: the scheme proposed by Yu et al., the scheme proposed by Gomez-Vilardebo and the scheme proposed by Tian-Chen. While the first two schemes operate on the binary field, the Tian-Chen scheme makes use of a finite field of order 2^m with, in some situations, $m \geq K \log_2(N)$ for a caching systems with K users and N files. The practical implications of this increase in the size of the field are equivalent to an increase, by a factor of m, in the number of subfile partitions required. We propose a novel caching scheme that approaches the rate-memory region achieved by the Tian-Chen scheme as the number of users in the system increases, which only requires a field of order 2^2.**

***Index Terms*—Centralized coded caching, network coding, index coding.**

I. Introduction

Mobile data traffic has been increasing exponentially fast for the last 5-6 years. The advent of multimedia-capable devices such as smartphones and tablets at economical cost, as well as the success of content based applications such as YouTube, Facebook, Netflix, and network gaming, are the main reasons behind this growth. All signs indicate that this trend is likely to continue, for instance, to support HQ video on demand streaming applications. This constantly increasing demand of contents has been guiding the efforts of the industry and research communication communities. One of the most promising techniques proposed so far is "content caching" [1]. The most direct application of the content caching concept is to combat peak hour traffic in content delivery services. The simplest ''uncoded" caching solutions [2], [3] work as follows: during low traffic periods, the storage resources available at the edge of the network are filled out with popular content. Then, whenever a user requests a content that is available at the edge cache, the content is served directly, thus, completely alleviating the backhaul network. This uncoded caching solution is only optimal in single cache systems. However, for multiple caches, the seminal work in [1] showed that important gains can be obtained by a new coded caching strategy. Specifically, in [1] authors show that, besides the local caching gain that is obtained by placing contents at user caches before they are requested, it is possible to obtain a global caching gain by creating broadcast opportunities. This is, by carefully choosing the content caches at different users, and using network coding techniques it is possible to transform the initial unicast network, where every user is requesting a different file, into a broadcast network, where every user requests exactly the same "coded" file, obtaining the new global caching gain.

The fundamental caching scheme developed in [1] was later extended to more realistic situations. The decentralized setting was considered in [4], non-uniforms demands in [5]–[7], online coded caching in [8], and hierarchical cache network were considered in [9], among others. In addition, new schemes pushing further the fundamental limits of caching systems have appeared in [10]–[18]. There have been also efforts to obtain theoretical lower bounds on the delivery rate. The cut-set bound was studied in [1]. A tighter lower bound was obtained in [19]. Through a computational approach a lower-bound for the special case with 3 files and 3 users is derived in [20]. Other lower bounds have appeared in [12], [21], [22].

Perhaps the most important challenge that needs further attention before coded caching solutions can be incorporated into real systems is what is known as the subpacketization level [23]–[25]. The theoretical gains of coded caching are demonstrated with strategies that require dividing each cached file into a number of parts that grows exponentially with the number of caches in the system. As an example, for the particular case where users store half of the full server data base, the centralized strategy in [1] which constitutes a fundamental building block for any of the current known extensions, needs to divide each packet into 10^{14} subfiles with only 50 cache-users. The practical implications of such subpacketizaion levels are important in terms of the overall system complexity, signaling overhead and delay [26].

A particularly interesting situation is when there are more users K than files N. It was shown in [7] that a near optimal caching strategy consists in dividing the files into groups with similar popularity, and then applying the coded caching strategy to each group separately. Since the amount of users in each groups remains the same, when there are many groups, the cache size dedicated to each group is small as well as the number of files per user in each group.

If there are more users than files and cache memories are small, some of the best rate-memory pairs known are achieved by the Tian-Chen coding scheme [16]. This coding scheme, however, makes use, in some situations, of a finite field of at least order N^K. The practical implications of this increase in the size of the field are equivalent to an increase, by a factor of $K \log_2 N$, in the number of subfile partitions required compared to the scheme in [1]. We propose a novel caching

Manuscript received ; revised ; accepted; approved by IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Editor . This work was supported in part by the Catalan Government under Grant SGR2017-1479, and in part by the Spanish Government under Grant TEC2013-44591-P (INTENSYV). The author is with the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), 08860 Barcelona, Spain (e-mail: jesus.gomez@cttc.cat).

scheme that approaches the rate-memory region achieved by the Tian-Chen scheme as the number of users in the system increases, which only requires a finite field of order 2^2. Moreover, instead of relying on the existence of some valid code as in [17], we provide an explicit combinatorial construction of the caching scheme, including, both, the prefetching, and delivery (broadcast and decoding) phases.

The rest of this paper is organized as follows. In Section II, we present the system model together with the more relevant previous results. In Section III, we summarize the main results. Section IV describes the caching scheme proposed. A detailed example is developed in Section V. Finally, conclusions are drawn in Section VI.

II. System Model and Previous Results

We consider a communication system with one server connected to K users, denoted as $U_1, ..., U_K$, through a shared, error-free link. There is a database at the server with N files, each of length F bits, denoted as $W_1, ..., W_N$. Each users is equipped with a local cache of capacity MF bits, $M \in [0, N]$, and is assumed to request only one full file. Here, we consider the special case where there are more users than files $N \leq K$.

We consider the communication model introduced in [1]. The caching system operates in two phases: the *placement phase* and the *delivery phase*. In the placement phase, users have access to the server database, and each user fills their cache. We allow coding in the prefetching phase. Thus, at user U_k, the caching function maps the database $\{W_1, ..., W_N\}$ to the cache content $\mathcal{M}_k$. Then, each user U_k requests a single full file $W_{\mathbf{d}(k)}$, where $\mathbf{d} = (\mathbf{d}(1), ..., \mathbf{d}(K))$ denotes the demand vector. We denote the number of distinct requests in $\mathbf{d}$ as $N_e(\mathbf{d})$. In the delivery phase, only the server has access to the database. After being informed of the user demands, the server transmits a signal Y of size RF bits over the shared link to satisfy all user requests simultaneously. The signal Y is a function of the demand vector $\mathbf{d}$, all the files in the data base $W_1, ..., W_N$, and the content in the user caches $\boldsymbol{\mathcal{M}} = \{\mathcal{M}_1, ..., \mathcal{M}_k\}$. Using the local cache content and the received signal Y, each user U_k reconstructs its requested file $W_{\mathbf{d}(k)}$.

Let $\mathcal{D} = \{1, ..., N\}^K$, for a caching system (M, N, K), given a particular prefetching $\boldsymbol{\mathcal{M}} = \{\mathcal{M}_1, ..., \mathcal{M}_k\}$ and a particular demand $\mathbf{d}$, we say that communication rate R is achievable if and only if there exists a message Y of length RF bits such that every user U_k is able to reconstruct its desired file $W_{\mathbf{d}(k)}$. For a particular prefetching $\boldsymbol{\mathcal{M}}$ and demand $\mathbf{d}$, we denote the achievable rate as $R(\mathbf{d}, \mathcal{M})$. Then, the rate needed for the worst demand is given by $R^*(\boldsymbol{\mathcal{M}}) = \max_{\mathbf{d}\in\mathcal{D}} R(\mathbf{d}, \boldsymbol{\mathcal{M}})$. Finally, we define the rate-memory pair (R^*, M), or the rate-memory trade-off function $R^*(M)$ as the minim rate R^* for different memory constraints M, i.e. we aim to find

$$R^* = \min_{\boldsymbol{\mathcal{M}}} R^*(\boldsymbol{\mathcal{M}})$$

where the minimization is over all caching schemes $\boldsymbol{\mathcal{M}}$ satisfying the memory load constrain M. Observe, that if users have no caching capacity $M = 0$, the server needs to send the full requested files and thus, the worst demand rate is $R^* = N$. Instead, if users can have a complete copy of the server's database $M = N$, then no information needs to be transmitted from the server $R^* = 0$.

As far as we known, for the centralized caching setting, the best known explicitly characterization of the rate-memory trade-off can be obtained by memory sharing between three rate-memory pair families: the rate-memory pairs presented in [27, Corollary 1.1],

$$(R^*_{\text{GV}}, M_{\text{GV}}) = \left(N - \frac{N}{K}\frac{N+1}{q+1}, \frac{N}{Kq}\right) \tag{1}$$

for $q \in \{1, ..., N\}$; the rate-memory pairs obtained in [15, Corollary 1],

$$(R^*_{\text{YU}}, M_{\text{YU}}) = \left(\frac{\binom{K}{r+1} - \binom{K-\min\{K,N\}}{r+1}}{\binom{K}{r}}, \frac{rN}{K}\right) \tag{2}$$

for $r \in \{0, 1, ..., K\}$, and the rate-memory pairs obtained in [16, Theorem 1]

$$(R^*_{\text{TC}}, M_{\text{TC}}) = \left(N\left(1 - \frac{r}{K}\right), \frac{r}{K} + (N-1)\frac{r}{K}\frac{r-1}{K-1}\right), \tag{3}$$

with $r \in \{0, 1, ..., K\}$.

The rate-memory pairs in (2) include as special case the rate-memory pair found in [10] for $M = \frac{N}{K}$. In addition, both (1) and (3) include as special case the rate-memory pair $(R^*, M) = \left(N - \frac{N}{K}, \frac{1}{K}\right)$ obtained in [14]. Recently, a method to obtain new rate-memory pairs has been developed in [17]. However, no explicit characterization of the achievable rate-memory pairs is provided. The optimal rate-memory trade-off for a caching systems remains an open problem. There have been efforts also to obtain theoretical lower bounds on the delivery rate. The cut-set bound was studied in [1]. A tighter lower bound was obtained in [19]. Through a computational approach a lower-bound for the special case $N = K = 3$ is derived in [20]. Other lower bounds have appeared in [12], [21], [22].

Yu et al. [15] and Tian-Chen [16] coded caching schemes, both define exactly the same subfile partitioning and assignment strategies. Each file is divided into $s = \binom{K}{r}$ subfiles, out of which to every user $s_a = \binom{K-1}{r-1}$ subfiles are assigned in a way that each subfile is assigned to r users. Since there are N files, the total number of subfiles assigned to a user is Ns_a. Let $s_{\bar{a}} = s - s_a$ denote the number subfiles of every file not assigned to a user. The prefetching and delivery strategies are however quite different for both schemes. In Yu et al. uncoded prefetching is proposed. At each user, the subfiles assigned are directly cached without further encoding. The number of subfiles cached is thus $c_{\text{YU}} = s_a$. The objective of the delivery strategy is to efficiently transmit the $s_{\bar{a}}$ requested subfiles missing at each user. Observe that a "greedy" uncoded transmission strategy could be to simply broadcast the $s_{\bar{a}}$ requested subfiles missing at each user. This "greedy" strategy requires a total of $Ks_{\bar{a}}$ broadcasted subfiles. The Yu et al. broadcasting strategy finds multicast opportunities that allows a set of users to simultaneously recover their requested subfiles, as a result at most $b_{\text{YU}} = \binom{K}{r+1} = \frac{K}{r+1}s_{\bar{a}}$ coded broadcasted subfiles are required. The Tian-Chen scheme, instead, make use of coded

prefetching. At each user, coded cached subfiles are obtained as linearly independent combinations, in some finite field, of the s_a subfiles assigned to the user. The total number of coded cached subfiles at each user is

$$c_{\text{TC}} = (N-1)\binom{K-2}{r-2} + \binom{K-1}{r-1}.$$

For any coded prefetching solution, the delivery strategy needs to, not only deliver the $s_{\bar{a}}$ requested subfiles missing at each user, but also to allow decoding the s_a requested subfiles encoded in the cache. The Tian-Chen delivery scheme achieves these two objectives but, additionally, requires each user to completely decode the Ns_a subfiles encoded in its cache[1]. Observe that a "greedy" delivery strategy in this case, could be to broadcast to each user the $s_{\bar{a}}$ missing requested subfiles uncoded, together with $Ns_a - c_{\text{TC}}$ linearly independent combination of the subfiles coded in the cache, that together with the c_{TC} coded subfiles already in the cache could allow the decoding of every subfile in the cache. For this greedy scheme, the required total number of brodcasted subfiles is

$$b_{GC} = K\left(s_{\bar{a}} + Ns_a - c_{TC}\right) = K\left(r\frac{N-1}{K-1} + 1\right)s_{\bar{a}}.$$

Instead, the Tian-Chen coded caching strategy is able to solve the same problem, broadcasting only $b_{TC} = Ns_{\bar{a}}$ coded subfiles. To obtain this result, the authors in [16], [17] employed codes, in particular maximum distant separable (MDS) codes and/or rank metric codes, operating in a finite field of at least order

$$\min\left(N^K, 2\binom{K-1}{r-1}N - \binom{K-2}{r-1}(N-1)\right). \quad (4)$$

This is in contrasts to the combinatorial description of schemes presented in [15] and [27], which only require binary operations.

