Multi-Server Coded Caching: Small Cache Size

Vijith Kumar K P
Email: kumarkp.vijith@gmail.com

Brijesh Kumar Rai
Email: brijesh.rai@gmail.com

Tony Jacob
IIT Guwahati, India
Email: tonyj@iitg.ac.in

***Abstract*—In this paper, we consider a multi-server cache network with L non-interacting servers, where server $l \in [L]$ has a library of N_l files, each of size F bits. Each of these L servers communicates with K users over shared, error-free broadcast channels. Each user is equipped with an isolated cache of size MF bits, where $M \in \left[0, \sum_{l\in[L]} N_l\right]$. Despite several prior efforts, the exact characterization of the optimal rate–memory tradeoff for the multi-server setting remains an open problem. In this paper, we characterize the optimal rate–memory tradeoff for a small-cache regime, specifically for $M \in \left[0, \frac{1}{K}\right]$. This is achieved by proposing a new caching scheme that operates at the memory–rate pair $\left(\frac{1}{K}, \sum_{l\in[L]} N_l - 1\right)$ and by deriving a set of information-theoretic lower bounds that establish the optimality of the proposed scheme.**

***Index Terms*—Coded caching, multi-server cache network, exact rate memory, content distribution networks**

I. Introduction

In their seminal work [1], Maddah-Ali and Niesen introduced the concept of coded caching for content distribution networks. They studied the (N, K) canonical cache network, in which a single server having a library of N files, each of size F bits, is connected to K users, where each user has access to a dedicated cache of size MF bits with $M \in [0, N]$. This network operates in two phases. In the first phase, referred to as the *placement phase*, the server populates each cache with functions of the files from its library, without any prior knowledge of the users' future demands. Once the users reveal their demands, the server broadcasts a set of messages that enables each user to recover its requested file using the contents of its cache. This phase is known as the *delivery phase*. The main objective of the caching problem is to design the placement phase such that the communication load incurred during the delivery phase is minimized. The (N, K) canonical cache network introduced in [1] has since been extended to study several variants in [2]–[14].

In [15], Sahraei and Gastpar considered a multi-server $(N_1, \ldots, N_L, K)$ cache network, where each of these L servers communicates with K users over error-free broadcast channels. Each server $l \in [L]$ (We use $[A]$ to represent the set $\{1, 2, \ldots, A\}$) has a library of N_l files, denoted by $\{W_1^l, \ldots, W_{N_l}^l\}$, each of size F bits, as shown in Fig.1. Each user k is equipped with an isolated cache memory Z_k of size MF bits, where $M \in \left[0, \sum_{l\in[L]} N_l\right]$. Let $\boldsymbol{d} = [W_{d_1}^{l_1}, W_{d_2}^{l_2}, \ldots, W_{d_K}^{l_K}]$ denote the users' demand vector, where $W_{d_q}^{l_q}$ represents the file requested by user U_q (with d_q denoting the file index) from server l_q. Once the servers receive the users' demands, each server broadcasts a set of packets in response. Let

$$X_{\boldsymbol{d}} = \{X_{\boldsymbol{d}}^1, X_{\boldsymbol{d}}^2, \ldots, X_{\boldsymbol{d}}^L\}, \tag{1}$$

where $X_{\boldsymbol{d}}^l$ denotes the signal transmitted by server l in response to the demand vector $\boldsymbol{d}$. The transmission $X_{\boldsymbol{d}}$ has size $R_{\boldsymbol{d}}(M)F$ bits and enables the users to recover their requested files using their cache contents. The quantity $R_{\boldsymbol{d}}(M)F$ represents the network load corresponding to demand $\boldsymbol{d}$, and $R_{\boldsymbol{d}}(M)$ denotes the corresponding rate. If there exists a caching scheme that allows every user to recover its requested file from transmissions of total size at most $R(M)F$ bits for a given cache size M, then the memory–rate pair (M, R) is said to be achievable. The optimal rate–memory tradeoff is defined as the minimum achievable rate for a given cache size M, and is given by

$$R^*(M) = \min\{R \mid (M, R) \text{ is achievable}\}. \tag{2}$$

The $(N_1, \ldots, N_L, K)$ multi-server cache network was extensively studied in [16]–[20]. In [16], [17], the authors considered a multi-server setting where all the servers have the same set of files. They propose a caching scheme and analyse coding delay for different network topologies. In [18], the authors consider a setting in which users are connected to a random subset of servers during the delivery phase, where all the servers have the same set of files. In [19], Luo et al. considered the multi-server cache network from a distributed storage perspective where the

