Successive Refinement to Caching for Dynamic Requests

Pinar Sen
University of California, San Diego
San Diego, USA
Email: psen@ucsd.edu

Michael Gastpar
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
Email: michael.gastpar@epfl.ch

Young-Han Kim
University of California, San Diego
San Diego, USA
Email: yhk@ucsd.edu

***Abstract*—In the celebrated coded caching problem studied by Maddah-Ali and Niesen, the peak-traffic network load is to be reduced by first caching some information about contents into individual memories of end users during the off-peak hours and then upon user requests broadcasting some other information about the contents, which, combined with cached information, can let each user recover their requested content. Thus, information-theoretic studies of coded caching involve the optimal tradeoff among communication rates for the two phases of cache placement and content delivery, and the optimal construction of codes for cache placement and content delivery. In order to allow better utilization of network resources, this paper introduces a new caching model in which user requests can arise at any point of time during the cache placement phase, and proposes a successive refinement approach as an answer to this *dynamic* caching problem. For uniformly random file requests, the optimal tradeoff among average-case delivery rates are characterized when the cache rate is above a well-defined threshold. For arbitrary file requests, a successive caching algorithm is developed to simultaneously reduce worst-case delivery rates at every request time, which are uniformly within a constant multiplicative factor of their respective optima.**

I. Introduction

Due to the ever growing number of devices, networks encounter heavy traffic during the peak-traffic hours of the day. One approach to shifting the network traffic to off-peak hours is to cache partial information about the contents on the local memories of end users during the off-peak hours for potential future use, referred to as cache placement. Once the user requests are revealed to the server, possibly during the peak-traffic hours, server broadcasts some other information about the contents, referred to as content delivery, so that every user can recover its requested content by combining this new information with its cache.

One line of research took coding theoretic approaches to propose practical and close-to-optimal codes for cache placement and content delivery. From earlier studies, [1] concentrated on the optimization of cache placement when the content delivery is fixed to orthogonal unicast or multicast transmissions, whereas [2] studied the optimization of the content delivery for fixed cache contents. A more recent work by Madah-Ali and Niesen [3] discussed the joint optimization of both and presented a coding scheme that can achieve the optimal tradeoff among communication rates for the two phases of cache placement and content delivery upto a constant multiplicative factor. In this pioneering work, each file is split into subfiles, where a set of properly chosen subfiles is cached at user devices and then upon request, a set of linearly encoded subfiles is broadcasted as the delivery content. Building on this celebrated work, similar results were provided for heterogeneous cache sizes in [4], [5], for nonuniform file popularity in [6], for minimizing the average delivery rate instead of the worst-case in [6], and for correlated contents in [7].

Another line of research took information theoretic approaches to provide single letter characterizations for the optimal trade-offs among cache placement rates and content delivery rates considering the statistics of the user requests. [8] formulated the single user caching problem through its similarity to Gray-Wyner network [9]. Extensions of the theoretical analysis in [8] to multi-user caching problems can be found in [10], [11]. Building on rate distortion theory, the counterparts of those caching problems for lossy reconstruction were also discussed in [12]–[15].

In all these versions of the caching problem, cache placement and content delivery are considered as two separate phases that do not intersect. To allow better utilization of the network resources, in this paper, we present a new caching model in which user requests can arise at any point of time possibly interrupting the cache placement, contrary to the temporal static requests in the traditional model. Inspired by [16], we propose a successive refinement approach to cache placement to address this dynamic nature. First, taking an information theoretic approach for uniformly random file requests in a single user network, we characterize the optimal trade-offs among average delivery rates when the cache rate is above a well-defined threshold. Second, taking a coding theoretic approach for arbitrary file requests in a network of multiple users, we propose a successive caching algorithm to simultaneously reduce the worst-case delivery rates at every request time, which are uniformly within a constant multiplicative factor of their respective optima. Our algorithm is built upon similar ideas with the caching scheme by Maddah-Ali and Niesen, and achieves the same performance with its static counterpart simultaneously at every time index.

The rest of the paper is organized as follows. Section II formulates the caching problem for dynamic requests. Section III and Section IV present the results for the expected delivery rates and for the worst-case delivery rates, respectively.

We adapt the notation in [17], [18]. The set of integers $\{1, 2, \ldots, n\}$ is denoted by $[n]$. A length-n sequence (vector) is denoted by $x^n = (x_1, x_2, \ldots, x_n) \in \mathcal{X}^n$. Upper case letters $X, Y, \ldots$ denote random variables. For a positive integer t, Σ_t denotes the set of all permutations over $[t]$. For a permutation $\sigma \in \Sigma_t$ and a set $\mathcal{S}$ of size t, $\sigma(\mathcal{S})$ denotes the sequence obtained by permuting the naturally ordered elements of $\mathcal{S}$ according to σ.