III. Main Result

The following theorem presents the delivery rate obtained by the proposed caching scheme for a particular demand $\mathbf{d}$.

Theorem 1. *For a caching problem with K users, N files, and local cache size MF bits at each user. Given a particular demand $\mathbf{d}$, let $N_e(\mathbf{d})$ be the number of distinct file requests, the proposed strategy achieves the rate-memory pairs*

$$R_{TC^+} = \min\left(N_e(\mathbf{d})+1, N\right)\left(1 - \frac{r}{K+1}\right)$$
$$M_{TC^+} = \frac{r}{K+1} + (N-1)\frac{r}{K+1}\frac{r-1}{K}$$

for $r \in \{0, ..., K+1\}$. Moreover, the new coded caching strategy operates in a finite field $\mathbb{F}_4$.

We prove this result in the following section by describing the new caching scheme. The delivery rate presented in Theorem 1 is valid for any K and N. However, it is particularly useful for $K \geq N$, as we detail in the next corollaries.

[1]This additional condition is probably limiting the performance of the strategy, however is a key ingredient in their result

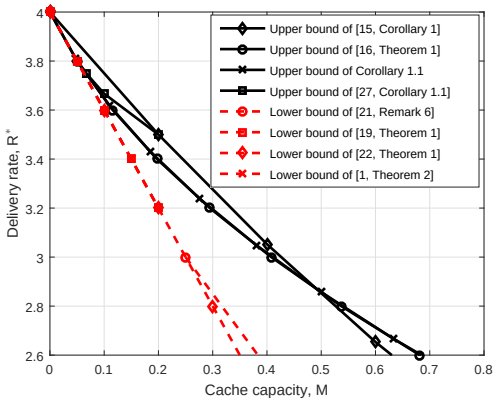

Figure 1: Rate-memory functions of the proposed scheme compared with existing schemes and lower bounds in the literature for $N = 4$ and $K = 20$.

The next corollary provides the rate-memory region for the worst demand as a direct consequence of Theorem 1.

Corollary 1.1. *For a caching problem with K users and N files, local cache size of MF bits at each user, given that $N_e(\mathbf{d}) \leq K$, the proposed strategy achieves the worst demand rate-memory pairs $(R^*, M) = (R^*_{TC^+}, M_{TC^+})$, with*

$$R^*_{TC^+} = \min\left(N, K+1\right)\left(1 - \frac{r}{K+1}\right)$$
$$M_{TC^+} = \frac{r}{K+1} + (N-1)\frac{r}{K+1}\frac{r-1}{K}$$

for $r \in \{0, ..., K+1\}$.

Remark 1.1. *Observe that, if $K \geq N$ the worst demand rate-memory function in Corollary 1.1 for K users coincides with the Tian-Chen rate-memory pair in (3) for $K+1$ users.*

The following corollary compares both rate-memory functions. The proof is provided in the Appendix.

Corollary 1.2. *Let $R^*_{TC^+}(M)$ be the worst demand rate-memory function achieved by memory sharing between the rate-memory pairs in Corollary 1.1, and let $R^*_{TC}(M)$ be the worst demand rate-memory function of the Tian-Chen scheme. For any number of files N and users K, satisfying $K \geq N$ and $M \in [0, N]$, we have*

$$\frac{R^*_{TC^+}(M)}{R^*_{TC}(M)} \leq 1 + \frac{1}{(K-1)(K+1)}.$$

Moreover, we have $R^*_{TC^+}(M) = R^*_{TC}(M)$ at

$$M = \frac{r+1}{K+1}\left(1 + (N-1)\frac{r}{K}\right)$$

for all $r = \{0, ..., K\}$, as well as, at the intervals

$$0 \leq M \leq \frac{1}{K+1}, \frac{K-1}{K+1}N + \frac{1}{K+1} \leq M \leq N.$$

Observe that the proposed scheme matches Tian-Chen result for very low and very high caching ratios. For a system with $N = 2$ files and $K = 3$ users, Fig. 2 in the appendix, shows

the regimes where the proposed scheme and the Tian-Chen scheme are equal and those where they differ. In Fig. 1 for a caching system with $N = 4$ files and $K = 20$ users, we compare the rate-memory pairs in Corollary 1.1 to the best known rate-memory functions in the literature, i.e. the rate-memory pairs in [15, Corollary 1], [27, Corollary 1.1] and [16, Theorem 1]. In addition, we depict for comparison, the cut set lower bound [1, Theorem 2], the information-theoretical lower bound obtained in [19, Theorem 1], and the lower bounds recently appeared in [22, Theorem 1] and [21, Remark 6]. Observe that, as expected from the result in Corollary 1.2, the new proposed scheme obtains a rate-memory function overlapping almost exactly the rate-memory function in [16], [17]. Moreover, the strategy presented here only requires a finite field of order 2^2. Instead, the direct evaluation of (4), for the rate-memory pair associated to $r = 11$, shows us that the Tian-Chen scheme requires a finite field of at least size 10^5.

IV. Proposed caching scheme

We prove Theorem 1 in the following subsections. First, we introduce the main ideas of the new scheme with three examples. Then, we describe a caching scheme that achieves the Tian-Chen rate-memory pairs only if all files are requested by some user. This scheme fails to satisfy other type of demands where some files are not requested. Next, we show how the same scheme can satisfy any demand if configured for $K+1$ caches instead of K. To help following every step of the description of the general scheme, we provide and extended example in the next section.

A. Motivation Examples

Here we present three examples. The first example considers the case when all files are requested but only by one user. The second example considers the case when not all files are requested. Finally, the third example considers the case when some files are requested by more than one user.

Example 1: Consider a caching system with $K = 4$ users U_1,...,U_4 and $N = 4$ files A, B, C, D. Let us fix $r = 3$, then each file is partitioned into $\binom{4}{3} = 4$ subfiles, and each subfile is assigned to $r = 3$ users. To make explicit the users to which each subfile is assigned, we denote subfiles as $\mathrm{A}_{1,2,3}$, $\mathrm{A}_{1,2,4}$, $\mathrm{A}_{1,3,4}$, and $\mathrm{A}_{2,3,4}$. Then, the subfiles assigned to user U_1 are

$$\mathrm{A}_{1,2,3}, \mathrm{A}_{1,2,4}, \mathrm{A}_{1,3,4}, \mathrm{B}_{1,2,3}, \mathrm{B}_{1,2,4}, \mathrm{B}_{1,3,4}$$
$$\mathrm{C}_{1,2,3}, \mathrm{C}_{1,2,4}, \mathrm{C}_{1,3,4}, \mathrm{D}_{1,2,3}, \mathrm{D}_{1,2,4}, \mathrm{D}_{1,3,4}$$

For the delivery strategy, we consider the uncoded transmission of the requested subfiles not encoded in the cache. For instance, if users U_1, U_2, U_3, U_4 request A, B, C, D, respectively. Then, the server broadcasts

$$\mathrm{A}_{2,3,4}, \mathrm{B}_{1,3,4}, \mathrm{C}_{1,2,4}, \mathrm{D}_{1,2,3}.$$

Observe that, in general, there are $Ks_{\bar{a}}$ of these subfiles. Observe also that, since $K = N$, this coincides with the the total number of broadcasted subfiles for the Tian-Chen schemes, i.e. $b_{\mathrm{TC}} = Ns_{\bar{a}}$. To design the prefetching scheme, we impose, as in the Tian-Chen scheme, the decoding of all the subfiles in the cache for any possible demand. For user U_1, observe that, for each of the files B, C, D not requested by user U_1, hereafter interfering files, the server broadcasts one subfile coded in the cache of user U_1, i.e. $\mathrm{B}_{1,3,4}, \mathrm{C}_{1,2,4}, \mathrm{D}_{1,2,3}$. Taking this into account, we design a set of coded subfiles only XORing subfiles of one file, such that after receiving any subfile of this file, the rest can be decoded. For this particular examples, for file B, we could fill the cache at user U_1 with

$$\mathrm{B}_{1,2,3} \oplus \mathrm{B}_{1,2,4}, \mathrm{B}_{1,2,3} \oplus \mathrm{B}_{1,3,4}.$$

Observe that after receiving $\mathrm{B}_{1,3,4}$, user U_1 can obtain all the subfiles of file B coded in his cache. For symmetry, the equivalent coded subfiles are cached for other other three files.

We still need to obtain the requested subfiles encoded in the cache. However, given that all interfering subfiles are already known, any set of coded subfiles, each XORing together subfiles of all N files, and satisfying that each subfile is XORed only once, will work. For instance, we can use

$$\mathrm{A}_{1,2,3} \oplus \mathrm{B}_{1,2,3} \oplus \mathrm{C}_{1,2,3} \oplus \mathrm{D}_{1,2,3}$$
$$\mathrm{A}_{1,2,4} \oplus \mathrm{B}_{1,2,4} \oplus \mathrm{C}_{1,2,4} \oplus \mathrm{D}_{1,2,4}$$
$$\mathrm{A}_{1,3,4} \oplus \mathrm{B}_{1,3,4} \oplus \mathrm{C}_{1,3,4} \oplus \mathrm{D}_{1,3,5}$$

Observe that we only needed binary operations.

Example 2: Next, yet assuming that each files is requested by one user at most, we address the situation when not all files are requested. Consider a caching system with $K = 3$ and $N = 4$ files. Observe that if we assume that there is a 4th "virtual" user that always requests the file not requested by the others, then we can use the coded caching strategy presented in Example 1 for $K = 4$ users. In general, for a system with K users, we can achieve the Tian-Chen rate-memory curve for $K + 1$ users.