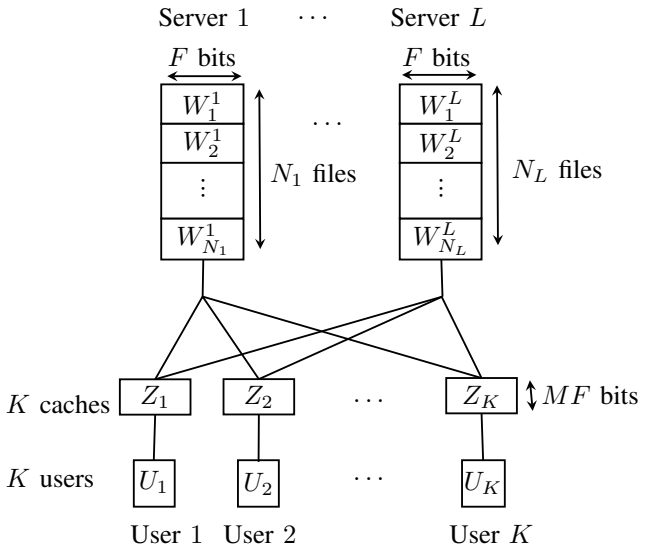

Fig. 1: The multi-server cache network

entire library of files is distributed equally across the servers with some added redundancy. In [20], Sojdeh et al., looked into multi-server coded caching problem from a privacy perspective.

In this paper, we consider the multi-server problem investigated in [15], where the servers have an independent set of files. Sahraei and Gastpar in [15] proposed a memory-sharing-based caching scheme. Resorting solely to memory sharing restricts the ability to exploit coding opportunities across files from different libraries. In this paper, we explore this possibility by allowing each user's cache, during the placement phase, to compute and store coded functions of subfiles originating from different servers. Furthermore, we demonstrate that the proposed scheme outperforms memory sharing and is optimal when cache size $M = \frac{1}{K}$.

The remainder of this paper is organized as follows. In Section II, we consider a $(2, 2, 4)$ multi-server cache network to demonstrate that coding across servers can reduce the delivery rate. In Section III, we present a general caching scheme for the regime $M \leq \frac{1}{K}$ and establish the optimality of the proposed scheme. Finally, Section IV concludes the paper.

II. The $(2, 2, 4)$ multi server cache network

In this section, we consider a two-server caching network. The network consists of two non-interacting servers, Server 1 and Server 2, each server connected to four users, $\{U_1, U_2, U_3, U_4\}$, through a common error-free broadcast channel. Server 1 has a library of two files, $\{A, B\}$, and Server 2 has a library of two files, $\{C, D\}$, where all files are of equal size F bits. We consider a demand vector $\mathbf{d}$ in which the users collectively request all the files available in both servers.

A. Cache $M = \frac{1}{4}$

First, consider a cache network in which each user has an isolated cache with a size of $MF = \frac{1}{4}F$ bits. For this network, we propose a caching scheme based on coding across servers. During the placement phase, each file is divided into four non-overlapping subfiles of $\frac{1}{4}F$ bit size. The subfiles are as follows:

File	Subfiles
A	A_1, A_2, A_3, A_4
B	B_1, B_2, B_3, B_4
C	C_1, C_2, C_3, C_4
D	D_1, D_2, D_3, D_4

The server transmits a function of these subfiles to each user's cache, and the caches combine the packets from each server and store. The cache contents of each user are shown in TABLE I. Each function is of size $\frac{1}{4}F$ bits and they occupy the space of $\frac{1}{4}F$ bits.