II. Problem Formulation

Consider a server with a fixed number N of files $(X_1^n, X_2^n, \ldots, X_N^n)$ drawn i.i.d. from $p(x_1, x_2, \ldots, x_N)$ and K users, each with cache rate of nC bits.

Each user successively caches some information about the contents in T steps of increments. At step $t \in [T]$, additional information $L_t^{(k)}$ of rate C_t is stored in the local cache of user $k \in [K]$, where $C_1 + C_2 + \cdots + C_T = C$. A request can arise at any time point $t \in [T]$, the knowledge of which server does not have a priori. For a request arising at $t \in [T]$ denoted by the request vector $\mathbf{d} = [d_1 \ d_2 \ \ldots \ d_K] \in [N]^K$ where d_k corresponds to the request of user $k \in [K]$, server broadcasts some other information about the contents $M_{\mathbf{d},t}$ of rate $R_{\mathbf{d},t}$ to all users so that each user is able to recover the rest of its desired file.

An $\left(n, (nC_t, nR_{\mathbf{d},t})_{\mathbf{d}\in[N]^K, t\in[T]}\right)$ T-successive caching scheme for dynamic requests consists of

- KT caching functions where
$$\phi_t^{(k)} : \{0,1\}^{nN} \to \{0,1\}^{nC_t}$$
maps the files $(X_1^n, X_2^n, \ldots, X_N^n)$ into a cache content
$$L_t^{(k)} := \phi_t^{(k)}(X_1^n, X_2^n, \ldots, X_N^n)$$
to be placed at user $k \in [K]$ during step t of the successive cache placement phase for $t \in [T]$,
- $N^K T$ encoding functions, where
$$\psi_{\mathbf{d},t} : \{0,1\}^{nN} \to \{0,1\}^{nR_{\mathbf{d},t}}$$
maps the files $(X_1^n, X_2^n, \ldots, X_N^n)$ to the delivery content
$$M_{\mathbf{d},t} := \psi_{\mathbf{d},t}(X_1^n, X_2^n, \ldots, X_N^n)$$
corresponding to the request vector $\mathbf{d} = [d_1 \ d_2 \ \ldots \ d_K] \in [N]^K$ received by the server at time index $t \in [T]$,
- $KN^K T$ decoding functions, where
$$\mu_{\mathbf{d},t}^{(k)} : \{0,1\}^{n(\sum_{i=1}^t C_i)} \times \{0,1\}^{nR_{\mathbf{d},t}} \to \{0,1\}^n$$
maps the cache contents placed until time index $t \in [T]$ and the delivery contents into an estimate
$$\hat{X}_{d_k,t}^n := \mu_{\mathbf{d},t}^{(k)}\big((L_i^{(k)} : i \in [t]), M_{\mathbf{d},t}\big)$$
of the requested file $X_{d_k}^n$ by user $k \in [K]$ when the request vector $\mathbf{d} \in [N]^K$ is received by the server at time index t.

The probability of error is defined as

$$P_e^{(n)} := \mathsf{P}(\hat{X}_{d_k,t}^n \neq X_{d_k}^n \text{ for some } t \in [T], k \in [K], d_k \in [N]).$$

Given a cache rate tuple $(C_1, C_2, \ldots, C_T)$, a delivery rate tuple $(R_{\mathbf{d},t})_{\mathbf{d}\in[N]^K, t\in[T]}$ is said to be *achievable for dynamic requests* if there exists an $\left(n, (nC_t, nR_{\mathbf{d},t})_{\mathbf{d}\in[N]^K, t\in[T]}\right)$ T-successive caching scheme for dynamic requests with $\lim_{n\to\infty} P_e^{(n)} = 0$.

III. Average Case

One of the commonly used performance criteria is the *expected* delivery rate in which the expectation is taken with respect to the random request vector $\mathbf{d}$ uniformly distributed over $[N]^K$. Given a tuple of cache rates $(C_1, C_2, \ldots, C_T)$, a rate tuple $(R_t)_{t\in[T]}$ is said to be *average-case achievable for dynamic requests* if there exists a delivery rate tuple $(R_{\mathbf{d},t})_{\mathbf{d}\in[N]^K, t\in[T]}$ achievable for dynamic requests such that

$$\frac{1}{N^K} \sum_{\mathbf{d}\in[N]^K} R_{\mathbf{d},t} \leq R_t$$

for every $t \in [T]$.