Example 3: Next, we address the situation where some files are requested by more than one user. For this case, the binary operation field would not be sufficient. We focus on a caching system with $N = 2$ files A, B, $K = 3$ users, and $r = 2$. By applying the coded prefetching scheme introduced in Example 1, we obtain

U_1	U_2	U_3
$\mathrm{A}_{1,2} \oplus \mathrm{B}_{1,2}$	$\mathrm{A}_{1,2} \oplus \mathrm{B}_{1,2}$	$\mathrm{A}_{1,3} \oplus \mathrm{B}_{1,3}$
$\mathrm{A}_{1,3} \oplus \mathrm{B}_{1,3}$	$\mathrm{A}_{2,3} \oplus \mathrm{B}_{2,3}$	$\mathrm{A}_{2,3} \oplus \mathrm{B}_{2,3}$
$\mathrm{A}_{1,2} \oplus \mathrm{A}_{1,3}$	$\mathrm{A}_{1,2} \oplus \mathrm{A}_{2,3}$	$\mathrm{A}_{1,3} \oplus \mathrm{A}_{2,3}$
$\mathrm{B}_{1,2} \oplus \mathrm{B}_{1,3}$	$\mathrm{B}_{1,2} \oplus \mathrm{B}_{2,3}$	$\mathrm{B}_{1,3} \oplus \mathrm{B}_{2,3}$

Suppose file A is requested by users U_1 and U_3, whereas file B is requested by user U_2. Then, the uncoded broadcasting strategy used for Example 1 transmits all the requested subfiles not encoded at the user requesting them, i.e. $\mathrm{A}_{1,2}, \mathrm{A}_{2,3}, \mathrm{B}_{1,3}$. Notice that these subfiles are sufficient for every user to obtain the requested subfiles and decode all the subfiles in the cache. However, since there are more users than files $K > N$, the uncoded broadcasted scheme needs $Ks_{\bar{a}}$ subfiles transmission instead of the $Ns_{\bar{a}}$ required by the Tian-Chen scheme. To see the limitations of the uncoded broadcasted scheme, observe that upon recovering all the subfiles in the cache user U_1 has $\mathrm{A}_{1,2}$ and user U_2 has $\mathrm{A}_{2,3}$ and thus, broadcasting $\mathrm{A}_{1,2} \oplus \mathrm{A}_{2,3}$ would be sufficient for both to recover the missing subfiles. However, the coded subfile $\mathrm{A}_{1,2} \oplus \mathrm{A}_{2,3}$

is already at user U_2 cache and is, thus, useless to him. To resolve this issue, we further divide each subfiles into two half-subfiles, $\mathrm{A}_{1,2} = \left\langle \mathrm{A}^I_{1,2}, \mathrm{A}^Q_{1,2} \right\rangle$ and consider the previous prefetching scheme over the half-subfiles. Then, the cache of users U_1 and U_2 read

U_1		U_2	
$\mathrm{A}^I_{1,2} \oplus \mathrm{B}^I_{1,2}$	$\mathrm{A}^Q_{1,2} \oplus \mathrm{B}^Q_{1,2}$	$\mathrm{A}^I_{1,2} \oplus \mathrm{B}^I_{1,2}$	$\mathrm{A}^Q_{1,2} \oplus \mathrm{B}^Q_{1,2}$
$\mathrm{A}^I_{1,3} \oplus \mathrm{B}^I_{1,3}$	$\mathrm{A}^Q_{1,3} \oplus \mathrm{B}^Q_{1,3}$	$\mathrm{A}^I_{2,3} \oplus \mathrm{B}^I_{2,3}$	$\mathrm{A}^Q_{2,3} \oplus \mathrm{B}^Q_{2,3}$
$\mathrm{A}^I_{1,2} \oplus \mathrm{A}^I_{1,3}$	$\mathrm{A}^Q_{1,2} \oplus \mathrm{A}^Q_{1,3}$	$\mathrm{A}^I_{1,2} \oplus \mathrm{A}^I_{2,3}$	$\mathrm{A}^Q_{1,2} \oplus \mathrm{A}^Q_{2,3}$
$\mathrm{B}^I_{1,2} \oplus \mathrm{B}^I_{1,3}$	$\mathrm{B}^Q_{1,2} \oplus \mathrm{B}^Q_{1,3}$	$\mathrm{B}^I_{1,2} \oplus \mathrm{B}^I_{2,3}$	$\mathrm{B}^Q_{1,2} \oplus \mathrm{B}^Q_{2,3}$

Now, observe that broadcasting the half-subfiles $\mathrm{A}^I_{1,2} \oplus \mathrm{A}^I_{2,3} \oplus \mathrm{A}^Q_{2,3}$, $\mathrm{A}^I_{1,2} \oplus \mathrm{A}^Q_{1,2} \oplus \mathrm{A}^Q_{2,3}$, $\mathrm{B}^I_{1,3}$, and $\mathrm{B}^Q_{1,3}$, allows every user to recover their requested subfiles. Observe that U_1 and U_3 upon receiving $\mathrm{B}^I_{1,3}$, and $\mathrm{B}^Q_{1,3}$ can cancel all the interference from file B in their cache. Consequently, U_1 also obtains $\mathrm{A}^I_{1,2}$ and $\mathrm{A}^Q_{1,2}$, which finally allows him to obtain first $\mathrm{A}^Q_{2,3}$ from $\mathrm{A}^I_{1,2} \oplus \mathrm{A}^Q_{1,2} \oplus \mathrm{A}^Q_{2,3}$ and then $\mathrm{A}^I_{2,3}$ from $\mathrm{A}^I_{1,2} \oplus \mathrm{A}^I_{2,3} \oplus \mathrm{A}^Q_{2,3}$. U_3 proceeds similarly. Observe that U_2 can also cancel the interference of file A using the broadcasted half-subfiles and the half-subfiles in their cache, by first computing

$$\begin{aligned}\left[\mathrm{A}^I_{1,2} \oplus \mathrm{A}^I_{2,3}\right] \oplus \left[\mathrm{A}^I_{1,2} \oplus \mathrm{A}^I_{2,3} \oplus \mathrm{A}^Q_{2,3}\right] &= \mathrm{A}^Q_{2,3} \\ \left[\mathrm{A}^Q_{1,2} \oplus \mathrm{A}^Q_{2,3}\right] \oplus \left[\mathrm{A}^I_{1,2} \oplus \mathrm{A}^Q_{1,2} \oplus \mathrm{A}^Q_{2,3}\right] &= \mathrm{A}^I_{1,2}\end{aligned}$$

and then

$$\begin{aligned}\left[\mathrm{A}^Q_{1,2} \oplus \mathrm{A}^Q_{2,3}\right] \oplus \mathrm{A}^Q_{2,3} &= \mathrm{A}^Q_{1,2} \\ \left[\mathrm{A}^I_{1,2} \oplus \mathrm{A}^I_{2,3}\right] \oplus \mathrm{A}^I_{1,2} &= \mathrm{A}^I_{2,3}.\end{aligned}$$

Finally, after removing all interference from file A, U_2 obtains the requested subfiles $\mathrm{B}^I_{1,2}$, $\mathrm{B}^Q_{1,2}$, $\mathrm{B}^I_{2,3}$, $\mathrm{B}^Q_{2,3}$ coded in its cache.

B. Coded caching scheme if all files are requested

Let us first consider a caching system that can only serve demand vectors that include all files in the library.

1) Prefetching scheme: Let us define the set of users indexes as $\mathcal{K} = \{1, ..., K\}$ and the set of file indexes as $\mathcal{F} = \{1, ..., N\}$. We partition each file W_f, $f \in \mathcal{F}$ into $\binom{K}{r}$ non-overlaping subfiles of equal size[2], $W_{f,\mathcal{S}}$, one for each subset $\mathcal{S}$ of $r \in \{0, ..., K\}$ users, i.e. $\mathcal{S} \in \mathcal{T}(r)$ with

$$\mathcal{T}(r) = \{\mathcal{S} \subseteq \mathcal{K} : |\mathcal{S}| = r\}.$$

Let $\mathcal{T}_k(r) = \{\mathcal{S} \in \mathcal{T}(r) : k \in \mathcal{S}\}$ be the set of subsets of r users that include a particular user U_k. User U_k caches the binary sum of the N subfiles associated to the same subset $\mathcal{S}$

$$Z_{\mathcal{S}} = \bigoplus_{f \in \mathcal{F}} W_{f,\mathcal{S}} \tag{5}$$

for all $\mathcal{S} \in \mathcal{T}_k(r)$. Consequently, each user caches $\binom{K-1}{r-1}$ coded subfiles. If $r = 1$ these are all the coded cached subfiles need at the caches. If $r \geq 2$, we need more coded cached subfiles. Specifically, we need user U_k to be able to obtain from its cache the coded subfiles

$$Z_{f,\mathcal{S}^-} = \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} W_{f,\mathcal{S}^- \cup s} \tag{6}$$

for every file $f \in \mathcal{F}$, and every subset $\mathcal{S}^- \in \mathcal{T}_k(r-1)$. As we show next, user U_k does not need to cache them all. Specifically, user U_k arbitrarily selects one file index g, and one user index l, and caches only the coded subfiles $Z_{f,\mathcal{S}^-}$ in (6) for all files $f \in \mathcal{F} \backslash g$ and all sets $\mathcal{S}^- \in \mathcal{T}_k(r-1)$ satisfying $l \notin \mathcal{S}^-$. The total number of coded subfiles $Z_{f,\mathcal{S}^-}$ cached at user U_k is then $(N-1)\binom{K-2}{r-2}$.

Then, whenever user U_k needs $Z_{f,\mathcal{S}^-}$ for any file $f \in \mathcal{F} \backslash g$ and any subset $\mathcal{S}^- \in \mathcal{T}_k(r-1)$ satisfying $l \in \mathcal{S}^-$, he only needs to XOR the coded subfiles $Z_{f,\{\mathcal{S}^- \backslash l\} \cup v}$ for all $v \in \mathcal{K} \backslash \mathcal{S}^-$. Observe that since $l \notin \{\mathcal{S}^- \backslash l\} \cup v$, this coded subfiles are cached at user U_k. Then, $Z_{f,\mathcal{S}^-}$

$$\begin{aligned}&= \bigoplus_{v \in \mathcal{K} \backslash \mathcal{S}^-} Z_{f,\{\mathcal{S}^- \backslash l\} \cup v} \\ &= \bigoplus_{v \in \mathcal{K} \backslash \mathcal{S}^-} \bigoplus_{s \in \mathcal{K} \backslash \{\{\mathcal{S}^- \backslash l\} \cup v\}} W_{f,\{\mathcal{S}^- \backslash l\} \cup v \cup s} \\ &= \bigoplus_{v \in \mathcal{K} \backslash \mathcal{S}^-} W_{f,\mathcal{S}^- \cup v} \oplus \bigoplus_{v \in \mathcal{K} \backslash \mathcal{S}^-} \bigoplus_{s \in \mathcal{K} \backslash \{\mathcal{S}^- \cup v\}} W_{f,\{\mathcal{S}^- \backslash l\} \cup v \cup s} \\ &= \bigoplus_{v \in \mathcal{K} \backslash \mathcal{S}^-} W_{f,\mathcal{S}^- \cup v} \qquad (7)\end{aligned}$$

where equality (7) follows since in the previous equality $W_{f,\{\mathcal{S}^- \backslash l\} \cup v \cup s}$ appear twice, and thus cancel out.

To obtain $Z_{g,\mathcal{S}^-}$ for any set $\mathcal{S}^- \in \mathcal{T}_k(r-1)$, user U_k XORs the coded cached subfiles $Z_{\mathcal{S}^- \cup s}$ for all $\mathcal{K} \backslash \mathcal{S}^-$ and the coded cached subfiles $Z_{f,\mathcal{S}^-}$ for all $f \in \mathcal{F} \backslash g$, as follows

$$\begin{aligned}Z_{g,\mathcal{S}^-} &= \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} Z_{\mathcal{S}^- \cup s} \oplus \bigoplus_{f \in \mathcal{F} \backslash g} Z_{f,\mathcal{S}^-} \\ &= \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} Z_{\mathcal{S}^- \cup s} \oplus \bigoplus_{f \in \mathcal{F}} Z_{f,\mathcal{S}^-} \oplus Z_{g,\mathcal{S}^-} \\ &= \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} Z_{\mathcal{S}^- \cup s} \oplus \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} Z_{\mathcal{S}^- \cup s} \oplus Z_{g,\mathcal{S}^-} \qquad (8)\end{aligned}$$

where (8) follows, since

$$\begin{aligned}\bigoplus_{f \in \mathcal{F}} Z_{f,\mathcal{S}^-} &= \bigoplus_{f \in \mathcal{F}} \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} W_{f,\mathcal{S}^- \cup s} \\ &= \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} \bigoplus_{f \in \mathcal{F}} W_{f,\mathcal{S}^- \cup s} \\ &= \bigoplus_{s \in \mathcal{K} \backslash \mathcal{S}^-} Z_{\mathcal{S}^- \cup s}.\end{aligned}$$

Finally, because each subfile has $F/\binom{K}{r}$ bits, the required cache load at users equals MF, with

$$M = \frac{\binom{K-1}{r-1} + (N-1)\binom{K-2}{r-2}}{\binom{K}{r}} = \frac{r}{K} + (N-1)\frac{r}{K}\frac{r-1}{K-1}.$$