Consider the demand $\boldsymbol{d} = [A, B, C, D]$. In response to this demand, server 1 broadcasts the set of messages:

$$X^1_{\boldsymbol{d}} = \{A_2, A_3, A_4, B_1, B_3, B_4, \} \tag{3}$$

Similarly, the server 2 broadcasts the set of messages:

$$X^2_{\boldsymbol{d}} = \{C_1, C_2, C_4, D_1, D_2, D_3\} \tag{4}$$

In response to the demand $\boldsymbol{d}$, both the servers together broadcasts 12 packets, each of size $\frac{1}{4}F$ bits. Thus, the rate that corresponds to the demand $\boldsymbol{d}$ is $R = 3$. To analyze how each user recovers its requested file, consider user 1. User 1 requests the file A from server 1. From the broadcast message $X^1_{\boldsymbol{d}}$, user 1 directly obtains the subfiles A_2, A_3, A_4. Using the received subfiles B_1 (in $X^1_{\boldsymbol{d}}$), C_1 and D_1 (in $X^2_{\boldsymbol{d}}$), the user computes the subfile A_1 from its cache content

$$A_1 + B_1 + C_1 + D_1$$

In a similar fashion, user 2 recovers the file B, user 3 recovers the file C and user 4 recovers the file D.

TABLE I: Cache contents when $M = \frac{1}{4}$

Caches	Contents
Z_1	$A_1 + B_1 + C_1 + D_1$
Z_2	$A_2 + B_2 + C_2 + D_2$
Z_3	$A_3 + B_3 + C_3 + D_3$
Z_4	$A_4 + B_4 + C_4 + D_4$

We can note that the scheme proposed at cache size $M = \frac{1}{4}$ is similar to the scheme proposed by Chen et al. [22] for the single server cache network where a server with four files communicating with four users. Using arguments similar to the one used in the single server network we can demonstrate the optimality of the propose scheme.

Lemma 1. *For the $(2, 2, 4)$ cache network the achievable memory rate tuple (M, R) must satisfy the following constraint:*

$$4M + R \geq 4$$

Proof. We have,

$$\begin{aligned} 4M + R \geq & H(Z_1) + H(Z_2) + H(Z_3) + H(Z_4) \\ & + H(X_{[A,B,C,D]}) \\ \overset{(a)}{\geq} & H(Z_1, Z_2, Z_3, Z_4, X_{[A,B,C,D]}) \\ \overset{(b)}{=} & H(A, B, C, D, Z_1, Z_2, Z_3, Z_4, X_{[A,B,C,D]}) \\ \overset{(c)}{=} & H(A, B, C, D) = 4 \end{aligned}$$

□

The new proposed scheme achieves the memory rate pair $(\frac{1}{4}, 3)$. It can be noted that this memory rate pair lies on the rate memory rate tradeoff in Lemma 1. Thus, we have the following theorem:

Theorem 1. *For the $(2, 2, 4)$ multi-serve cache network, the proposed scheme is optimal for the demands where all files in the libraries are requested by the users.*

B. Cache size $M = \frac{1}{3}$

Now, consider a cache network in which each user has an isolated cache with a size of $MF = \frac{1}{3}F$ bits. For this network, we propose a caching scheme based on coding across servers.

During the placement phase, each file is divided into 12 non-overlapping subfiles of $\frac{1}{12}F$ bit size. The subfiles are as follows:

File	Subfiles
A	$A_1,\ A_2,\ A_3,\ A_4,\ A_5,\ A_6,\ A_7,\ A_8,\ A_9,\ A_{10},\ A_{11},\ A_{12}$
B	$B_1,\ B_2,\ B_3,\ B_4,\ B_5,\ B_6,\ B_7,\ B_8,\ B_9,\ B_{10},\ B_{11},\ B_{12}$
C	$C_1,\ C_2,\ C_3,\ C_4,\ C_5,\ C_6,\ C_7,\ C_8,\ C_9,\ C_{10},\ C_{11},\ C_{12}$
D	$D_1,\ D_2,\ D_3,\ D_4,\ D_5,\ D_6,\ D_7,\ D_8,\ D_9,\ D_{10},\ D_{11},\ D_{12}$

Each server transmit a set of functions of these subfiles to each user's cache and caches store combined versions of the received packets as shown in TABLE II. Each user store four function each of size $\frac{1}{12}F$ bits in its cache and they occupy the space of $\frac{1}{3}F$ bits.