Our goal in this section is to characterize the set of average-case achievable rate tuples for dynamic requests. We follow an information theoretic approach. For simplicity, we consider a single user ($K = 1$) and a pair of files ($N = 2$) X_1^n and X_2^n that are drawn i.i.d. from the pmf $p(x_1, x_2)$ over a finite alphabet. We assume that one of the files is to be requested by the user randomly and equally likely over two possible time index ($T = 2$). As illustrated in Fig. 1, the request is dynamic in time in the sense that it is received either at $t = 1$ or at $t = 2$, which correspond to the time right after placing nC_1 and $n(C_1 + C_2)$ bits of cache, respectively. Depending on the *random* request $d \in [2]$ received at time index t, the server transmits $nR_{d,t}$ bits to deliver the remaining part of the requested file X_d^n.

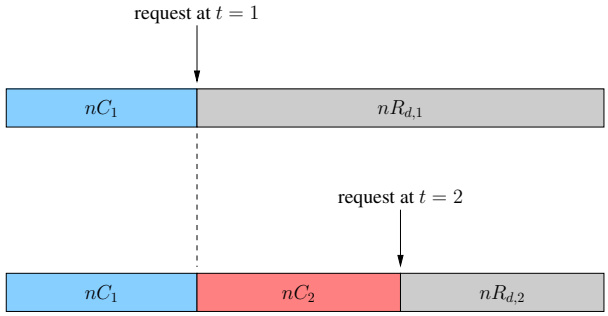

Fig. 1: Dynamic requests in the caching problem

To characterize the average-case achievable rate pairs for dynamic requests, we start with *static requests*, which corresponds to $T = 1$ in our setup. Define the optimal average-case delivery rate function for static requests as

$$R_{\text{avg}}^*(C) := \min\{R : R \text{ is average-case achievable for static requests}\}, \quad (1)$$

where the achievability for static requests is defined similarly by letting $T = 1$. This function captures the tradeoff between the utilized cache memory and the average delivery rate for

the traditional caching problem and characterized by Wang, Lim, and Gastpar [8], [10] as follows.

Proposition 1 ([8, Proposition 3]): The optimal average-case delivery rate function for static requests is

$$R^*_{\text{avg}}(C) = \min_{\substack{p(w|x_1,x_2):\\ I(X_1,X_2;W)\le C}} \frac{H(X_1|W)+H(X_2|W)}{2} = \frac{H(X_1,X_2)-C}{2} + \frac{1}{2}\min_{\substack{p(w|x_1,x_2):\\ I(X_1,X_2;W)=C}} I(X_1;X_2|W). \tag{2}$$

In particular, for $C \ge C(X_1;X_2)$

$$R^*_{\text{avg}}(C) = \frac{H(X_1,X_2)-C}{2}.$$

Returning back to our problem of dynamic requests with $T = 2$, the question is whether it is possible to achieve $(R_1,R_2) = \big(R^*_{\text{avg}}(C_1), R^*_{\text{avg}}(C_1+C_2)\big)$. Note that the problem is not trivial since a successive caching scheme that is optimal at the first and the second intermediate step may not achieve $R^*_{\text{avg}}(C_1+C_2)$, which is obtained by optimizing over two steps combined. We next present a sufficient condition on the cache rates to simultaneously achieve these lower bounds.

Theorem 1: Given $C_1 > 0$, let W^* denote the random variable defined by the conditional pmf $p(w^*|x_1,x_2)$ that achieves $R^*_{\text{avg}}(C_1)$ in (2) and let $C(X_1;X_2|W^*)$ be the conditional Wyner common information defined [19] as

$$C(X_1;X_2|W^*) := \min_{\substack{p(v|x_1,x_2,w^*)\\ X_1-(W^*,V)-X_2}} I(X_1,X_2;V|W^*). \tag{3}$$

For every $C_1 > 0$ and $C_2 \ge C(X_1;X_2|W^*)$, a rate pair (R_1,R_2) is average-case achievable for dynamic requests if

$$R_1 \ge R^*_{\text{avg}}(C_1), \tag{4a}$$
$$R_2 \ge R^*_{\text{avg}}(C_1+C_2). \tag{4b}$$

Conversely, given any $C_1, C_2 > 0$, if a rate pair (R_1,R_2) is average-case achievable for dynamic requests, then it must satisfy (4).

Theorem 1 implies that if the user is equipped with sufficiently large memory left for the cache update, then the problem of dynamic requests can be handled as well as two separate problems of static requests. We next relax the sufficient condition on the cache rate for the refinement, C_2, and put the burden on the amount of cache placed at the first step.

Corollary 1: For every $C_1 \ge C(X_1;X_2)$ and $C_2 \ge 0$, a rate pair (R_1,R_2) is average-case achievable for dynamic requests if and only if it satisfies (4).