The server further divides each subfile $W_{f,\mathcal{S}}$ into two half-subfiles of equal size $W^I_{f,\mathcal{S}}$ and $W^Q_{f,\mathcal{S}}$. This can be seen, equivalently, as extending the field of the system to $\mathbb{F}_{2^2}$. Let us write

[2]We define $\binom{n}{k} = 0$ if $n < k$.

a subfile from its half-subfiles as $W_{f,\mathcal{S}} = \left\langle W^I_{f,\mathcal{S}}, W^Q_{f,\mathcal{S}} \right\rangle$. At the caches, users use the same partition function to also divide the coded cached subfiles into two coded half-subfiles $Z_{\mathcal{S}} = \left\langle Z^I_{S}, Z^Q_{S} \right\rangle$ and $Z_{f,\mathcal{S}^-} = \left\langle Z^I_{f,\mathcal{S}^-}, Z^Q_{f,\mathcal{S}^-} \right\rangle$. Then, the coded half-subfiles are given by

$$Z^I_{\mathcal{S}} = \bigoplus_{f\in\mathcal{F}} W^I_{f,\mathcal{S}}, \qquad Z^Q_{\mathcal{S}} = \bigoplus_{f\in\mathcal{F}} W^Q_{f,\mathcal{S}},$$
$$Z^I_{f,\mathcal{S}^-} = \bigoplus_{s\notin\mathcal{S}^-} W^I_{f,\mathcal{S}^-\cup s}, \quad Z^Q_{f,\mathcal{S}^-} = \bigoplus_{s\notin\mathcal{S}^-} W^Q_{f,\mathcal{S}^-\cup s}.$$

2) Broadcasting scheme*:* Let us denote the binary sum of the two half-subfiles of subfile $W_{f,\mathcal{S}}$ as $\bar{W}_{f,\mathcal{S}} = W^I_{f,\mathcal{S}} \oplus W^Q_{f,\mathcal{S}}$ and the set of users requesting file W_f as $\mathcal{K}(f)$. The server, first, arbitrarily selects one user leader $u_f \in \mathcal{K}(f)$ for each file $f \in \mathcal{F}$, and then broadcasts the coded subfile $Y_{f,\mathcal{S}} = \left\langle Y^I_{f,\mathcal{S}}, Y^Q_{f,\mathcal{S}} \right\rangle$ with half-subfiles, given by

$$Y^I_{f,\mathcal{S}} = W^I_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)} \bar{W}_{f,u_f\cup\{\mathcal{S}\setminus s\}}, \tag{9}$$
$$Y^Q_{f,\mathcal{S}} = \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)} W^Q_{f,u_f\cup\{\mathcal{S}\setminus s\}} \tag{10}$$

for every file $f \in \mathcal{F}$ and every set $\mathcal{S}$ of r users, i.e. $\mathcal{S} \in \mathcal{T}(r)$, excluding the user leader u_f, i.e. $u_f \notin \mathcal{S}$.

Because, there are N files, and $\binom{K-1}{r}$ sets $\mathcal{S} \in \{\mathcal{S} \in \mathcal{T}(r) : u_f \notin \mathcal{S}\}$ for each file, we need $N\binom{K-1}{r}$ subfile length broadcast transmissions, thus, requiring the rate

$$R = \frac{N\binom{K-1}{r}}{\binom{K}{r}} = N\left(1-\frac{r}{K}\right).$$

In order to facilitate the decoding process described next, we divide the broadcast of these subfiles into $r+1$ consecutive phases $p \in \{0,....,r\}$. At phase p, for every file W_f, we broadcast the subfiles $Y_{f,\mathcal{S}}$ satisfying $|\mathcal{S}\cap\mathcal{K}(f)| = p$.

Recall the definition of $\mathcal{T}_k(r)$ as the set of subsets of r users that include a particular user k and define its complement in $\mathcal{T}(r)$ as $\bar{\mathcal{T}}_k(r) = \{\mathcal{S} \in \mathcal{T}(r) : k \notin \mathcal{S}\}$.

The decoding algorithm is summarized in Algorithm 1. User U_k divides the decoding of all the requested subfiles $W_{\mathbf{d}(k),\mathcal{S}}$ for all $\mathcal{S} \in \mathcal{T}(r)$ into two steps. First, user U_k obtains all subfiles $W_{\mathbf{d}(k),\mathcal{S}}$ for all sets $\mathcal{S} \in \mathcal{T}_k(r)$. This subfiles are coded at some coded subfile in the cache of user U_k. Then, user U_k obtains the subfiles cached by other users, i.e. subfiles $W_{\mathbf{d}(k),\mathcal{S}}$ for all sets $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$.

3) Decoding of requested subfiles coded in the cache of user U_k: We first describe how user U_k obtains subfiles $W_{\mathbf{d}(k),\mathcal{S}}$ for all sets $\mathcal{S} \in \mathcal{T}_k(r)$. Recall that for any set $\mathcal{S} \in \mathcal{T}_k(r)$, user k caches

$$Z_{\mathcal{S}} = W_{\mathbf{d}(k),\mathcal{S}} \oplus \bigoplus_{f\in\mathcal{F}\setminus\mathbf{d}(k)} W_{f,\mathcal{S}}$$

and thus, can obtain $W_{\mathbf{d}(k),\mathcal{S}}$ for all sets $\mathcal{S} \in \mathcal{T}_k(r)$ by first obtaining all the interfering subfiles $W_{f,\mathcal{S}}$ for all $f \in \mathcal{F}\setminus\mathbf{d}(k)$, and then computing

$$W_{\mathbf{d}(k),\mathcal{S}} = Z_{\mathcal{S}} \oplus \bigoplus_{f\in\mathcal{F}\setminus\mathbf{d}(k)} W_{f,\mathcal{S}}. \tag{11}$$

The decoding of these interfering subfiles is divided into r consecutive phases $p = 0,...,r-1$. At phase p, for every file $f \in \mathcal{F}\setminus\mathbf{d}(k)$, user U_k obtains subfile $W_{f,\mathcal{S}}$ for all subsets $\mathcal{S} \in \mathcal{T}_k(r)$ with p users requesting file W_f, i.e. satisfying $|\mathcal{S}\cap\mathcal{K}(f)| = p$. Let us denote this family of sets as $\mathcal{T}_{k,f}(r,p) = \{\mathcal{S} \in \mathcal{T}_k(r) : |\mathcal{S}\cap\mathcal{K}(f)| = p\}$.

Consider first the decoding of the interfering subfiles at phase $p = 0$. This is, subfiles $W_{f,\mathcal{S}}$ for any set $\mathcal{S} \in \mathcal{T}_{k,f}(r,0)$. Observe that, $u_f \notin \mathcal{S}$, and thus, there is a broadcasted subfile $Y_{f,\mathcal{S}}$ associated to each of the sets $\mathcal{S} \in \mathcal{T}_{k,f}(r,0)$, with half-subfiles given by $Y^I_{f,\mathcal{S}} = W^I_{f,\mathcal{S}}$, and $Y^Q_{f,\mathcal{S}} = \bar{W}_{f,\mathcal{S}}$. Consequently, we can obtain $W_{f,\mathcal{S}} = \left\langle W^I_{f,\mathcal{S}}, W^Q_{f,\mathcal{S}} \right\rangle$ as

$$W^I_{f,\mathcal{S}} = Y^I_{f,\mathcal{S}}, \quad W^Q_{f,\mathcal{S}} = Y^Q_{f,\mathcal{S}} \oplus Y^I_{f,\mathcal{S}}. \tag{12}$$

Next, consider the decoding of interfering subfiles at phases p, $r > p \geq 1$. This is, interfering subfiles $W_{f,\mathcal{S}}$ for any set $\mathcal{S} \in \mathcal{T}_{k,f}(r,p)$ for $p = 1,...,r-1$. At phase p, user U_k decodes first all the interfering subfiles $W_{f,\mathcal{S}}$ also cached by the user leader, i.e. $u_f \in \mathcal{S}$. Then, the interfering subfiles $W_{f,\mathcal{S}}$ for all sets $\mathcal{S} \in \mathcal{T}_{k,f}(r,p)$ not cached at the user leader $u_f \notin \mathcal{S}$ can

Algorithm 1 Decoding requested subfiles.

1) **Decode** $W_{f,\mathcal{S}}$ **for all** $\mathcal{S} \in \mathcal{T}_k(r)$ **and** $f \in \mathcal{F}\setminus\mathbf{d}(k)$
   a) **Input:**
      - $Z_{\mathcal{S}}$ for all $\mathcal{S} \in \mathcal{T}_k(r)$,
      - $Z_{f,\mathcal{S}^-}$ for all $f \in \mathcal{F}$ and $\mathcal{S}^- \in \mathcal{T}_k(r-1)$,
      - $Y_{f,\mathcal{S}}$ for all $f \in \mathcal{F}$ and $\mathcal{S} \in \mathcal{T}_k(r)$ satisfying $u_f \notin \mathcal{S}$.
   b) **For** all $f \in \mathcal{F}\setminus\mathbf{d}(k)$, and **for** $p = 0,...,r-1$,
      i) **For** all $\mathcal{S} \in \mathcal{T}_k(r)$ with $|\mathcal{S}\cap\mathcal{K}(f)| = p$ and $u_f \in \mathcal{S}$
         A) Compute $C_{f,\mathcal{S}\setminus u}$ **for** all $u \in \mathcal{S}\cap\mathcal{K}(f)$ as (15)
         B) Compute $\Upsilon_{f,\mathcal{S}\setminus u_f}$ as in (16)
         C) Compute $\Upsilon^I_{f,\mathcal{S}}$ as in (17) and $\Upsilon^Q_{f,\mathcal{S}}$ as in (18)
         D) Compute $W_{f,\mathcal{S}}$ as in (19)
      ii) **For** all $\mathcal{S} \in \mathcal{T}_k(r)$ with $|\mathcal{S}\cap\mathcal{K}(f)| = p$ and $u_f \notin \mathcal{S}$
         A) Compute $W_{f,\mathcal{S}}$ according to (13) and (14)
2) **Decode** $W_{\mathbf{d}(k),\mathcal{S}}$ **for all** $\mathcal{S} \in \mathcal{T}_k(r)$
   a) **Input:**
      - $W_{f,\mathcal{S}}$ for all $\mathcal{S} \in \mathcal{T}_k(r)$ and $f \in \mathcal{F}\setminus\mathbf{d}(k)$
      - $Z_{\mathcal{S}}$ for all $\mathcal{S} \in \mathcal{T}_k(r)$
   b) **For** all $\mathcal{S} \in \mathcal{T}_k(r)$, compute $W_{\mathbf{d}(k),\mathcal{S}}$ as (11)
3) **Decode** $W_{\mathbf{d}(k),\mathcal{S}}$ **for all** $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$
   a) **Input:**
      - $W_{\mathbf{d}(k),\mathcal{S}}$ for all $\mathcal{S} \in \mathcal{T}_k(r)$,
      - $Y_{\mathbf{d}(k),\mathcal{S}}$ for all $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ satisfying $u_f \notin \mathcal{S}$
   b) **Set** $f = \mathbf{d}(k)$
   c) **If** $k = u_f$,
      i) **For** all $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ compute $W_{f,\mathcal{S}}$ according to (25) and (26)
   d) **If** $k \neq u_f$,
      i) **For** all $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ satisfying $u_f \in \mathcal{S}$ compute $W_{f,\mathcal{S}}$ according to (27) and (28)
      ii) **For** all $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ satisfying $u_f \notin \mathcal{S}$ compute $W_{f,\mathcal{S}}$ according to (25) and (26)

be obtained as, $W_{f,\mathcal{S}} = \left\langle W^I_{f,\mathcal{S}}, W^Q_{f,\mathcal{S}} \right\rangle$ with

$$W^I_{f,\mathcal{S}} = Y^I_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)} \bar{W}_{f,u_f\cup\{\mathcal{S}\setminus s\}}, \quad (13)$$

$$W^Q_{f,\mathcal{S}} = Y^Q_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)} W^Q_{f,u_f\cup\{\mathcal{S}\setminus s\}} \oplus W^I_{f,\mathcal{S}}. \quad (14)$$

Observe that, given that $u_f \notin \mathcal{S}$, the broadcasted subfile $Y_{f,\mathcal{S}}$ exits. Moreover, for all $u \in \mathcal{S}\cap\mathcal{K}(f)$ the set $u_f\cup\mathcal{S}\setminus u$ contains u_f and satisfies $|u_f \cup \mathcal{S}\setminus u| = p$ and thus, user U_k has already obtained $W_{f,u_f\cup\mathcal{S}\setminus u}$.