TABLE II: Cache contents when $M = \frac{1}{3}$

Caches	Contents
Z_1	$A_1 + B_1 + C_1,\ A_2 + B_2 + D_1,$ $A_3 + C_2 + D_2,\ B_3 + C_3 + D_3$
Z_2	$A_4 + B_4 + C_4,\ A_5 + B_5 + D_4,$ $A_6 + C_5 + D_5,\ B_6 + C_6 + D_6$
Z_3	$A_7 + B_7 + C_7,\ A_8 + B_8 + D_7,$ $A_9 + C_8 + D_8,\ B_9 + C_9 + D_9$
Z_4	$A_{10} + B_{10} + C_{10},\ A_{11} + B_{11} + D_{10},$ $A_{12} + C_{11} + D_{11},\ B_{12} + C_{12} + D_{12}$

Consider the demand $\boldsymbol{d} = [A, B, C, D]$. In response to the demand $\boldsymbol{d}$ the server 1 broadcasts the set of messages:

$$X^1_{\boldsymbol{d}} = \left\{ \begin{array}{c} A_4,\ A_5,\ A_7,\ A_8,\ A_9,\ A_{10},\ A_{11},\ A_{12}, B_1,\ B_2, \\ B_7,\ B_8,\ B_9,\ B_{10},\ B_{11},\ B_{12}, A_6 + B_3 \end{array} \right\}$$

Similarly, the server 2 broadcasts the set of messages:

$$X^2_{\boldsymbol{d}} = \left\{ \begin{array}{c} C_1,\ C_2,\ C_3,\ C_4,\ C_5,\ C_6,\ C_{11},\ C_{12}, \\ D_1,\ D_2,\ D_3,\ D_4,\ D_5,\ D_6,\ D_8,\ D_9, \\ C_{10} + D_7 \end{array} \right\}$$

During the delivery phase both the servers together broadcast a total 34 messages each of size $1\frac{1}{12}F$ bits. Thus, the rate corresponding the demand $\boldsymbol{d}$ is $R = \frac{17}{6}$.

To analyze how each user recovers its requested file, consider user 1. User 1 requests the file A from server 1. From the broadcast message $X^1_{\boldsymbol{d}}$, user 1 directly obtains the subfiles $A_4,\ A_5,\ A_7,\ A_8,\ A_9,\ A_{10},\ A_{11},\ A_{12}$. The user can compute subfiles A_1, A_2, A_3, and B_3 by combining the packets received in $X^2_{\boldsymbol{d}}$ and cached packets as shown below:

Received Packet	Cached Packet	Computed Packet
$B_1,\ C_1$	$A_1 + B_1 + C_1$	A_1
$B_2,\ D_1$	$A_2 + B_1 + C_1$	A_2
$C_2,\ D_2$	$A_3 + C_2 + D_2$	A_3
$C_3,\ D_3$	$B_3 + C_3 + D_3$	B_3

Once the user obtain the subfile B_3, it compute the subfile A_6 from the packet

$$A_6 + B_3$$

in $X^1_{\boldsymbol{d}}$. In similar fashion user 2 recover the file B, user 3 recover the file C and user 4 recover the file D.

C. Cache size $M = \frac{1}{2}$

Now, consider a cache network in which each user has an isolated cache with a size of $MF = \frac{1}{2}F$ bits. For this network, we propose a caching scheme based on coding across servers. During the placement phase, each file is divided into 8 non-overlapping subfiles of $\frac{1}{8}F$ bit size. The subfiles are as follows:

File	Subfiles
A	$A_1,\ A_2,\ A_3,\ A_4,\ A_5,\ A_6,\ A_7,\ A_8$
B	$B_1,\ B_2,\ B_3,\ B_4,\ B_5,\ B_6,\ B_7,\ B_8$
C	$C_1,\ C_2,\ C_3,\ C_4,\ C_5,\ C_6,\ C_7,\ C_8$
D	$D_1,\ D_2,\ D_3,\ D_4,\ D_5,\ D_6,\ D_7,\ D_8$

Each server transmit a set of functions of these subfiles to each user's cache and caches store combined versions of the received packets as shown in TABLE III. Each user store four function each of size $\frac{1}{8}F$ bits in its cache and they occupy the space of $\frac{1}{2}F$ bits.