Proof: The condition in Theorem 1 depends on the behavior of W^* that achieves $R^*_{\text{avg}}(C_1)$. If $C_1 \ge C(X_1;X_2)$, we know by Proposition 1 that

$$R^*_{\text{avg}}(C_1) = \frac{H(X_1,X_2)-C_1}{2}$$

and is achieved by a conditional pmf $p_{W^*|X_1,X_2}$ that satisfies $I(X_1;X_2|W^*) = 0$. Such a conditional pmf results in

$$C(X_1;X_2|W^*) = 0, \tag{5}$$

which proves the corollary combined with Theorem 1. To see (5), let $p(w|x_1,x_2)$ be a conditional pmf with $X_1 \to W \to X_2$ forming a Markov chain. Then, the conditional Wyner common information is zero by definition since

$$C(X_1;X_2|W) = \min_{\substack{p_{V|W,X_1,X_2}(v|w,x_1,x_2)\\ I(X_1;X_2|W,V)=0}} I(X_1,X_2;V|W) = 0$$

is achieved by letting $V = \emptyset$. ∎

For independent files, we have the full characterization of the average-case achievable rate pairs for dynamic requests.

Example 1 (Independent files): If $p(x_1,x_2) = p(x_1)p(x_2)$, then the Wyner common information between X_1 and X_2 is $C(X_1;X_2) = 0$. Therefore, by Corollary 1, for *every* given $C_1, C_2 \ge 0$, a rate pair (R_1,R_2) is average-case achievable for dynamic requests if and only if it satisfies (4). For a given memory pair (C_1,C_2), the set of average-case achievable rate pairs for dynamic requests is demonstrated in Fig. 2.

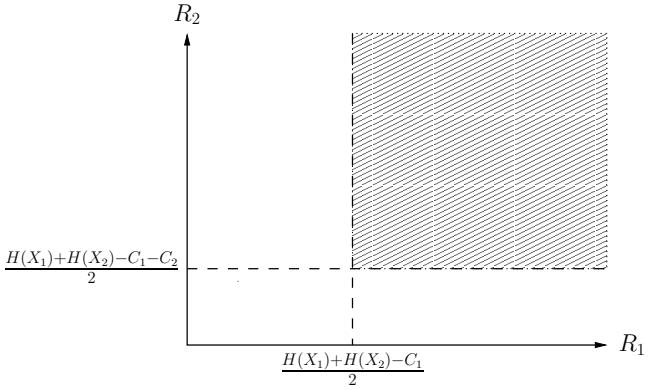

Fig. 2: Average-case achievable rate pairs for dynamic requests of independent files

We continue our discussion with the worst case delivery rates instead of the average.

IV. Worst Case

Another commonly used performance criterion is the *worst-case delivery rate*, in which the caching strategy must be designed considering every possible request vector $\mathbf{d} = [d_1 \;\; d_2 \;\; \dots d_K] \in [N]^K$. Given a tuple of cache rates $(C_1,C_2,\dots,C_T)$, a rate tuple $(R_1,R_2,\dots,R_T)$ is said to be *worst-case achievable for dynamic requests* if there exists a delivery rate tuple $(R_{\mathbf{d},t})_{\mathbf{d}\in[N]^K, t\in[T]}$ achievable for dynamic requests such that

$$\max_{\mathbf{d}\in[N]^K} R_{\mathbf{d},t} \le R_t$$

for every $t \in [T]$.

In this section, for the simplicity of the notation, we denote the file X^n as $\mathbf{X}$. We assume that the file tuple

$(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N)$ is i.i.d. $\mathrm{Bern}(1/2)$. Our goal is to characterize the set of worst-case achievable rate tuples for dynamic requests and provide a successive caching algorithm to simultaneously reduce the worst-case delivery rates. Similar to the average-case, we start with revisiting the static request problem $(T = 1)$ from a coding theoretic perspective studied by Maddah Ali and Niesen [3]. We then generalize this result for the dynamic requests. Let $R_{\text{worst}}(C)$ be the lower convex envelope of

$$K\Big(1 - \frac{C}{N}\Big) \min\Big\{\frac{1}{1 + KC/N}, \frac{N}{K}\Big\}, \tag{6}$$

for $C \in \{0, N/K, 2N/K, \ldots, N\}$, as demonstrated in Fig 3 for $(K, N) = (5, 2)$. Define the optimal worst-case delivery

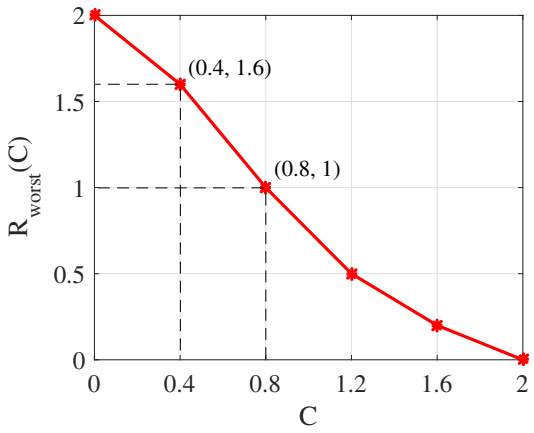

Fig. 3: Demonstration of the function $R_{\text{worst}}(C)$

rate function for static requests as

$$R^*_{\text{worst}}(C) := \min\{R : R \text{ is worst-case achievable for static requests}\}.$$

The following result presents an upper bound on the function $R^*_{\text{worst}}(C)$ and characterizes the gap between them. The tightened gap is due to the improved lower bound in [20].