Finally, we detail the decoding of interfering subfiles $W_{f,\mathcal{S}}$ for any set $\mathcal{S} \in \mathcal{T}_{k,f}(r,p)$ with $u_f \in \mathcal{S}$. We summarize the decoding process first, and prove it later. Observe that for all $u \in \mathcal{S}\cap\mathcal{K}(f)$, we have $|\{\mathcal{S}\setminus u\}\cap\mathcal{K}(f)| = p-1$, and thus from decoding phase $p-1$, user U_k already has $W_{f,\{\mathcal{S}\setminus u\}\cup v}$ for all $v \in \bar{\mathcal{K}}(f)\setminus\{\mathcal{S}\setminus u\}$. This allows user U_k to compute

$$C_{f,\mathcal{S}\setminus u} = Z_{f,\mathcal{S}\setminus u} \oplus \bigoplus_{v\in\bar{\mathcal{K}}(f)\setminus\{\mathcal{S}\setminus u\}} W_{f,\{\mathcal{S}\setminus u\}\cup v} \quad (15)$$

for all $u \in \mathcal{S}\cap\mathcal{K}(f)$. Next, observe that since $u_f \in \mathcal{S}$, and $u_f \notin \mathcal{K}(f)\setminus\mathcal{S}$, there is a broadcasted subfile $Y_{f,\{\mathcal{S}\setminus u_f\}\cup s}$ for all $s \in \mathcal{K}(f)\setminus\mathcal{S}$. This allows user U_k to compute

$$\Upsilon_{f,\mathcal{S}\setminus u_f} = \bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} Y_{f,\{\mathcal{S}\setminus u_f\}\cup s}. \quad (16)$$

Next, we partition $\Upsilon_{f,\mathcal{S}\setminus u_f}$ and $C_{f,\mathcal{S}\setminus u}$ for all $u \in \mathcal{S}\cap\mathcal{K}(f)$ into half-subfiles, i.e. $\Upsilon_{f,\mathcal{S}\setminus u_f} = \left\langle \Upsilon^I_{f,\mathcal{S}\setminus u_f}, \Upsilon^Q_{f,\mathcal{S}\setminus u_f} \right\rangle$, $C_{f,\mathcal{S}\setminus u} = \left\langle C^I_{f,\mathcal{S}\setminus u}, C^Q_{f,\mathcal{S}\setminus u} \right\rangle$ and define $\bar{\Upsilon}_{f,\mathcal{S}\setminus u_f} = \Upsilon^I_{f,\mathcal{S}\setminus u_f} \oplus \Upsilon^Q_{f,\mathcal{S}\setminus u_f}$, and $\bar{C}_{f,\mathcal{S}\setminus u} = C^I_{f,\mathcal{S}\setminus u} \oplus C^Q_{f,\mathcal{S}\setminus u}$. Then, user U_k computes

$$\Upsilon^I_{f,\mathcal{S}} = \Upsilon^I_{f,\mathcal{S}\setminus u_f} \oplus C^I_{f,\mathcal{S}\setminus u_f} \oplus \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} \bar{C}_{f,\mathcal{S}\setminus u} \quad (17)$$

$$\Upsilon^Q_{f,\mathcal{S}} = \Upsilon^Q_{f,\mathcal{S}\setminus u_f} \oplus \bar{C}_{f,\mathcal{S}\setminus u_f} \oplus \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} C^Q_{f,\mathcal{S}\setminus u} \quad (18)$$

and obtains subfile $W_{f,\mathcal{S}} = \left\langle W^I_{f,\mathcal{S}}, W^Q_{f,\mathcal{S}} \right\rangle$ as

$$W^I_{f,\mathcal{S}} = \begin{cases} \Upsilon^Q_{f,\mathcal{S}} & \text{if } |\mathcal{K}(f)\setminus u_f| \text{ is odd} \\ \Upsilon^I_{f,\mathcal{S}} & \text{otherwise} \end{cases}$$
$$W^Q_{f,\mathcal{S}} = \begin{cases} \Upsilon^I_{f,\mathcal{S}} & \text{if } |\mathcal{K}(f)\setminus u_f| \text{ is odd} \\ \Upsilon^I_{f,\mathcal{S}} \oplus \Upsilon^Q_{f,\mathcal{S}} & \text{otherwise} \end{cases} \quad (19)$$

In the following, we prove the correctness of this decoding process. First, observe that for all $u \in \mathcal{S}\cap\mathcal{K}(f)$, we have

$$\begin{aligned} C_{f,\mathcal{S}\setminus u} &= Z_{f,\mathcal{S}\setminus u} \oplus \bigoplus_{v\in\bar{\mathcal{K}}(f)\setminus\{\mathcal{S}\setminus u\}} W_{f,\{\mathcal{S}\setminus u\}\cup v} \\ &= \bigoplus_{s\in\mathcal{K}\setminus\{\mathcal{S}\setminus u\}} W_{f,\{\mathcal{S}\setminus u\}\cup s} \bigoplus_{v\in\bar{\mathcal{K}}(f)\setminus\{\mathcal{S}\setminus u\}} W_{f,\{\mathcal{S}\setminus u\}\cup v} \\ &= \bigoplus_{s\in\mathcal{K}(f)\setminus\{\mathcal{S}\setminus u\}} W_{f,\{\mathcal{S}\setminus u\}\cup s} \\ &= W_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} W_{f,\{\mathcal{S}\setminus u\}\cup s} \end{aligned}$$

from which we can write

$$\bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} W_{f,\{\mathcal{S}\setminus u\}\cup s} = C_{f,\mathcal{S}\setminus u} \oplus W_{f,\mathcal{S}}. \quad (20)$$

Next, observe that for any $s \in \mathcal{K}(f)\setminus\mathcal{S}$, we can write $Y^I_{f,\{\mathcal{S}\setminus u_f\}\cup s}$

$$\begin{aligned} &= W^I_{f,\{\mathcal{S}\setminus u_f\}\cup s} \oplus \bigoplus_{u\in\{\{\mathcal{S}\setminus u_f\}\cup s\}\cap\mathcal{K}(f)} \bar{W}_{f,u_f\cup\{\{\mathcal{S}\setminus u_f\}\cup s\}\setminus u} \\ &= W^I_{f,\{\mathcal{S}\setminus u_f\}\cup s} \oplus \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} \bar{W}_{f,\{\mathcal{S}\setminus u\}\cup s} \quad (21) \end{aligned}$$

where (21) follows since for $u = s$, we have $u_f \cup \{\{\mathcal{S}\setminus u_f\}\cup s\}\setminus u = \mathcal{S}$, and for $u \neq s$, since $u_f \in \mathcal{S}$, we have $u_f \cup \{\{\mathcal{S}\setminus u_f\}\cup s\}\setminus u = \{\mathcal{S}\setminus u\}\cup s$. Similarly, we can write $Y^Q_{f,\{\mathcal{S}\setminus u_f\}\cup s}$

$$= \bar{W}_{f,\{\mathcal{S}\setminus u_f\}\cup s} \oplus W^Q_{f,\mathcal{S}} \oplus \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} W^Q_{f,\{\mathcal{S}\setminus u\}\cup s}. \quad (22)$$

Now, let $\Upsilon^I_{f,\mathcal{S}\setminus u_f} \triangleq \bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} Y^I_{f,\{\mathcal{S}\setminus u_f\}\cup s}$, and $\Upsilon^Q_{f,\mathcal{S}\setminus u_f} \triangleq \bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} Y^Q_{f,\{\mathcal{S}\setminus u_f\}\cup s}$ then, it follows from (20) and (21) that

$$\begin{aligned} \Upsilon^I_{f,\mathcal{S}\setminus u_f} = C^I_{f,\mathcal{S}\setminus u_f} \oplus W^I_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} \bar{W}_{f,\mathcal{S}} \\ \oplus \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} \bar{C}_{f,\mathcal{S}\setminus u} \oplus \bar{W}_{f,\mathcal{S}} \quad (23) \end{aligned}$$

and, similarly, from (20) and (22) that

$$\begin{aligned} \Upsilon^Q_{f,\mathcal{S}\setminus u_f} = \bar{C}_{f,\mathcal{S}\setminus u_f} \oplus \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} W^Q_{f,\mathcal{S}} \\ \oplus \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} C^Q_{f,\mathcal{S}\setminus u} \oplus W^Q_{f,\mathcal{S}}. \quad (24) \end{aligned}$$

Next observe that

$$\begin{aligned} &\textstyle\bigoplus_{s\in\mathcal{K}(f)\setminus\mathcal{S}} W_{f,\mathcal{S}} \oplus \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} W_{f,\mathcal{S}} \\ &= \bigoplus_{s\in\mathcal{K}(f)\setminus u_f} W_{f,\mathcal{S}} \\ &= \begin{cases} W_{f,\mathcal{S}} & \text{if } |\mathcal{K}(f)\setminus u_f| \text{ is odd} \\ \mathbf{0} & \text{otherwise.} \end{cases} \end{aligned}$$

and thus $\Upsilon^I_{\mathcal{S}\setminus u_f} \oplus C^I_{f,\mathcal{S}\setminus u_f} \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} \bar{C}_{f,\mathcal{S}\setminus u}$

$$\begin{aligned} &= W^I_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{K}(f)\setminus u_f} \bar{W}_{f,\mathcal{S}} \\ &= \begin{cases} W^Q_{f,\mathcal{S}} & \text{if } |\mathcal{K}(f)\setminus u_f| \text{ is odd} \\ W^I_{f,\mathcal{S}} & \text{otherwise} \end{cases} \end{aligned}$$

and $\Upsilon^Q_{\mathcal{S}\setminus u_f} \oplus \bar{C}_{f,\mathcal{S}\setminus u_f} \bigoplus_{u\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} C^Q_{f,\mathcal{S}\setminus u}$

$$\begin{aligned} &= \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{K}(f)\setminus u_f} W^Q_{f,\mathcal{S}} \\ &= \begin{cases} W^I_{f,\mathcal{S}} & \text{if } |\mathcal{K}(f)\setminus u_f| \text{ is odd} \\ \bar{W}_{f,\mathcal{S}} & \text{otherwise.} \end{cases} \end{aligned}$$

If $|\mathcal{K}(f)\setminus u_f|$ is even, we obtain $W^Q_{f,\mathcal{S}}$ as $W^Q_{f,\mathcal{S}} = \bar{W}_{f,\mathcal{S}} \oplus W^I_{f,\mathcal{S}}$. Finally, we have $W_{f,\mathcal{S}} = \left\langle W^I_{f,\mathcal{S}}, W^Q_{f,\mathcal{S}} \right\rangle$.