TABLE III: Cache contents when $M = \frac{1}{2}$

Caches	Contents
Z_1	$A_1 + C_1,\ A_2 + D_2,\ B_1 + C_2,\ B_2 + D_2$
Z_2	$A_3 + C_3,\ A_4 + D_3,\ B_3 + C_4,\ B_4 + D_4$
Z_3	$A_5 + C_5,\ A_6 + D_5,\ B_5 + C_6,\ B_6 + D_6$
Z_4	$A_7 + C_7,\ A_8 + D_7,\ B_7 + C_8,\ B_8 + D_8$

Consider the demand $\boldsymbol{d} = [A, B, C, D]$. In response to this demand, server 1 broadcasts the set of messages:

$$X^1_{\boldsymbol{d}} = \left\{ \begin{array}{c} A_5,\ A_6, A_7,\ A_8, B_5,\ B_6,\ B_7,\ B_8, \\ A_3 + B_1,\ A_4 + B_2 \end{array} \right\}$$

Similarly the server 2 broadcasts the set of messages:

$$X^2_{\boldsymbol{d}} = \left\{ \begin{array}{c} C_1,\ C_2, C_3,\ C_4,\ D_1,\ D_2,\ D_3,\ D_4, \\ C_7 + D_5,\ C_8 + D_6 \end{array} \right\}$$

In response to the demand $\boldsymbol{d}$, both the servers combined broadcast 20 packets each of size $\frac{1}{8}F$ bits. Thus, the rate that corresponds to the demand $\boldsymbol{d}$ is

$$R = \frac{5}{2}$$

To analyze how each user recover its requested file, consider user 1. User 1 requests the file A from server 1. From the broadcast message $X^1_{\boldsymbol{d}}$, user 1 directly obtain the subfiles $A_5,\ A_6,\ A_7$, and A_8. The user can compute subfiles A_1, A_2, B_1, and B_2 by combining the packets received in $X^2_{\boldsymbol{d}}$ and cached packets as shown below:

Received Packet	Cached Packet	Computed Packet
C_1	$A_1 + C_1$	A_1
D_1	$A_2 + D_1$	A_2
C_2	$B_1 + C_2$	B_1
D_2	$B_2 + D_2$	B_2

Once the user obtain the subfiles B_1 and B_2, it compute the subfiles A_3 and A_4 from the packets $A_3 + B_1$ and $A_4 + B_2$ (received in $X_{\boldsymbol{d}}^1$), respectively. In similar fashion user 2 recover the file B, user 3 recover the file C and user 4 recover the file D.

These results are summarized in Fig. 2 and TABLE IV, along with the rates obtained memory sharing MAN scheme, proposed in [1], and memory sharing the CFL scheme, proposed in [22].

Cache Size	Memory sharing MAN scheme [1]	Memory sharing CFL scheme [22]	Our scheme
$0 \le M \le \frac{1}{4}$	$4 - \frac{3}{2}M$	$4 - 2M$	$4 - 4M$
$\frac{1}{4} \le M \le \frac{1}{3}$	$4 - \frac{3}{2}M$	$4 - 2M$	$\frac{7}{2} - 2M$
$\frac{1}{3} \le M \le \frac{1}{2}$	$4 - \frac{3}{2}M$	$4 - 2M$	$\frac{21}{7} - 2M$
$\frac{1}{2} \le M \le 1$	$4 - \frac{3}{2}M$	$4 - 2M$	$\frac{17}{6} - \frac{2}{3}M$

TABLE IV: Achievable rates for $(2, 2, 4)$ multi-server cache network

III. General scheme

In this section, we present a caching scheme for an L-server cache network that achieves the memory–rate pair

$$(M, R) = \left(\frac{1}{K}, \left(\sum_{l \in [L]} N_l \right) (K - 1) \right).$$