Proposition 2 ([3], [20]): For N files and K users each with cache of size $C \le N$,

$$1 \le \frac{R_{\text{worst}}(C)}{R^*_{\text{worst}}(C)} \le 4.$$

We are now ready to present the main result of this section.

Theorem 2: For N files, K users, and given a tuple of cache rates $(C_1, C_2, \ldots, C_T)$, a rate tuple $(R_1, R_2, \ldots, R_T)$ is worst-case achievable for dynamic requests if

$$R_t \ge R_{\text{worst}}(C_1 + C_2 + \cdots + C_t) \tag{7}$$

for every $t \in [T]$. Conversely, given a tuple of cache memory sizes $(C_1, C_2, \ldots, C_T)$, if a rate tuple $(R_1, R_2, \ldots, R_T)$ is worst-case achievable for dynamic requests, then it must satisfy

$$R_t \ge \frac{R_{\text{worst}}(C_1 + C_2 + \cdots + C_t)}{4} \tag{8}$$

for every $t \in [T]$.

We next present a proof sketch over an example to illustrate the achievability for the corner points of $R_{\text{worst}}(C)$ that corresponds to $C_t = N/K$ for every $t \in [T]$. It is easy to generalize the arguments to an arbitrary set of points on the curve $R_{\text{worst}}(C)$ by using memory sharing arguments.

Proof Sketch: We provide the key ideas in the proof using an illustrative example with a network of $K = 5$ users and a server of $N = 2$ files. For $T = 2$, suppose that cache rates successively utilized at the users are given as $(C_1, C_2) = (0.4, 0.4)$. Our goal is to show that the rate pair $(R_1, R_2) = (1.6, 1)$ pointed in Fig. 3 is worst-case achievable for dynamic requests. In the first step of the cache placement, file $\mathbf{X}_j$, $j \in [N]$, is split into K disjoint subfiles of equal size, which are labeled as $(\mathbf{X}_j)_{(i_1)}, i_1 \in [K]$, where the subfile $(\mathbf{X}_j)_{(i_1)}$ is cached at user i_1. This corresponds to user $k \in [K]$ storing the collection of subfiles

$$L_1^{(k)} = \big((\mathbf{X}_1)_{(k)}, (\mathbf{X}_2)_{(k)}, \ldots, (\mathbf{X}_N)_{(k)}\big)$$

in its cache, which is of rate N/K. For example, user 2 at step $t = 1$ stores $(\mathbf{X}_1)_{(2)}$ and $(\mathbf{X}_2)_{(2)}$, each of which is of rate 0.2 resulting in a total cache rate of 0.4.

In the second step, every subfile $(\mathbf{X}_j)_{(i_1)}, j \in [N], i_1 \in [K]$, is further split into $(K-1)$ disjoint subfiles of equal size, which are labeled as $(\mathbf{X}_j)_{(i_1,i_2)}, i_2 \in [K] \setminus \{i_1\}$, where the subfile $(\mathbf{X}_j)_{(i_1,i_2)}$ is cached at user i_2 in the second step. This corresponds to user $k \in [K]$ adding the collection of subfiles

$$L_2^{(k)} = \big((\mathbf{X}_j)_{(i_1,k)} : i_1 \in [K] \setminus \{k\}, j \in [N]\big)$$

to its cache at step $t = 2$, which is of rate $N\frac{K-1}{K(K-1)} = N/K$. For example, user 2 at step $t = 2$ stores $(\mathbf{X}_j)_{(1,2)}$, $(\mathbf{X}_j)_{(3,2)}$, $(\mathbf{X}_j)_{(4,2)}$, and $(\mathbf{X}_j)_{(5,2)}$ for $j \in \{1, 2\}$, each of which is of rate $1/20$ resulting in a total cache rate of 0.4. Partition of file $\mathbf{X}_j$ for this two-step successive caching strategy is demonstrated in Fig. 4.