4) Decoding of requested subfiles not cached: Next, we show how user U_k obtains all the remaining requested subfiles $W_{\mathbf{d}(k),\mathcal{S}}$ for all sets $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$. These subfiles satisfy $k \notin \mathcal{S}$ and are, thus, not coded in the cache of user U_k. Hereafter $f = \mathbf{d}(k)$. Let us first suppose user U_k is leader, i.e. $k = u_f$. Observe that then, associated to any set $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$, since $k = u_f \notin \mathcal{S}$, there exists a broadcasted subfile $Y_{f,\mathcal{S}}$. In addition, user $k = u_f$ has already obtained all the requested subfiles coded in its cache, in particular subfiles $W_{f,u_f \cup \mathcal{S}\backslash s}$ for all $s \in \mathcal{S} \cap \mathcal{K}(f)$, and can compute $W_{f,\mathcal{S}} = \left\langle W^I_{f,\mathcal{S}}, W^Q_{f,\mathcal{S}} \right\rangle$ as

$$W^I_{f,\mathcal{S}} = Y^I_{f,\mathcal{S}} \oplus \bigoplus_{s \in \mathcal{S} \cap \mathcal{K}(f)} \bar{W}_{f,u_f \cup \mathcal{S}\backslash s}, \quad (25)$$

$$W^Q_{f,\mathcal{S}} = W^I_{f,\mathcal{S}} \oplus Y^Q_{f,\mathcal{S}} \oplus \bigoplus_{s \in \mathcal{S} \cap \mathcal{K}(f)} W^Q_{f,u_f \cup \mathcal{S}\backslash s}. \quad (26)$$

for all sets $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$. Instead, if user U_k is no leader, i.e. $k \in \mathcal{K}(f)\backslash u_f$ he obtains, first, the subfiles $W_{f,\mathcal{S}}$ for all the set $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ satisfying $u_f \in \mathcal{S}$. Observe that for each of these sets, associated to the set $\mathcal{S} \cup k \backslash u_f$, there exists the broadcasted subfile $Y_{f,\mathcal{S}\cup k\backslash u_f}$ with half-subfiles given by $Y^I_{f,\mathcal{S}\cup k\backslash u_f}$

$$= W^I_{f,\mathcal{S}\cup k\backslash u_f} \oplus \bigoplus_{s \in \{\mathcal{S}\cup k\backslash u_f\} \cap \mathcal{K}(f)} \bar{W}_{f,u_f \cup \{\{\mathcal{S}\cup k\backslash u_f\}\backslash s\}}$$

$$= W^I_{f,\mathcal{S}\cup k\backslash u_f} \oplus \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{s \in \{\mathcal{S}\backslash u_f\} \cap \mathcal{K}(f)} \bar{W}_{f,\mathcal{S}\cup k\backslash s}$$

and $Y^Q_{f,\mathcal{S}\cup k\backslash u_f}$

$$= \bar{W}_{f,\mathcal{S}\cup k\backslash u_f} \oplus W^Q_{f,\mathcal{S}} \oplus \bigoplus_{s \in \{\mathcal{S}\backslash u_f\} \cap \mathcal{K}(f)} W^Q_{f,\mathcal{S}\cup k\backslash s}.$$

Given that user U_k has already obtained all the requested subfiles coded in its own cache, in particular subfiles $W_{f,\mathcal{S}\cup k\backslash s}$ for all $s \in \mathcal{S} \cap \mathcal{K}(f)$, he can compute $W_{f,\mathcal{S}} = \left\langle W^I_{f,\mathcal{S}}, W^Q_{f,\mathcal{S}} \right\rangle$ as

$$W^Q_{f,\mathcal{S}} = Y^Q_{f,\mathcal{S}\cup k\backslash u_f} \oplus \bar{W}_{f,\mathcal{S}\cup k\backslash u_f} \oplus \bigoplus_{s \in \{\mathcal{S}\backslash u_f\} \cap \mathcal{K}(f)} W^Q_{f,\mathcal{S}\cup k\backslash s} \quad (27)$$

$$W^I_{f,\mathcal{S}} = W^Q_{f,\mathcal{S}} \oplus Y^I_{f,\mathcal{S}\cup k\backslash u_f} \oplus W^I_{f,\mathcal{S}\cup k\backslash u_f} \oplus \bigoplus_{s \in \{\mathcal{S}\backslash u_f\} \cap \mathcal{K}(f)} \bar{W}_{f,\mathcal{S}\cup k\backslash s} \quad (28)$$

for all sets $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ with $u_f \in \mathcal{S}$. Now user U_k can mimic the computations performed at the user leader u_f in (25) and (26), to obtain the remaining subfiles, i.e. $W_{f,\mathcal{S}}$ for all sets $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ satisfying $u_f \notin \mathcal{S}$.

C. Extension to arbitrary requested files

The strategy presented in previous subsection requires all N files to be requested by at least one user. In this section, we show that the same scheme can be applied for any request if configured to $K+1$ users instead of K.

1) Prefetching scheme*:* We use the subfile partitions and prefetching scheme specified in previous section, but as if there were $K+1$ users instead of K. As a result, each file is partitioned into $\binom{K+1}{r}$ subfiles and thus, the required cache load at users equals MF, with

$$M = \frac{r}{K+1} + (N-1)\frac{r}{K+1}\frac{r-1}{K}$$

2) Delivery scheme*:* For the delivery, if all files are requested, since the caching system is configured for $K+1$ users, we need to add one file request for the $K+1$th "virtual" user. Given that all files are already requested by some user, the $K+1$th user can request any of the files in $\mathcal{F}$. Then, we apply the delivery strategy as described in previous subsections. In that case, every user obtains their desired file, with a rate

$$R = N\left(1 - \frac{r}{K+1}\right). \quad (29)$$

Instead, if some files are not requested, i.e. $N_e(\mathbf{d}) \leq N-1$, we first transform the coded caching system with N files into caching system with $N_e(\mathbf{d})+1$ files. To that end, we define a new file W_0 as the binary sum of all files not requested

$$W_0 = \bigoplus_{f \in \mathcal{F}\backslash\mathcal{N}_e(\mathbf{d})} W_f$$

where $\mathcal{N}_e(\mathbf{d}) \subseteq \mathcal{F}$. Then, the set of files in the system $\mathcal{F}$ is given by $\mathcal{F}_0(\mathbf{d}) = \{0 \cup \mathcal{N}_e(\mathbf{d})\}$. The server can obtain the subfiles of the new file W_0 from the not requested subfiles as

$$W_{0,\mathcal{S}} = \bigoplus_{f \in \mathcal{F}\backslash\mathcal{N}_e(\mathbf{d})} W_{f,\mathcal{S}}.$$

Similarly, users can consider the coded cached subfiles in (5), as if the set of file was $\mathcal{F}_0(\mathbf{d})$, since

$$Z_{\mathcal{S}} = \bigoplus_{f \in \mathcal{F}} W_{f,\mathcal{S}}$$

$$= \bigoplus_{f \in \mathcal{F}\backslash\mathcal{N}_e(\mathbf{d})} W_{f,\mathcal{S}} \oplus \bigoplus_{f \in \mathcal{N}_e(\mathbf{d})} W_{f,\mathcal{S}}$$

$$= \bigoplus_{f \in 0 \cup \mathcal{N}_e(\mathbf{d})} W_{f,\mathcal{S}}.$$

and, can obtain the coded subfiles in (6) associated to file W_0 as

$$Z_{0,\mathcal{S}^-} = \bigoplus_{f \in \mathcal{F}\backslash\mathcal{N}_e(\mathbf{d})} Z_{f,\mathcal{S}^-}$$

$$= \bigoplus_{s \in \mathcal{K}\backslash\mathcal{S}^-} \bigoplus_{f \in \mathcal{F}\backslash\mathcal{N}_e(\mathbf{d})} W_{f,\mathcal{S}^- \cup s}$$

$$= \bigoplus_{s \in \mathcal{K}\backslash\mathcal{S}^-} W_{0,\mathcal{S}.}$$

Finally, since we have replaced the set of files $\mathcal{F}$ by $\mathcal{F}_0(\mathbf{d})$, we can assume that the $K+1$ virtual user requests file W_0. Then, observe that all files in $\mathcal{F}_0(\mathbf{d})$ are requested, and thus we can apply the delivery strategy as described in previous subsection, with a rate

$$R = (N_e(\mathbf{d})+1)\left(1 - \frac{r}{K+1}\right). \quad (30)$$

Combining (29) and (30), we can write the rate as presented in Theorem 1.

Coded cached subfiles at user U_1	
$Z_{\{1,2,3\}}$	$W_{1,\{1,2,3\}} \oplus W_{2,\{1,2,3\}}$
$Z_{\{1,2,4\}}$	$W_{1,\{1,2,4\}} \oplus W_{2,\{1,2,4\}}$
$Z_{\{1,2,5\}}$	$W_{1,\{1,2,5\}} \oplus W_{2,\{1,2,5\}}$
$Z_{\{1,3,4\}}$	$W_{1,\{1,3,4\}} \oplus W_{2,\{1,3,4\}}$
$Z_{\{1,3,5\}}$	$W_{1,\{1,3,5\}} \oplus W_{2,\{1,3,5\}}$
$Z_{\{1,4,5\}}$	$W_{1,\{1,4,5\}} \oplus W_{2,\{1,4,5\}}$
$Z_{1,\{1,2\}}$	$W_{1,\{1,2,3\}} \oplus W_{1,\{1,2,4\}} \oplus W_{1,\{1,2,5\}}$
$Z_{1,\{1,3\}}$	$W_{1,\{1,3,2\}} \oplus W_{1,\{1,3,4\}} \oplus W_{1,\{1,3,5\}}$
$Z_{1,\{1,4\}}$	$W_{1,\{1,4,2\}} \oplus W_{1,\{1,4,3\}} \oplus W_{1,\{1,4,5\}}$
Coded subfiles computed at user U_1	
$Z_{1,\{1,5\}}$	$W_{1,\{1,5,2\}} \oplus W_{1,\{1,5,3\}} \oplus W_{1,\{1,5,4\}}$
$Z_{2,\{1,2\}}$	$W_{2,\{1,2,3\}} \oplus W_{2,\{1,2,4\}} \oplus W_{2,\{1,2,5\}}$
$Z_{2,\{1,3\}}$	$W_{2,\{1,3,2\}} \oplus W_{2,\{1,3,4\}} \oplus W_{2,\{1,3,5\}}$
$Z_{2,\{1,4\}}$	$W_{2,\{1,4,2\}} \oplus W_{2,\{1,4,3\}} \oplus W_{2,\{1,4,5\}}$
$Z_{2,\{1,5\}}$	$W_{2,\{1,5,2\}} \oplus W_{2,\{1,5,3\}} \oplus W_{2,\{1,5,4\}}$

Table I: Coded cached and computable coded subfiles at user U_1 for the caching system of Example 1.

V. Example

In this section, we apply the caching scheme presented in section IV-B to a particular example. Consider a caching system with $N = 2$ files, $K = 5$ users, and a caching capacity of MF bits with $M = \frac{9}{10}$, which corresponds to $r = 3$. Each file W_f, $f = \{1, 2\}$ is divide into $\binom{K}{r} = 10$ subfiles, i.e. $W_{f,\{1,2,3\}}$, $W_{f,\{1,2,4\}}$, $W_{f,\{1,2,5\}}$, $W_{f,\{1,3,4\}}$, $W_{f,\{1,3,5\}}$, $W_{f,\{1,4,5\}}$, $W_{f,\{2,3,4\}}$, $W_{f,\{2,3,5\}}$, $W_{f,\{2,4,5\}}$, and $W_{f,\{3,4,5\}}$.

The cache of user U_k stores a linear combination of the subfiles $W_{f,\mathcal{S}}$ of both files satisfying $k \in \mathcal{S}$. For the running example, the cache of user U_1 is detailed in Table I. Observe that in this case, we have chosen to omit coded cached subfiles $Z_{f,\mathcal{S}^-}$ satisfying $f = 2$ or $5 \in \mathcal{S}^-$. These subfiles (see the Table I) can be computed from the cache content at user U_1 as

$$\begin{aligned}
Z_{1,\{1,5\}} &= Z_{1,\{1,2\}} \oplus Z_{1,\{1,3\}} \oplus Z_{1,\{1,4\}},\\
Z_{2,\{1,2\}} &= Z_{1,\{1,2\}} \oplus Z_{\{1,2,3\}} \oplus Z_{\{1,2,4\}} \oplus Z_{\{1,2,5\}},\\
Z_{2,\{1,3\}} &= Z_{1,\{1,3\}} \oplus Z_{\{1,3,2\}} \oplus Z_{\{1,3,4\}} \oplus Z_{\{1,3,5\}},\\
Z_{2,\{1,4\}} &= Z_{1,\{1,4\}} \oplus Z_{\{1,4,2\}} \oplus Z_{\{1,4,3\}} \oplus Z_{\{1,4,5\}},\\
Z_{2,\{1,5\}} &= Z_{1,\{1,5\}} \oplus Z_{\{1,5,2\}} \oplus Z_{\{1,5,3\}} \oplus Z_{\{1,5,4\}}.
\end{aligned}$$

Given that every file is divided into 10 subfiles, and each user caches 9 coded subfiles, the cache load condition $M = \frac{9}{10}$ is satisfied.

