A novel coded caching scheme with coded prefetching

Jesus Gomez-Vilardebo
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Castelldefels, Spain
jesus.gomez@cttc.es

***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 $m \geq K \log_2(N)$ in some situations, 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

The 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 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, there 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.

This work was partially supported by the Catalan Government under grant SGR2017-1479 and the Spanish Government under grant TEC2013-44591-P (INTENSYV).

The fundamental caching scheme developed in [1] was latter extended to more realistic situations. The decentralized setting was considered in [2], non-uniforms demands in [3], online coded caching in [4], and hierarchical cache network in [5], among others. In addition, new schemes pushing further the fundamental limits of caching systems have appeared in [6]–[12]. 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 [13]. Other lower bounds have appeared in [7], [14], [15].

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 [11]. 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 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 [12], 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. Finally, conclusions are drawn in Section V.

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 $N \leq K$. As 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. 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. 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 demand $\mathbf{d}$, we say that communication rate $R(\mathbf{d}, \boldsymbol{\mathcal{M}})$ 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 $W_{\mathbf{d}(k)}$. 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 worst demand rate-memory pair (R^*, M) and the worst demand rate-memory function $R^*(M)$.

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 [16, Corollari 1.1], the rate-memory pairs obtained in [10, Corollary 1], and the rate-memory pairs obtained in [11, Theorem 1]

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

with $r \in \{0, 1, ..., K\}$. The rate-memory pairs in [10, Corollary 1] include as special case the rate-memory pair obtained in [6] for $M = \frac{N}{K}$. In addition, both [16, Corollari 1.1], and [11, Theorem 1] include the rate-memory pair $(R^*, M) = \left(N - \frac{N}{K}, \frac{1}{K}\right)$ obtained in [9]. Recently, a method to obtain new rate-memory pairs has been developed in [12]. However, no explicit characterization of the achievable rate-memory pairs is provided. While the schemes in [10] and [16] only require binary operations, the scheme developed in [11], [12] makes use of codes in a finite field of order N^K in some situations. Finally, there have also been efforts to obtain lower bounds on the delivery rate. The cut-set bound was studied in [1]. A tighter lower bound was obtained in [13]. Other lower bounds have appeared in [7], [14], [15].

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

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

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

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.

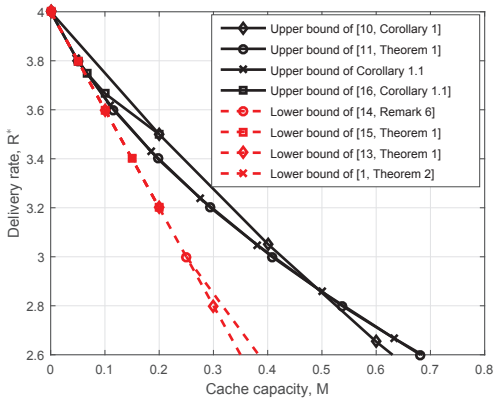

Figure 1: Rate-memory functions of the proposed scheme compared with existing schemes and lower bounds in the literature for $N = 4$ and $K = 20$.

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 $(R^*, M) = (R^*_{\text{TC}^+}, M_{\text{TC}^+})$, with

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

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

Remark 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 (1) for $K+1$ users. The following corollary compares both rate-memory functions. The proof is omitted due to space limitations it can be found in [17].

Corollary 1.2. Let $R^*_{\text{TC}^+}(M)$ be the worst demand rate-memory function achieved by memory sharing between the rate-memory pairs in Corollary 1.1, and let $R^*_{\text{TC}}(M)$ 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 $\frac{R^*_{\text{TC}^+}(M)}{R^*_{\text{TC}}(M)} \leq 1 + \frac{1}{(K-1)(K+1)}$.

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 [10, Corollary 1], [16, Corollary 1.1] and [11, Theorem 1]. In addition, we depict for comparison, the cut set lower bound [1, Theorem 2], the information-theoretical lower bound obtained in [13, Theorem 1], and the lower bounds recently appeared in [15, Theorem 1] and [14, Remark 6]. Observe that, for the special situation considered here, the new proposed scheme obtains a rate-memory function overlapping almost exactly the rate-memory function in [11], [12]. Moreover, the strategy presented here only requires a finite field of order 2^2. Instead, to obtain the rate-memory pair associated to $r = 11$, the Tian-Chen scheme requires a finite field of at least order 10^5.