For the content delivery phase, we propose two schemes, depending on whether t satisfies

$$\frac{N}{K} \le \frac{1}{1 + K\sum_{i=1}^{t} C_i/N} = \frac{1}{1+t}. \tag{9}$$

Suppose that the request vector $\mathbf{d} = [1\ 2\ 1\ 1\ 2]$ is received at step $t = 1$, for which the condition in (9) holds. In this case, for every file $\mathbf{X}_j, j \in [N]$, server first chooses a user u that requests file $\mathbf{X}_j$ (i.e., $d_u = j$) and then for every subset $\mathcal{S} \subset [K]$ such that $u \in \mathcal{S}$ and $|\mathcal{S}| = t + 1$, and for every permutation $\sigma \in \Sigma_t$ broadcasts the linear combination

$$\bigoplus_{s \in \mathcal{S}} (X_j^n)_{\sigma(\mathcal{S}\setminus\{s\})}. \tag{10}$$

Every such linear combination has the rate of $\prod_{r=1}^{t} \frac{1}{K-r+1}$, resulting in the total delivery rate of

$$\frac{N\binom{K-1}{t}t!}{K(K-1)\cdots(K-t+1)} = N\frac{K-t}{K} = R_{\text{worst}}(tN/K)$$

when combined. For example, for the file $\mathbf{X}_1$, server chooses $u = 1$ and broadcasts linear combinations of $(\mathbf{X}_1)_1 \oplus (\mathbf{X}_1)_2$,

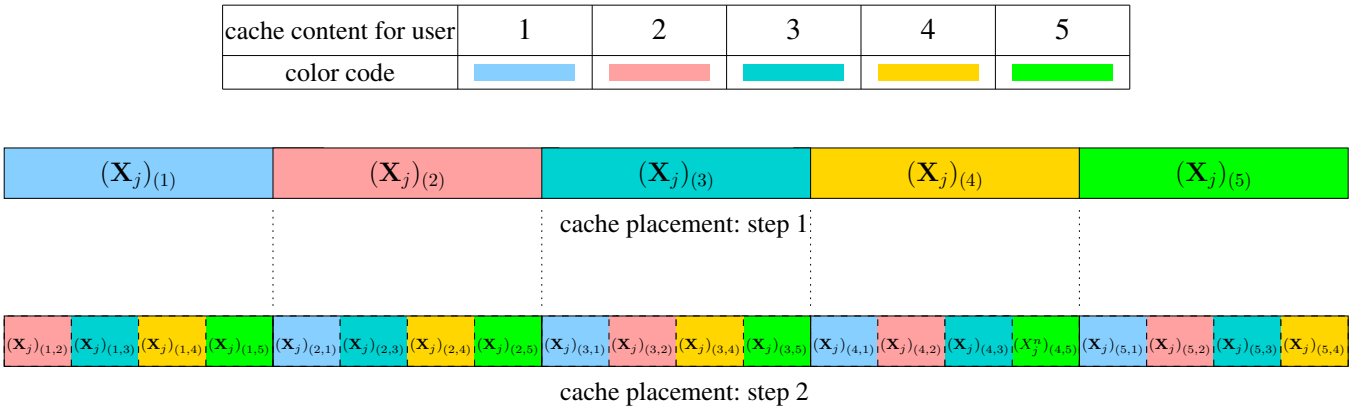

Fig. 4: Two-step caching strategy for $(C_1, C_2) = (0.4, 0.4)$ when $K = 5$ users and $N = 2$ files

$(\mathbf{X}_1)_1 \oplus (\mathbf{X}_1)_3$, $(\mathbf{X}_1)_1 \oplus (\mathbf{X}_1)_4$, and $(\mathbf{X}_1)_1 \oplus (\mathbf{X}_1)_5$. Note that user $v \in \{3, 4\}$ gains access to the cache of user 1 by recovering $(\mathbf{X}_1)_1$ from the transmitted signal $(\mathbf{X}_1)_1 \oplus (\mathbf{X}_1)_v$. Similarly for the file X_2^n, server chooses $u = 2$ and broadcasts linear combinations of $(\mathbf{X}_2)_2 \oplus (\mathbf{X}_2)_1$, $(\mathbf{X}_2)_2 \oplus (\mathbf{X}_2)_3$, $(\mathbf{X}_2)_2 \oplus (\mathbf{X}_2)_4$, and $(\mathbf{X}_2)_2 \oplus (\mathbf{X}_2)_5$. Note that user 5 gains access to the cache of user 2 by recovering $(X_2^n)_2$ from the transmitted signal $(X_2^n)_2 \oplus (X_1^n)_5$. Since every user requesting file $\mathbf{X}_j$, $j \in \{1, 2\}$, has access to $(\mathbf{X}_j)_j$, the rest of the subfiles are recoverable by simply canceling $(\mathbf{X}_j)_j$ from the delivered linear combinations. Every linear combination delivered is of rate $1/K = 0.2$, resulting in the total delivery rate of $8 \times 0.2 = 1.6$.