Given the above prefetching scheme, we illustrate our proposed delivery strategy for a representative demand, where users U_1, U_3, U_5 request file W_1, and users U_2 and U_4 request file W_2. This corresponds to the demand vector $\mathbf{d} = [1, 2, 1, 2, 1]$, and $\mathcal{K}(1) = \{1, 3, 5\}$ and $\mathcal{K}(2) = \{2, 4\}$. Observe that all files are requested. First, the server selects one user leader for each file, i.e. $u_1 = 1$ and $u_2 = 2$. Then, the server broadcasts, according to (9) and (10), the coded subfiles $Y_{f,\mathcal{S}} = \left\langle Y^I_{f,\mathcal{S}}, Y^Q_{f,\mathcal{S}} \right\rangle$ specified in Table II. Observe that the total number of broadcasted subfiles is $N\binom{K-1}{r} = 8$

Next, we detail the decoding process at user U_1. He first obtains the subfiles $W_{1,\mathcal{S}}$ coded its cache, i.e. $1 \in \mathcal{S}$.

$Y_{1,\{2,3,4\}}$	$Y^I_{1,\{2,3,4\}}$	$W^I_{1,\{2,3,4\}} \oplus \bar{W}_{1,\{1,2,4\}}$
	$Y^Q_{1,\{2,3,4\}}$	$\bar{W}_{1,\{2,3,4\}} \oplus W^Q_{1,\{1,2,4\}}$
$Y_{1,\{2,3,5\}}$	$Y^I_{1,\{2,3,5\}}$	$W^I_{1,\{2,3,5\}} \oplus \bar{W}_{1,\{2,1,5\}} \oplus \bar{W}_{1,\{2,3,1\}}$
	$Y^Q_{1,\{2,3,5\}}$	$\bar{W}_{1,\{2,3,5\}} \oplus W^Q_{1,\{2,1,5\}} \oplus W^Q_{1,\{2,3,1\}}$
$Y_{1,\{2,4,5\}}$	$Y^I_{1,\{2,4,5\}}$	$W^I_{1,\{2,4,5\}} \oplus \bar{W}_{1,\{2,4,1\}}$
	$Y^Q_{1,\{2,4,5\}}$	$\bar{W}_{1,\{2,4,5\}} \oplus W^Q_{1,\{2,4,1\}}$
$Y_{1,\{3,4,5\}}$	$Y^I_{1,\{3,4,5\}}$	$W^I_{1,\{3,4,5\}} \oplus \bar{W}_{1,\{1,4,5\}} \oplus \bar{W}_{1,\{3,4,1\}}$
	$Y^Q_{1,\{3,4,5\}}$	$\bar{W}_{1,\{3,4,5\}} \oplus W^Q_{1,\{1,4,5\}} \oplus W^Q_{1,\{3,4,1\}}$
$Y_{2,\{1,3,4\}}$	$Y^I_{2,\{1,3,4\}}$	$W^I_{2,\{1,3,4\}} \oplus \bar{W}_{2,\{1,3,2\}}$
	$Y^Q_{2,\{1,3,4\}}$	$\bar{W}_{2,\{1,3,4\}} \oplus W^Q_{2,\{1,3,2\}}$
$Y_{2,\{1,3,5\}}$	$Y^I_{2,\{1,3,5\}}$	$W^I_{2,\{1,3,5\}}$
	$Y^Q_{2,\{1,3,5\}}$	$\bar{W}_{2,\{1,3,5\}}$
$Y_{2,\{1,4,5\}}$	$Y^I_{2,\{1,4,5\}}$	$W^I_{2,\{1,4,5\}} \oplus \bar{W}_{2,\{1,2,5\}}$
	$Y^Q_{2,\{1,4,5\}}$	$\bar{W}_{2,\{1,4,5\}} \oplus W^Q_{2,\{1,2,5\}}$
$Y_{2,\{3,4,5\}}$	$Y^I_{2,\{3,4,5\}}$	$W^I_{2,\{3,4,5\}} \oplus \bar{W}_{2,\{3,2,5\}}$
	$Y^Q_{2,\{3,4,5\}}$	$\bar{W}_{2,\{3,4,5\}} \oplus W^Q_{2,\{3,2,5\}}$

Table II: Broadcasted coded half-subfiles for the demand $\mathbf{d} = [1, 2, 1, 2, 1]$

$p = \lvert\mathcal{S} \cap \mathcal{K}(2)\rvert$	$u_2 = 2$	$W_{2,\mathcal{S}}$
0	$u_2 \notin \mathcal{S}$	$W_{2,\{1,3,5\}}$
1	$u_2 \in \mathcal{S}$	$W_{2,\{1,2,3\}}$
		$W_{2,\{1,2,5\}}$
	$u_2 \notin \mathcal{S}$	$W_{2,\{1,3,4\}}$
		$W_{2,\{1,4,5\}}$
2	$u_2 \in \mathcal{S}$	$W_{2,\{1,2,4\}}$

Table III: Classification of interfering subfiles $W_{2,\mathcal{S}}$ at user U_1, i.e. $1 \in \mathcal{S}$

According to (11), user U_1 can obtain these subfiles by, first, obtaining the interfering subfile $W_{2,\mathcal{S}}$, and then computing

$$W_{1,\mathcal{S}} = Z_{\mathcal{S}} \oplus W_{2,\mathcal{S}}$$

for all sets $\mathcal{S}$ satisfying $1 \in \mathcal{S}$. To that end, user U_1 divides the decoding of these interfering subfiles into 3 consecutive phases $p = \{0, 1, 2\}$. In phase p, user U_1 obtains all subfiles $W_{2,\mathcal{S}}$ satisfying $1 \in \mathcal{S}$ and $p = \lvert\mathcal{S} \cap \mathcal{K}(2)\rvert$. Moreover, within each phase he obtains, first, the subfiles that are coded cached at the user leader, $u_2 = 2$, i.e. $2 \in \mathcal{S}$. For the running example, the interfering subfiles at user U_1 are classified in Table III according to the above decoding phases. The only phase 0 interfering subfile is $W_{2,\{1,3,5\}}$, which can be obtained according to (12) as

$$\begin{aligned}
W^I_{2,\{1,3,5\}} &= Y^I_{2,\{1,3,5\}},\\
W^Q_{2,\{1,3,5\}} &= Y^Q_{2,\{1,3,5\}} \oplus Y^I_{2,\{1,3,5\}}.
\end{aligned}$$

Phase 1 interfering subfiles satisfying $u_2 \in \mathcal{S}$ are $W_{2,\{1,2,3\}}$ and $W_{2,\{1,2,5\}}$. Observe that for both sets $\mathcal{S} \in \{\{1, 2, 3\}, \{1, 2, 5\}\}$, from (17) and (18), we obtain

$$\begin{aligned}
\Upsilon^I_{2,\mathcal{S}} &= \Upsilon^I_{2,\mathcal{S}\setminus 2} \oplus C^I_{2,\mathcal{S}\setminus 2},\\
\Upsilon^Q_{2,\mathcal{S}} &= \Upsilon^Q_{2,\mathcal{S}\setminus 2} \oplus \bar{C}_{2,\mathcal{S}\setminus 2}
\end{aligned}$$

and from (15) and (16), respectively

$$\begin{aligned}
C_{2,\mathcal{S}\setminus 2} &= Z_{2,\mathcal{S}\setminus 2} \oplus W_{2,\{1,3,5\}},\\
\Upsilon_{2,\mathcal{S}\setminus 2} &= Y_{2,\mathcal{S}\setminus 2}.
\end{aligned}$$

Then, given that $|\mathcal{K}(2)\backslash u_f| = 1$ is odd, user U_1 obtains $W_{2,\mathcal{S}} = \left\langle W^I_{2,\mathcal{S}}, W^Q_{2,\mathcal{S}} \right\rangle$ as

$$\begin{aligned} W^I_{2,\mathcal{S}} &= \Upsilon^Q_{2,\mathcal{S}} \\ &= Y^Q_{2,\{\mathcal{S}\backslash 2\}\cup 4} \oplus \left[\bar{Z}_{2,\mathcal{S}\backslash 2} \oplus \bar{W}_{2,\{1,3,5\}}\right], \end{aligned}$$

$$\begin{aligned} W^Q_{2,\mathcal{S}} &= \Upsilon^I_{2,\mathcal{S}} \\ &= Y^I_{2,\{\mathcal{S}\backslash 2\}\cup 4} \oplus \left[Z^I_{2,\mathcal{S}\backslash 2} \oplus W^I_{2,\{1,3,5\}}\right]. \end{aligned}$$

Phase 1 interfering subfiles satisfying $u_2 \notin \mathcal{S}$ are $W_{2,\{1,3,4\}}$ and $W_{2,\{1,4,5\}}$. User U_1 obtains this interfering subfile according to (13) and 14) as

$$\begin{aligned} W^I_{2,\{1,3,4\}} &= Y^I_{2,\{1,3,4\}} \oplus \bar{W}_{2,\{1,2,3\}}, \\ W^Q_{2,\{1,3,4\}} &= Y^Q_{2,\{1,3,4\}} \oplus W^Q_{2,\{1,2,3\}} \oplus W^I_{2,\{1,3,4\}}, \end{aligned}$$

and

$$\begin{aligned} W^I_{2,\{1,4,5\}} &= Y^I_{2,\{1,4,5\}} \oplus \bar{W}_{2,\{1,2,5\}}, \\ W^Q_{2,\{1,4,5\}} &= Y^Q_{2,\{1,4,5\}} \oplus W^Q_{2,\{1,2,5\}} \oplus W^I_{2,\{1,4,5\}}. \end{aligned}$$

Finally, user U_1 can obtain the remaining phase 2 subfile $W_{2,\{1,2,4\}}$, according to (17) and (18). However, given that this is the only interfering subfile left, he can obtain it directly from, either $Z_{2,\{1,4\}}$ or $Z_{2,\{1,2\}}$, as

$$\begin{aligned} W_{2,\{1,2,4\}} &= Z_{2,\{1,4\}} \oplus W_{2,\{1,4,3\}} \oplus W_{2,\{1,4,5\}}, \\ W_{2,\{1,2,4\}} &= Z_{2,\{1,2\}} \oplus W_{2,\{1,2,3\}} \oplus W_{2,\{1,2,5\}}. \end{aligned}$$

Once all interfering subfiles $W_{2,\mathcal{S}}$ for all $\mathcal{S}$ satisfying $1 \in \mathcal{S}$ are decoded. User U_1 decodes the requested subfiles only available at other users' cache. These are the subfiles $W_{1,\mathcal{S}}$ for $\mathcal{S} \in \{\{2,3,4\},\{2,3,5\},\{2,4,5\}\}$. Given that user U_1 is leader, he obtains these subfiles according to (25) and (26), as

$$W^I_{1,\{2,3,4\}} = Y^I_{1,\{2,3,4\}} \oplus \bar{W}_{1,\{1,2,4\}}, \tag{31}$$
$$W^Q_{1,\{2,3,4\}} = W^I_{1,\{2,3,4\}} \oplus Y^Q_{1,\{2,3,4\}} \oplus W^Q_{1,\{1,2,4\}}, \tag{32}$$
$$W^I_{1,\{2,3,5\}} = Y^I_{1,\{2,3,5\}} \oplus \bar{W}_{1,\{1,2,5\}} \oplus \bar{W}_{1,\{1,2,3\}}, \tag{33}$$
$$W^Q_{1,\{2,3,5\}} = W^I_{1,\{2,3,5\}} \oplus Y^Q_{1,\{2,3,5\}} \oplus W^Q_{1,\{1,2,5\}} \oplus W^Q_{1,\{1,2,3\}} \tag{34}$$
$$W^I_{1,\{2,4,5\}} = Y^I_{1,\{2,4,5\}} \oplus \bar{W}_{1,\{1,2,4\}}, \tag{35}$$
$$W^Q_{1,\{2,4,5\}} = W^I_{1,\{2,4,5\}} \oplus Y^Q_{1,\{2,4,5\}} \oplus W^Q_{1,1,2,4}. \tag{36}$$