First, each server partitions every file into K non-overlapping subfiles, each of size $\frac{F}{K}$ bits. Specifically, the subfiles of file W_n^l in the server $l \in [L]$ are denoted as

$$W_n^l = \{W_{n,k}^l : k \in [K]\}. \tag{5}$$

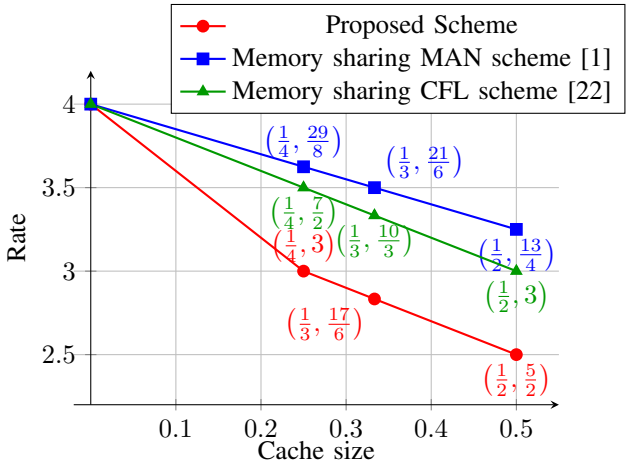

Fig. 2: Achievable rates for $(2, 2, 4)$ multi-server cache network

During the placement phase, server l transmits the function

$$\bigoplus_{n_l \in [N_l]} W_{n_l,k}^l \tag{6}$$

to user $k \in [K]$, where each such packet has size $\frac{F}{K}$ bits. Upon receiving these packets from all servers, user k stores the combined function

$$Z_k = \bigoplus_{l \in [L]} \bigoplus_{n_l \in [N_l]} W_{n_l,k}^l. \tag{7}$$

Since the size of Z_k is $\frac{F}{K}$ bits, it fully utilizes the cache memory of user k.

In the delivery phase, the servers receive a demand vector $\mathbf{d}$, where $W_{d_k}^l$ denotes the file requested by user k from server l. Let $N_{\mathbf{d}}^l$ denote the set of users that request files from server l, for $l \in [L]$. For a given file W_n^l, let $N_{\mathbf{d}}(W_n^l)$ represent the set of users requesting that file.

In response to the demand $\mathbf{d}$, server l broadcasts the following set of packets:

$$X_{\mathbf{d},1}^l = \left\{ \begin{array}{l} \bigcup\limits_{i \in N_{\mathbf{d}}^l} \bigcup\limits_{j \in [K] \setminus \{i\}} W_{d_i,j}^l, \\ \bigcup\limits_{p \in [K] \setminus N_{\mathbf{d}}^l} \bigcup\limits_{n_l \in [N_l]} W_{n_l,p}^l, \\ \bigcup\limits_{i \in N_{\mathbf{d}}^l} \bigcup\limits_{j \in N_{\mathbf{d}}(W_{d_i}^l)} \left(W_{d_i,i}^l \oplus W_{d_i,j}^l \right) \end{array} \right\}. \tag{8}$$

Let

$$X_{\mathbf{d}} = \{X_{\mathbf{d},1}^1, X_{\mathbf{d},1}^2, \ldots, X_{\mathbf{d},1}^L\} \tag{9}$$

denote the collection of packets transmitted by all servers.

The total number of packets transmitted by server l during the delivery phase is

$$\begin{aligned} &|N_{\mathbf{d}}^l|(N_l - 1) + N_l\big(K - |N_{\mathbf{d}}^l|\big) + \sum_{n_l \in [N_l]} \big(|N_{\mathbf{d}}(W_{n_l}^l)| - 1\big) \\ &\qquad = N_l K - N_l, \end{aligned} \tag{10}$$

where each packet has size $\frac{F}{K}$ bits.

Thus, across all servers, the total number of transmitted packets is

$$\sum_{l \in [L]} (N_l K - N_l) = (K - 1) \sum_{l \in [L]} N_l, \tag{11}$$

each of size $\frac{F}{K}$ bits. Therefore, the delivery rate corresponding to demand $\mathbf{d}$ is

$$R = \frac{(K - 1) \sum_{l \in [L]} N_l}{K}. \tag{12}$$

To explain how users recover their requested files, consider user k, who requests file $W_{d_k}^l$. Using its cache content Z_k and the received packets $X_{\mathbf{d}}$, user k must recover all subfiles $\{W_{d_k,j}^l : j \in [K]\}$. The recovery process proceeds in two stages, which we describe next.