Suppose now that the request $\mathbf{d} = [1\ 2\ 1\ 1\ 2]$ is received at step $t = 2$, for which the condition in (9) fails. In this case, for every subset $\mathcal{S} \subset [K]$ such that $|\mathcal{S}| = t + 1$ and for every $\sigma \in \Sigma_t$, server broadcasts the linear combination

$$\bigoplus_{s \in \mathcal{S}} (\mathbf{X}_{d_s})_{\sigma(\mathcal{S} \setminus \{s\})}. \tag{11}$$

Contrary to (10), here the subfiles across different files are coded linearly. Every such linear combination has the rate of $\prod_{r=1}^{t} \frac{1}{K-r+1}$, resulting in the total rate of

$$\frac{\binom{K}{t+1} t!}{K(K-1)\cdots(K-t+1)} = \frac{K-t}{t+1} = R_{\text{worst}}(tN/K)$$

when combined. Given such a subset $\mathcal{S}$ and permutation σ, user $k \in \mathcal{S}$ can recover all the subfiles of the form $(\mathbf{X}_{d_k})_{\sigma(\mathcal{S} \setminus \{k\})}$ from the linear combination in (11). For example, when $\mathcal{S} = \{1, 2, 3\}$, the corresponding linear combinations for different permutations are $(\mathbf{X}_1)_{(2,3)} \oplus (\mathbf{X}_2)_{(1,3)} \oplus (\mathbf{X}_1)_{(1,2)}$ and $(\mathbf{X}_1)_{(3,2)} \oplus (\mathbf{X}_2)_{(3,1)} \oplus (\mathbf{X}_1)_{(2,1)}$. User 1 having cached $(\mathbf{X}_2)_{(1,3)}$, $(\mathbf{X}_2)_{(3,1)}$, $(\mathbf{X}_1)_{(1,2)}$, and $(\mathbf{X}_1)_{(2,1)}$, is able to recover the subfiles $(\mathbf{X}_1)_{(2,3)}$ and $(\mathbf{X}_1)_{(3,2)}$ of its requested file $\mathbf{X}_1$. Generalizing to every set $\mathcal{S} \subset [5]$ such that $|\mathcal{S}| = 3$, the linear combinations broadcasted can be listed as

$$\begin{aligned}
&(\mathbf{X}_1)_{(2,3)} \oplus (\mathbf{X}_2)_{(1,3)} \oplus (\mathbf{X}_1)_{(1,2)},\\
&(\mathbf{X}_1)_{(3,2)} \oplus (\mathbf{X}_2)_{(3,1)} \oplus (\mathbf{X}_1)_{(2,1)},\\
&(\mathbf{X}_1)_{(2,4)} \oplus (\mathbf{X}_2)_{(1,4)} \oplus (\mathbf{X}_1)_{(1,2)},\\
&(\mathbf{X}_1)_{(4,2)} \oplus (\mathbf{X}_2)_{(4,1)} \oplus (\mathbf{X}_1)_{(2,1)},\\
&(\mathbf{X}_1)_{(2,5)} \oplus (\mathbf{X}_2)_{(1,5)} \oplus (\mathbf{X}_2)_{(1,2)},\\
&(\mathbf{X}_1)_{(5,2)} \oplus (\mathbf{X}_2)_{(5,1)} \oplus (\mathbf{X}_2)_{(2,1)},\\
&(\mathbf{X}_1)_{(3,4)} \oplus (\mathbf{X}_1)_{(1,4)} \oplus (\mathbf{X}_1)_{(1,3)},\\
&(\mathbf{X}_1)_{(4,3)} \oplus (\mathbf{X}_1)_{(4,1)} \oplus (\mathbf{X}_1)_{(3,1)},\\
&(\mathbf{X}_1)_{(3,5)} \oplus (\mathbf{X}_1)_{(1,5)} \oplus (\mathbf{X}_2)_{(1,3)},\\
&(\mathbf{X}_1)_{(5,3)} \oplus (\mathbf{X}_1)_{(5,1)} \oplus (\mathbf{X}_2)_{(3,1)},\\
&(\mathbf{X}_1)_{(4,5)} \oplus (\mathbf{X}_1)_{(1,5)} \oplus (\mathbf{X}_2)_{(1,4)},\\
&(\mathbf{X}_1)_{(5,4)} \oplus (\mathbf{X}_1)_{(5,1)} \oplus (\mathbf{X}_2)_{(4,1)},\\
&(\mathbf{X}_2)_{(3,4)} \oplus (\mathbf{X}_1)_{(2,4)} \oplus (\mathbf{X}_1)_{(2,3)},\\
&(\mathbf{X}_2)_{(4,3)} \oplus (\mathbf{X}_1)_{(4,2)} \oplus (\mathbf{X}_1)_{(3,2)},\\
&(\mathbf{X}_2)_{(3,5)} \oplus (\mathbf{X}_1)_{(2,5)} \oplus (\mathbf{X}_2)_{(2,3)},\\
&(\mathbf{X}_2)_{(5,3)} \oplus (\mathbf{X}_1)_{(5,2)} \oplus (\mathbf{X}_2)_{(3,2)},\\
&(\mathbf{X}_2)_{(4,5)} \oplus (\mathbf{X}_1)_{(2,5)} \oplus (\mathbf{X}_2)_{(2,4)},\\
&(\mathbf{X}_2)_{(5,4)} \oplus (\mathbf{X}_1)_{(5,2)} \oplus (\mathbf{X}_2)_{(4,2)},\\
&(\mathbf{X}_1)_{(4,5)} \oplus (\mathbf{X}_1)_{(3,5)} \oplus (\mathbf{X}_2)_{(3,4)},\\
&(\mathbf{X}_1)_{(5,4)} \oplus (\mathbf{X}_1)_{(5,3)} \oplus (\mathbf{X}_2)_{(4,3)},
\end{aligned}$$