For completeness, as the decoding of the requested subfiles at other users' cache depends on whether the decoding user is leader or not, let us also consider the decoding process at the non user leader U_3. User U_3 first obtains the subfiles $W_{1,\{1,2,4\}}$, $W_{1,\{1,2,5\}}$, $W_{1,\{1,4,5\}}$ coded cached at the user leader U_1, according to (27) and (28) as

$$\begin{aligned} W^Q_{1,\{1,2,4\}} &= Y^Q_{1,\{2,3,4\}} \oplus \bar{W}_{1,\{2,3,4\}}, \\ W^I_{1,\{1,2,4\}} &= W^Q_{1,\{1,2,4\}} \oplus Y^I_{1,\{2,3,4\}} \oplus W^I_{1,\{2,3,4\}}, \\ W^Q_{1,\{1,2,5\}} &= Y^Q_{1,\{2,3,5\}} \oplus \bar{W}_{1,\{2,3,5\}} \oplus W^Q_{1,\{1,2,3\}}, \\ W^I_{1,\{1,2,5\}} &= W^Q_{1,\{1,2,5\}} \oplus Y^I_{1,\{2,3,5\}} \oplus W^I_{1,\{2,3,5\}} \oplus \bar{W}_{1,\{1,2,3\}}, \\ W^Q_{1,\{1,4,5\}} &= Y^Q_{1,\{3,4,5\}} \oplus \bar{W}_{1,\{3,4,5\}} \oplus W^Q_{1,\{1,3,4\}}, \\ W^I_{1,\{1,4,5\}} &= W^Q_{1,\{1,4,5\}} \oplus Y^I_{1,\{3,4,5\}} \oplus W^I_{1,\{3,4,5\}} \oplus \bar{W}_{1,\{1,3,4\}}. \end{aligned}$$

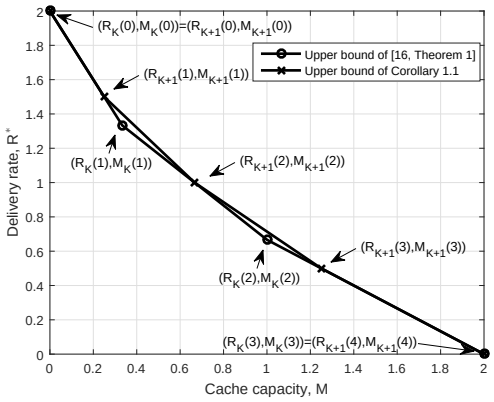

Figure 2: Rate-memory functions of the proposed scheme and the Tian-Chen scheme for $N = 2$ and $K = 3$.

Finally, user U_3 obtains subfile $W_{1,\{2,4,5\}}$, as in (35) and (36).

VI. Conclusions

In this work, we proposed a novel caching scheme that approaches the rate-memory region achieved by the Tian-Chen scheme as the number of users in the system increases, which only requires a finite field of order 2^2. Moreover, instead of relying on the existence of some valid code, we provided an explicit combinatorial construction of the caching scheme.

Appendix A
Proof of Corollary 1.2

Let us define

$$\begin{aligned} R^*_K(r) &\triangleq N\left(1 - \frac{r}{K}\right), \\ M_K(r) &\triangleq \frac{r}{K} + (N-1)\frac{r\,(r-1)}{K\,(K-1)} \end{aligned}$$

for all $r \in \{0, ..., K\}$. In a caching system with K users and N files, the worst demand rate-memory pairs, as a function of the parameter r, for the Tian-Chen scheme are given by $R_{\text{TC}}(r) = R_K(r)$ and $M_{\text{TC}}(r) = M_K(r)$, and for the strategy presented here by $R^*_{\text{TC}^+}(r) = R_{K+1}(r)$ and $M_{\text{TC}^+}(r) = M_{K+1}(r)$.

To support this proof, in Fig. 2 we compare the rate-memory functions obtained by both schemes in a particular caching system with $N = 2$ files and $K = 3$ user. We first obtain the memory points and intervals where the Tian-Chen and the proposed strategy coincide. By memory sharing between the rate-memory pairs in $(R^*_K(r), M_K(r))$ at r and $r+1$, we obtain the rate-memory pairs

$$\begin{aligned} R^*_K(r,y) &= (1-y)R^*_K(r) + yR^*_K(r+1) \\ &= N\left(1 - \frac{r}{K}\right) - y\frac{N}{K} \qquad (37) \\ M_K(r,y) &= (1-y)M_K(r) + yM_K(r+1) \\ &= \frac{r}{K} + (N-1)\frac{r\,(r-1)}{K\,(K-1)} \\ &\quad + y\left(\frac{1}{K} + 2\frac{r\,(N-1)}{K\,(K-1)}\right) \qquad (38) \end{aligned}$$

for $y \in [0,1]$ and $r = \{0,...,K-1\}$. Next, particularizing (37) and (38) at $\hat{y}(r) = \frac{K-r}{K+1}$, we obtain

$$\begin{aligned} R^*_K\left(r, \hat{y}(r)\right) &= R^*_{K+1}(r+1), \\ M_K\left(r, \hat{y}(r)\right) &= M_{K+1}(r+1) \end{aligned}$$

for $r \in \{0,...K\}$. This is, the delivery rate for the Tian-Chen and the proposed scheme coincide $R^*_{\text{TC}^+}(M) = R^*_{\text{TC}}(M)$ at $M = M_{K+1}(r+1) = \frac{r+1}{K+1}\left(1+(N-1)\frac{r}{K}\right)$ for $r = \{0,...,K\}$. Furthermore, at $r = 0$, we have $R^*_{K+1}(0) = R^*_K(0) = N$, and $M_{K+1}(0) = M_K(0) = 0$ and thus, equality holds also for the memory interval

$$0 \leq M \leq M_{K+1}(1) = \frac{1}{K+1}.$$

Similarly, we have $R^*_{K+1}(K+1) = R^*_K(K) = 0$, and $M_{K+1}(K+1) = M_K(K) = N$, and thus, the equality holds also for the memory interval

$$N \geq M \geq \frac{K}{K+1} + (N-1)\frac{K-1}{K+1}.$$

Next, we obtain the bound on the maximum rate ratio between both strategies. It can be sown that the equality $M_K(r) = M_{K+1}(r,y)$ is achieved at $y = \check{y}(r)$ with

$$\check{y}(r) = \frac{r}{K-1}\left(1 - \frac{2(N-1)+1}{K+2(N-1)r}\right)$$

for $r = \{1,...K-1\}$. Given that both rate-memory functions coincide at $M = M_{K+1}(r+1)$, that $M_{K+1}(r) \leq M_K(r) \leq M_{K+1}(r+1)$ for $r = \{1...,K-1\}$, and the convexity of the two piece-wise linear rate-memory functions, we have, see Fig. 2, that the maximum rate ratio must be found at one of the memory points $M_K(r) = M_{K+1}(r, \check{y}(r))$, $r = \{1,...,K-1\}$, this is

$$\begin{aligned} \max_{0\leq M\leq N} \frac{R^*(M)}{R^*_{\text{TC}}(M)} &= \max_{r\in\{0,...,K-1\}} \frac{R_{K+1}(r,\check{y})}{R_K(r)} \\ &= 1 + \frac{r - \check{y}(r)K}{(K-r)(K+1)} \\ &\leq 1 + \frac{1}{(K-1)(K+1)}. \end{aligned} \quad (39)$$

To obtain inequality (39), observe that $\check{y}(r)$ decreases as a function of N, and thus, as N approaches infinity, we have for $r = \{1...,K-1\}$ $\lim_{N\to\infty} \check{y}(r) = \frac{r}{K-1}\left(1-\frac{1}{r}\right)$.

REFERENCES

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[2] M. R. Korupolu, C. Plaxton, and R. Rajaraman, "Placement algorithms for hierarchical cooperative caching," *Journal of Algorithms*, vol. 38, no. 1, pp. 260 – 302, 2001.

[3] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *Multimedia Systems*, vol. 4, no. 3, pp. 112–121, 1996.

[4] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug 2015.

[5] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146–1158, Feb 2017.

[6] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," in *ITA workshop*, Feb 2015, pp. 98–107.

[7] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3923–3949, June 2017.

[8] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 836–845, April 2016.

[9] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3212–3229, June 2016.

[10] M. M. Amiri, Q. Yang, and D. Gündüz, "Coded caching for a large number of users," in *2016 IEEE Information Theory Workshop (ITW)*, Sept 2016, pp. 171–175.

[11] S. Sahraei and M. Gastpar, "K users caching two files: An improved achievable rate," in *2016 Annual Conference on Information Science and Systems (CISS)*, March 2016, pp. 620–624.

[12] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," in *IEEE ISIT*, June 2015, pp. 1696–1700.

[13] M. M. Amiri and D. Gündüz, "Fundamental limits of coded caching: Improved delivery rate-cache capacity tradeoff," *IEEE Transactions on Communications*, vol. 65, no. 2, pp. 806–815, Feb 2017.

[14] Z. Chen, P. Fan, and K. B. Letaief, "Fundamental limits of caching: improved bounds for users with small buffers," *IET Communications*, vol. 10, no. 17, pp. 2315–2318, 2016.

[15] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, Feb 2018.

[16] C. Tian and J. Chen, "Caching and delivery via interference elimination," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1548–1560, March 2018.

[17] C. Tian and K. Zhang, "From uncoded prefetching to coded prefetching in coded caching," *CoRR*, vol. abs/1704.07901, 2017. [Online]. Available: http://arxiv.org/abs/1704.07901

[18] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *IEEE ISIT*, July 2016, pp. 135–139.

[19] A. Sengupta, R. Tandon, and T. C. Clancy, "Improved approximation of storage-rate tradeoff for caching via new outer bounds," in *IEEE ISIT*, June 2015, pp. 1691–1695.

[20] C. Tian, "A note on the fundamental limits of coded caching," *CoRR*, vol. abs/1503.00010, 2015.

[21] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," in *IEEE ISIT*, June 2017, pp. 386–390.

[22] C. Y. Wang, S. S. Bidokhti, and M. Wigger, "Improved converses and gap-results for coded caching," pp. 2428–2432, June 2017.

[23] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using ruzsa-szemeredi graphs," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 1237–1241.

[24] L. Tang and A. Ramamoorthy, "Low subpacketization schemes for coded caching," in *2017 IEEE International Symposium on Information Theory (ISIT)*, June 2017, pp. 2790–2794.

[25] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., "Finite-length analysis of caching-aided coded multicasting," *IEEE Transactions on Information Theory*, vol. 62, no. 10, pp. 5524–5537, Oct 2016.

[26] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 5559–5564.

[27] J. Gómez-Vilardebó, "Fundamental Limits of Caching: Improved Bounds with Coded Prefetching," *ArXiv e-prints*, Dec. 2016.

Jesús Gómez-Vilardebó received his M.Sc. and Ph.D. degrees in Telecommunication Engineering from the Universitat Politècnica de Catalunya (UPC) in October 2003 and July 2009, respectively. In September 2005, he was granted by the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA) to obtain the Ph.D. on Signal Theory and Communications at the UPC. He is now with the CTTC holding a Research Associate position. His research interests are in the areas of information and communication theory, and their applications in multi-user communications, coded caching and information privacy.