A. Stage 1

In the first stage, user k directly recovers the subfiles $W^l_{d_k,j}$ for all $j \in [K] \setminus \{k\}$ from the received packets. Next, the user recovers the remaining subfile $W^l_{d_k,k}$. Observe that the subfile $W^l_{d_k,k}$ is available in the user's cache in a coded form as

$$W^l_{d_k,k} \oplus \bigoplus_{p\in[N_l]\setminus\{d_k\}} W^l_{p,k} \oplus \bigoplus_{q\in[L]\setminus\{l\}} \bigoplus_{n\in[N_q]} W^q_{n,k}.$$

User k combines this cached packet with the received packets $W^l_{p,k}$, for $p \in [N_l] \setminus \{d_k\}$, transmitted by server l, and the packets $W^q_{n,k}$ transmitted by the remaining servers $q \in [L] \setminus \{l\}$. This allows user k to recover the subfile $W^l_{d_k,k}$. Using all recovered subfiles, the user reconstructs the requested file $W^l_{d_k}$.

B. Stage 2

In this stage, we focus on users requesting the same file. For user k, the remaining missing subfiles are $\{W^l_{d_k,s} : s \in N_{\mathbf{d}}(W^l_{d_k}) \setminus \{k\}\}$. Note that user k has already recovered the subfile $W^l_{d_k,k}$ in the first stage. The user now recovers the remaining subfiles using the received packets

$$\bigcup_{j\in N_{\mathbf{d}}(W^l_{d_k})} \left(W^l_{d_k,k} \oplus W^l_{d_k,j}\right).$$

Combining these packets with the recovered subfile $W^l_{d_k,k}$, user k successfully recovers all remaining subfiles and hence reconstructs the requested file $W^l_{d_k}$.

These observations are summarized in the following theorem.

Theorem 2. *For the $(N_1, \ldots, N_L, K)$ multi-server cache network, the memory–rate pair*

$$(M, R) = \left(\frac{1}{K}, \left(\sum_{l\in[L]} N_l\right)(K-1)\right)$$

is achievable using symmetric caching schemes.

Next, we establish the optimality of the proposed scheme.

Lemma 2. *For the $(N_1, \ldots, N_L, K)$ multi-server cache network, any achievable memory–rate pair (M, R) must satisfy*

$$NM + R \geq \sum_{l\in[L]} N_l, \tag{13}$$

where $N = \sum_{l\in[L]} N_l$.

Proof. Consider a demand $\mathbf{d}$ in which users collectively request all files across all servers. Without loss of generality, assume that the first N users request distinct files and collectively request all N files. Then,

$$\begin{aligned} NM + R &\geq \sum_{k\in[N]} H(Z_k) + H(X_{\mathbf{d}}) \\ &\overset{(a)}{\geq} H(Z_1, \ldots, Z_N, X_{\mathbf{d}}) \\ &\overset{(b)}{=} H\left(\bigcup_{l\in[L]} \bigcup_{n_l\in[N_l]} W^l_{n_l}, Z_1, \ldots, Z_N, X_{\mathbf{d}}\right) \\ &\overset{(c)}{=} H\left(\bigcup_{l\in[L]} \bigcup_{n_l\in[N_l]} W^l_{n_l}\right) \\ &= \sum_{l\in[L]} N_l, \end{aligned}$$

where (a) follows from the submodularity of entropy, (b) follows from the fact that each user can recover its requested file from its cache content and the received packets, and (c) follows since the cache contents and broadcast packets are deterministic functions of the files in the servers. □

The new proposed scheme achieves the memory rate pair $(\frac{1}{K}, \frac{(K-1)\sum_{l\in[L]} N_l}{K})$. It can be noted that this memory rate pair lies on the rate memory rate tradeoff in Lemma 2. Thus, we have the following theorem:

Theorem 3. *For the $(N_1, \ldots, N_L, K)$ multi-serve cache network, the proposed scheme is optimal for the demands where all files in the libraries are requested by the users.*