IV. Proposed caching scheme

We prove Theorem 1 in the following subsections. First, we describe a caching scheme that achieves the Tian-Chen rate-memory pairs only if all files are requested by some user. Then, we show that the same scheme can be applied to any demand if configured for $K+1$ caches instead of K.

A. 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, $W_{f,\mathcal{S}}$, one for each subset $\mathcal{S}$ of r 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 user U_k. User U_k caches the binary sum of the N subfiles associated to $\mathcal{S}$

$$Z_{\mathcal{S}}=\bigoplus_{f\in\mathcal{F}}W_{f,\mathcal{S}} \tag{2}$$

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

$$Z_{f,\mathcal{S}^-}=\bigoplus_{s\in\mathcal{K}\backslash\mathcal{S}^-}W_{f,\mathcal{S}^-\cup s} \tag{3}$$

for every file $f\in\mathcal{F}$, and every subset $\mathcal{S}^-\in\mathcal{T}_k(r-1)$. It can be shown, see [17], that user U_k only needs to cache the coded subfiles $Z_{f,\mathcal{S}^-}$ in (3) for all files $f\in\mathcal{F}\backslash g$ and all sets $\mathcal{S}^-\in\mathcal{T}_k(r-1)$ satisfying $l\notin\mathcal{S}^-$, for an arbitrary file g, and user l. The total number of coded subfiles $Z_{f,\mathcal{S}^-}$ cached at user U_k is then $(N-1)\binom{K-2}{r-2}$.

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, selects one user leader $u_f\in\mathcal{K}(f)$ for each file $f\in\mathcal{F}$, and then broadcasts the coded subfile $Y_{f,\mathcal{S}}=\left\langle Y^I_{f,\mathcal{S}},Y^Q_{f,\mathcal{S}}\right\rangle$

$$Y^I_{f,\mathcal{S}} = W^I_{f,\mathcal{S}}\oplus\bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)}\bar{W}_{f,u_f\cup\{\mathcal{S}\backslash s\}}, \tag{4}$$

$$Y^Q_{f,\mathcal{S}} = \bar{W}_{f,\mathcal{S}}\oplus\bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)}W^Q_{f,u_f\cup\{\mathcal{S}\backslash s\}} \tag{5}$$

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 $\mathcal{S}\in\mathcal{T}(r)$ with $u_f\notin\mathcal{S}$ for each file, we need $N\binom{K-1}{r}$ subfile length broadcast transmissions, thus, requiring the rate $R=N\binom{K-1}{r}/\binom{K}{r}=N\left(1-\frac{r}{K}\right)$.

Next we describe the decoding process. Each 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 $\mathcal{S}\in\mathcal{T}_k(r)$. These 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 $\mathcal{S}\in\bar{\mathcal{T}}_k(r)$, with $\bar{\mathcal{T}}_k(r)=\{\mathcal{S}\in\mathcal{T}(r):k\notin\mathcal{S}\}$.

3) Decoding of requested subfiles coded in the cache of user U_k: Recall that for any set $\mathcal{S}\in\mathcal{T}_k(r)$, user U_k caches $Z_{\mathcal{S}}=W_{\mathbf{d}(k),\mathcal{S}}\oplus\bigoplus_{f\in\mathcal{F}\backslash\mathbf{d}(k)}W_{f,\mathcal{S}}$ 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 $f\in\mathcal{F}\backslash\mathbf{d}(k)$, and then computing

$$W_{\mathbf{d}(k),\mathcal{S}}=Z_{\mathcal{S}}\oplus\bigoplus_{f\in\mathcal{F}\backslash\mathbf{d}(k)}W_{f,\mathcal{S}}. \tag{8}$$

The decoding of these interfering subfiles is divided into r consecutive phases $p=0,...,r-1$. At phase p, for every file $f\in\mathcal{F}\backslash\mathbf{d}(k)$, 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 $W_{f,\mathcal{S}}$ as $\left\langle W^I_{f,\mathcal{S}},W^Q_{f,\mathcal{S}}\right\rangle=\left\langle Y^I_{f,\mathcal{S}},Y^Q_{f,\mathcal{S}}\oplus Y^I_{f,\mathcal{S}}\right\rangle$. 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 be obtained as, $W_{f,\mathcal{S}}=\left\langle W^I_{f,\mathcal{S}},W^Q_{f,\mathcal{S}}\right\rangle$