each of which has the rate $\frac{1}{K(K-1)}$, resulting in the total delivery rate of $\binom{5}{3} \times 2! \times 0.05 = 1$. From these linear combinations, user $k \in [K]$ is able to recover all subfiles of the form

$$\{(\mathbf{X}_{d_k})_{(\mathcal{T})} : (\mathcal{T}) \text{ is an ordered subset of } [K] \setminus \{k\}, |\mathcal{T}| = t\}$$

of its requested file $\mathbf{X}_{d_k}$. Since the remaining subfiles are already available in its cache, user k is able to completely recover its requested file $\mathbf{X}_{d_k}$. We can conclude that every user is able to recover its requested file in a successful manner since $k \in [K]$ is arbitrary.

Our arguments presented thus far for $T = 2$ can be generalized for larger T values by further splitting each subfiles at step $t = 2$ of the cache placement into $(K-2)$ subfiles at step $t = 3$ and continue splitting upto step $t = K$ at most, which eventually results in $K!$ many subfiles. ■

REFERENCES

[1] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. 29th Ann. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, March 2010, pp. 1–9.

[2] Y. Birk and T. Kol, "Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2825–2830, Jun. 2006.

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

[4] Q. Yang and D. Gündüz, "Coded caching and content delivery with heterogeneous distortion requirements," *IEEE Trans. Inf. Theory*, vol. 64, no. 6, pp. 4347–4364, June 2018.

[5] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Coded caching for heterogeneous systems: An optimization perspective," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5321–5335, Aug 2019.

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

[7] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, "Correlation-aware distributed caching and coded delivery," in *Proc. IEEE Inf. Theory Workshop*, Sep. 2016, pp. 166–170.

[8] C. Wang, S. H. Lim, and M. Gastpar, "Information-theoretic caching: Sequential coding for computing," *IEEE Transactions on Information Theory*, vol. 62, no. 11, pp. 6393–6406, Nov 2016.

[9] R. M. Gray and A. D. Wyner, "Source coding for a simple network," *Bell Syst. Tech. J.*, vol. 53, no. 9, pp. 1681–1721, 1974.

[10] C.-Y. Wang, *Function Computation over Networks Efficient Information Processing for Cache and Sensor Applications*. Lausanne: EPFL, 2015.

[11] S. H. Lim, C. Wang, and M. Gastpar, "Information-theoretic caching: The multi-user case," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7018–7037, Nov 2017.

[12] P. Hassanzadeh, E. Erkip, J. Llorca, and A. Tulino, "Distortion-memory tradeoffs in cache-aided wireless video delivery," in *Proc. 53th Ann. Allerton Conf. Comm. Control Comput.*, Sep. 2015, pp. 1150–1157.

[13] G. Op 't Veld and M. C. Gastpar, "Caching gaussians: Minimizing total correlation on the gray–wyner network," *Proceedings of the 50th Annual Conference on Information Systems and Sciences (CISS)*, p. 6, 2016.

[14] Q. Yang and D. Gündüz, "Centralized coded caching for heterogeneous lossy requests," in *Proc. IEEE Int. Symp. Inf. Theory*, July 2016, pp. 405–409.

[15] R. Timo, S. S. Bidokhti, M. Wigger, and B. C. Geiger, "A rate-distortion approach to caching," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1957–1976, March 2018.

[16] P. Sen and M. Gastpar, "Successive refinement to caching for dynamic content," in *Proc. IEEE Int. Symp. Inf. Theory*, July 2019, pp. 2484–2488.

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley, 2006.

[18] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge: Cambridge University Press, 2011.

[19] A. Lapidoth and M. Wigger, "Conditional and relevant common information," in *2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)*, Nov 2016, pp. 1–5.

[20] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," *IEEE Trans. Inf. Theory*, vol. 63, no. 7, pp. 4388–4413, July 2017.