IV. Conclusions

In this paper, we studied the $(2, 2, 4)$ multi-server cache network consisting of two servers, each having a library of two files. For this network, we proposed a new caching scheme that achieves the memory rate pairs $(\frac{1}{4}, 3)$, $(\frac{1}{3}, \frac{17}{6})$, and $(\frac{1}{2}, 2.5)$. We also derived a converse bound and showed that the proposed scheme is optimal when the cache size is $M = \frac{1}{4}$. Furthermore, we extended both the proposed scheme and the converse bound to the $(N_1, \ldots, N_l, K)$ multi-server cache network in the regime where $M = \frac{1}{K}$. This extension led to an exact characterization of the rate memory tradeoff for the L-server cache network in the small-cache regime. These results demonstrate that coding across servers can strictly improve the achievable delivery rate compared to conventional memory-sharing strategies. The design of a general caching scheme based on coding across servers for larger cache sizes, particularly for $M \in [0, \frac{N}{K}]$, remains an open problem and is left for future work.

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, 2014.

[2] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, 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, 2017.

[4] 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, 2016.

[5] E. Parrinello and P. Elia, "Coded caching with optimized shared-cache sizes," in *2019 IEEE ITW*. IEEE, 2019, pp. 1–5.

[6] A. Malik, B. Serbetci, E. Parrinello, and P. Elia, "Fundamental limits of stochastic caching networks," *arXiv preprint arXiv:2005.13847*, 2020.

[7] Ç. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of d2d coded caching with uncoded cache placement and one-shot delivery," *IEEE Transactions on Communications*, vol. 67, no. 12, pp. 8179–8192, 2019.

[8] K. Wan, D. Tuninetti, M. Ji, G. Caire, and P. Piantanida, "Fundamental limits of decentralized data shuffling," *IEEE Transactions on Information Theory*, vol. 66, no. 6, pp. 3616–3637, 2020.

[9] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 3108–3141, 2017.

[10] K. S. Reddy and N. Karamchandani, "Rate-memory trade-off for multi-access coded caching with uncoded placement," *IEEE Transactions on Communications*, vol. 68, no. 6 pp. 3261–3274, 2020.

[11] K. Krishnan Namboodiri and B. Sundar Rajan, "An improved lower bound for multi-access coded caching,"*IEEE Transactions on Communications*, vol. 70, no. 7, pp. 4454–4468, 2022.

[12] S. Sasi and B. S. Rajan, "An improved multi-access coded caching with uncoded placement," *arXiv preprint arXiv:2009.05377*, 2020.

[13] F. Brunero and P. Elia, "Fundamental limits of combinatorial multi-access caching," *IEEE Transactions on Information Theory*, vol. 69, no. 2, pp. 1037–1056, 2022.

[14] A. Malik, B. Serbetci, E. Parrinello, and P. Elia, "Fundamental limits of stochastic shared-cache networks," *IEEE Transactions on Communications*, vol. 69, no. 7, pp. 4433–4447, 2021.

[15] Saeid Sahraei, and Michael Gastpar, "Multi-library coded caching," in *2016 Allerton*. IEEE, 2016, pp. 1012–1017.

[16] S.P. Shariatpanahi., S.A. Motahari, and B.H. Khalaj , "Multi-server coded caching,"*IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 7253–7271, 2016.

[17] S.P. Shariatpanahi., S.A. Motahari, and B.H. Khalaj, "On multi-server coded caching in the low memory regime," *arXiv preprint arXiv:1803.07655*, 2018.

[18] Nitish Mital, Deniz Gündüz, and Cong Ling, "Coded caching in a multi-server system with random topology,"*IEEE Transactions on communications*, vol. 68, no. 8, pp. 4620–4631, 2020.

[19] Tianqiong Luo, Vaneet Aggarwal, and Borja Peleato, "Coded caching with distributed storage,"*IEEE Transactions on Information Theory*, vol. 65, no. 12, pp. 7742–7755, 2019.

[20] M. J. Sojdeh, M. Letafati, and S.P. Shariatpanahi, and B. H. Khalaj, "Secure multi-server coded caching,"*Elsevier Computer Networks*, vol. 253, no. 12, pp. 110715, 2024.

[21] 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.

[22] 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.