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

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

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

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 $|\{\mathcal{S}\backslash u\}\cap\mathcal{K}(f)|=p-1$, and thus from decoding phase $p-1$, user U_k already has $W_{f,\{\mathcal{S}\backslash u\}\cup v}$ for all $v\in\bar{\mathcal{K}}(f)\backslash\{\mathcal{S}\backslash u\}$. This allows user U_k to compute

$$C_{f,\mathcal{S}\backslash u}=Z_{f,\mathcal{S}\backslash u}\oplus\bigoplus_{v\in\bar{\mathcal{K}}(f)\backslash\{\mathcal{S}\backslash u\}}W_{f,\{\mathcal{S}\backslash u\}\cup v} \tag{11}$$

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

$$\Upsilon_{f,\mathcal{S}\backslash u_f}=\bigoplus_{s\in\mathcal{K}(f)\backslash\mathcal{S}}Y_{f,\{\mathcal{S}\backslash u_f\}\cup s}. \tag{12}$$

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

$$\Upsilon^I_{f,\mathcal{S}\backslash u_f} = C^I_{f,\mathcal{S}\backslash u_f} \oplus W^I_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{K}(f)\backslash\mathcal{S}} \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{u\in\{\mathcal{S}\backslash u_f\}\cap\mathcal{K}(f)} \bar{C}_{f,\mathcal{S}\backslash u} \oplus \bar{W}_{f,\mathcal{S}} \quad (6)$$

$$\Upsilon^Q_{f,\mathcal{S}\backslash u_f} = \bar{C}_{f,\mathcal{S}\backslash u_f} \oplus \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{K}(f)\backslash\mathcal{S}} W^Q_{f,\mathcal{S}} \oplus \bigoplus_{u\in\{\mathcal{S}\backslash u_f\}\cap\mathcal{K}(f)} C^Q_{f,\mathcal{S}\backslash u} \oplus W^Q_{f,\mathcal{S}}. \quad (7)$$

$\Upsilon^I_{f,\mathcal{S}\backslash u_f} \oplus \Upsilon^Q_{f,\mathcal{S}\backslash u_f}$, and $\bar{C}_{f,\mathcal{S}\backslash u} \triangleq C^I_{f,\mathcal{S}\backslash u} \oplus C^Q_{f,\mathcal{S}\backslash u}$. Then, user U_k computes

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

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

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

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

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

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

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

from which we can write

$$\bigoplus_{s\in\mathcal{K}(f)\backslash\mathcal{S}} W_{f,\{\mathcal{S}\backslash u\}\cup s} = C_{f,\mathcal{S}\backslash u} \oplus W_{f,\mathcal{S}}. \quad (13)$$

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

$$\begin{aligned} &= W^I_{f,\{\mathcal{S}\backslash u_f\}\cup s} \oplus \bigoplus_{u\in\{\{\mathcal{S}\backslash u_f\}\cup s\}\cap\mathcal{K}(f)} \bar{W}_{f,u_f\cup\{\{\mathcal{S}\backslash u_f\}\cup s\}\backslash u} \\ &= W^I_{f,\{\mathcal{S}\backslash u_f\}\cup s} \oplus \bar{W}_{f,\mathcal{S}} \oplus \bigoplus_{u\in\{\mathcal{S}\backslash u_f\}\cap\mathcal{K}(f)} \bar{W}_{f,\{\mathcal{S}\backslash u\}\cup s} \quad (14) \end{aligned}$$

Similarly, we can write $Y^Q_{f,\{\mathcal{S}\backslash u_f\}\cup s}$

$$= \bar{W}_{f,\{\mathcal{S}\backslash u_f\}\cup s} \oplus W^Q_{f,\mathcal{S}} \oplus \bigoplus_{u\in\{\mathcal{S}\backslash u_f\}\cap\mathcal{K}(f)} W^Q_{f,\{\mathcal{S}\backslash u\}\cup s}. \quad (15)$$

Now, it follows from (13) and (14) that $\Upsilon^I_{f,\mathcal{S}\backslash u_f} = \bigoplus_{s\in\mathcal{K}(f)\backslash\mathcal{S}} Y^I_{f,\{\mathcal{S}\backslash u_f\}\cup s}$ can be written as in (6), and, from (13) and (15) that $\Upsilon^Q_{f,\mathcal{S}\backslash u_f} = \bigoplus_{s\in\mathcal{K}(f)\backslash\mathcal{S}} Y^Q_{f,\{\mathcal{S}\backslash u_f\}\cup s}$ can be written as in (7). Next observe that

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

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

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

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

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

If $|\mathcal{K}(f)\backslash u_f|$ 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

$$W^I_{f,\mathcal{S}} = Y^I_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)} \bar{W}_{f,u_f\cup\mathcal{S}\backslash s}, \quad (16)$$

$$W^Q_{f,\mathcal{S}} = W^I_{f,\mathcal{S}} \oplus Y^Q_{f,\mathcal{S}} \oplus \bigoplus_{s\in\mathcal{S}\cap\mathcal{K}(f)} W^Q_{f,u_f\cup\mathcal{S}\backslash s}. \quad (17)$$

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

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

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 already has all the requested subfiles coded in its own cache, in particular $W_{f,\mathcal{S}\cup k\setminus s}$ 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 $W^Q_{f,\mathcal{S}}$

$$= Y^Q_{f,\mathcal{S}\cup k\setminus u_f} \oplus \bar{W}_{f,\mathcal{S}\cup k\setminus u_f} \oplus \bigoplus_{s\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} W^Q_{f,\mathcal{S}\cup k\setminus s} \quad (18)$$

and $W^I_{f,\mathcal{S}}$

$$= W^Q_{f,\mathcal{S}} \oplus Y^I_{f,\mathcal{S}\cup k\setminus u_f} \oplus W^I_{f,\mathcal{S}\cup k\setminus u_f} \oplus \bigoplus_{s\in\{\mathcal{S}\setminus u_f\}\cap\mathcal{K}(f)} \bar{W}_{f,\mathcal{S}\cup k\setminus s} \quad (19)$$

for all sets $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ with $u_f \in \mathcal{S}$. Now user U_k can mimic the user leader u_f and obtain the remaining subfiles, i.e. $W_{f,\mathcal{S}}$ for all sets $\mathcal{S} \in \bar{\mathcal{T}}_k(r)$ satisfying $u_f \notin \mathcal{S}$ from (16) and (17).

B. Extension to arbitrary requested files

In previous subsection, we required all files to be requested. In this section, we show that the same scheme can be applied to any request if configured for $K+1$ 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, we add the $K+1$th "virtual" user to the system i.e. $\mathcal{K} = \{1, ..., K+1\}$ and assume that the $K+1$th user requests any of the files i.e. $\mathbf{d}(K+1) \in \mathcal{F}$. Then, we apply the delivery strategy as described in previous subsections. In that case, every user obtains their desired file, with a rate

$$R = N\left(1 - \frac{r}{K+1}\right). \quad (20)$$

Instead, if some files are not requested, the additional user requests the binary sum of all files not requested, i.e. $W_0 \triangleq \bigoplus_{f\in\mathcal{F}\setminus\mathcal{N}_e(\mathbf{d})} W_f$ where $\mathcal{N}_e(\mathbf{d})$ denotes the set of files requested. Now, we can replace the set of files in the system $\mathcal{F}$ 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 subfiles not requested as $W_{0,\mathcal{S}} \triangleq \bigoplus_{f\in\mathcal{F}\setminus\mathcal{N}_e(\mathbf{d})} W_{f,\mathcal{S}}$. Similarly, users can consider the cached subfiles in (2), as if the set of file was $\mathcal{F}_0(\mathbf{d})$, since

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

and, obtain the coded subfiles in (3) associated to file W_0 as

$$\begin{aligned} Z_{0,\mathcal{S}^-} &= \bigoplus_{f\in\mathcal{F}\setminus\mathcal{N}_e(\mathbf{d})} Z_{f,\mathcal{S}^-} \\ &= \bigoplus_{s\in\mathcal{K}\setminus\mathcal{S}^-} \bigoplus_{f\in\mathcal{F}\setminus\mathcal{N}_e(\mathbf{d})} W_{f,\mathcal{S}^-\cup s} \\ &= \bigoplus_{s\in\mathcal{K}\setminus\mathcal{S}^-} W_{0,\mathcal{S}.} \end{aligned}$$

Now, since we have replaced the set of files $\mathcal{F}$ by $\mathcal{F}_0(\mathbf{d})$, and all files in $\mathcal{F}_0(\mathbf{d})$ are requested, we can apply the delivery strategy as described in previous subsection, with a rate

$$R = (N_e(\mathbf{d})+1)\left(1 - \frac{r}{K+1}\right). \quad (21)$$

Combining (20) and (21), we can write the rate as presented in Theorem 1.

V. 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.

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] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug 2015.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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

[13] 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.

[14] 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.

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

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

[17] ——, "A novel centralized coded caching scheme with coded prfetching," Dec. 2017. [Online]. Available: http://www.cttc.es/publication/coded-caching/