Multi-Access Cache-Aided Multi-User Private Information Retrieval

Kanishak Vaidya, Student Member, IEEE, and B. Sundar Rajan, Life Fellow, IEEE

Abstract—In a Multi-user Private Information Retrieval (MuPIR) problem, there are $N$ files replicated across $S$ non-colluding servers and $K$ users, each wanting to retrieve a file from the servers without letting the servers get any information about the demanded files. In a dedicated-cache-aided MuPIR problem each user is equipped with a cache that can store $M$ files. In this paper, we consider a generalized version, called multi-access cache-aided MuPIR problem, where there are $K$ users and $C \leq K$ caches each capable of storing $M$ files and each user can access several cache nodes and every cache node can be accessed by several users. The cache nodes are filled with the content of the files before users decide their demands. Then each user chooses a file index, and users cooperatively send queries to the servers to retrieve their desired files privately. The aim is to reduce the size of broadcast done by the servers as a response to these queries. We propose a scheme that utilizes multi-access caches in generalised combinatorial topology, introduced by (Brunero and Elia, 2023) and show that our scheme is order-optimal within a multiplicative factor of 2, assuming unencoded cache placement. Also, we compare the per-user rate of our setup with dedicated cache setup of (Zhang et al., 2021) in various settings.

Index Terms—Coded caching, private information retrieval (PIR), multi-access caches.

I. INTRODUCTION

The problem of Private Information Retrieval (PIR), first described in [1] deals with privately retrieving data from a set of distributed servers. A user wishes to retrieve one file amongst a set of files stored across the servers. But the servers should not know the identity of the desired file. A PIR scheme that minimizes the download cost for the user is described in [2]. After that, the PIR problem is solved for various other settings, for instance PIR with colluding servers in [3], weakly private information retrieval in [4]. PIR with private side information in [5] and PIR from coded storage at the servers in [6]. Recently, PIR is being studied with another content delivery scenario called coded caching. As described in [7], in coded caching, there are multiple cache-equipped users and one server storing some files. During off-peak hours, users’ caches are filled with the files’ content; then, during peak network traffic hours, users demand files from the server. The server will perform coded transmissions such that a single transmission can benefit multiple users simultaneously. After receiving the transmissions, users can decode their demanded files with the help of content stored in their respective caches. Recently, in [10] a cache-aided PIR strategy is described where multiple users, each with access to dedicated caches, want to recover files from non-colluding servers privately. An order-optimal retrieval scheme is described that combines the coding benefits of PIR in [2] and coded caching [7]. In [11], another dedicated cache-aided multi-user PIR strategy is provided, which reduces upload cost and the number of subpackets a file has to be divided into (also known as subpacketization level) in cache-aided PIR setup.

This paper uses a variation of the coded caching setup known as multi-access coded caching in PIR. In multi-access coded caching, users do not have access to dedicated caches. Instead, there are helper cache nodes, which the users access. The connection between users and helper cache nodes can be facilitated through wireless access points with limited coverage. For instance, the access points can be WiFi or emerging small cells (such as femtocells), providing high data rates for short distances. As a result of the high data rate provided by these access points, the communication cost of accessing the cache nodes can be effectively ignored. Therefore, caches can be placed at local access points. Multiple users can access one helper cache, and a user can access multiple caches. As is common ([10], [11]), we assume that all the servers know the contents stored in every cache node. We will use the multi-access setup described in [13] called generalised combinatorial topology. Generally, it is assumed that users can access the caches without any communication cost. However, one of the benefits of dedicated caches over multi-access caches is that the local storage of user devices can be used as dedicated caches. In contrast, multi-access caches must be connected to multiple users simultaneously, so local user storage cannot be used as multi-access cache nodes. We are also considering no communication cost between the users and the caches.

Notations: For integers $m$ and $n$, $[m:n]$ is set of integers \( \{N \in \mathbb{Z}|m \leq N \leq n\} \); \( [N] \) is same as \( [1:N] \). For a set $S$ of
size $|S|$ and integer $N\leq|S|$, $\binom{S}{N}$ denote set of all subsets of $S$ of size $N$. For set $\{a_n|n\in[N]\}$ and $N\subseteq[N]$, $a_N$ denotes set $\{a_n|n\in N\}$.

We first briefly describe single user PIR of [2], dedicated-cache coded-caching setup of [7] and multi-Access coded caching setup of [12] and [13] in the following subsections.

A. Private Information Retrieval [2]

In a Private Information Retrieval (PIR) problem, there is one user and a set of $N$ files $\mathcal{W} = \{W_n\}_{n=1}^N$ stored across $S$ non-colluding servers. These files could be replicated across these servers or can be split and encoded before storing them on the servers. In the PIR problem considered in [2], all the files are of equal size, and the files are replicated across the servers. The user wants to retrieve one out of $N$ files, say file $W_\theta$, $\theta \in \{1,2,\ldots, N\}$, but does not want the servers to know the identity of the file. In other words, the user wants to hide the index $\theta$ from the servers. In order to retrieve this desired file privately, the user generates $S$ queries $\{Q_s^\theta\}_{s=1}^S$ and sends the query $Q_s^\theta$ to server $s$. After receiving their respective queries, the servers construct answers which are a function of the queries and the files they have. Server $s$ constructs answer $A_s^\theta(Q_s^\theta, \mathcal{W})$ and sends it to the user. After receiving answers from all $S$ servers, the user should be able to decode its desired file. Privacy and correctness conditions are formally stated as follows: For privacy, we need that $I(\theta;Q_s^\theta) = 0, \forall s \in \{1,\ldots, S\}$ and for correctness $H(W_\theta) = H(A_1^\theta(Q_1^\theta, \mathcal{W}), \ldots, A_S^\theta(Q_S^\theta, \mathcal{W}), Q_1^\theta, \ldots, Q_S^\theta) = 0$.

The rate of PIR is a parameter that describes the user’s download cost (or the server’s transmission cost), which is defined as $R_{PIR} = \frac{\log |\mathcal{W}|}{R_{PIR}(S, N)}$. Note that every file is of same size, i.e. $H(W_\theta) = H(W_\theta), \forall \theta \in [N]$. Moreover, the size of the transmissions done by the servers, $H(A_s^\theta(|\mathcal{W}|))$, have to be independent of the user demand $\theta$ (otherwise, the size of transmission can give information about user demand). Therefore, the rate of PIR should be independent of user demand $\theta$. A retrieval scheme is provided in [2] which achieves the optimal rate $R_{PIR}^*(S, N)$ given by

$$R_{PIR}^*(S, N) = \frac{1}{S} + \frac{1}{S^2} + \ldots + \frac{1}{S^{N-1}}.$$  

Sending queries $Q_s^\theta$ to the servers incur an upload cost of $H(Q_s^\theta)$ bits for the user. Retrieval scheme provided in [2] incurs an upload cost of $SN \log_2(\frac{S^N}{S^{N-1}})$ bits. Another PIR scheme provided in [14] achieves PIR capacity while only incurring an upload cost of $S(N-1) \log_2 S$ bits. This upload cost is also shown to be optimal among all capacity-achieving uniformly decomposable PIR codes.

Note: In the existing literature, the term “rate” is commonly employed to characterize the system’s performance. However, its interpretation can vary among different works. For instance, in reference [2], the rate is defined as being inversely proportional to the number of bits a user needs to download. Conversely, reference [7] defines the rate as being directly proportional to the number of bits downloaded by the user. In our paper, to avoid ambiguity and to align with the conventions followed in references [7], [12], and [15], we use the term “transmission cost” interchangeably with “rate”. Thus, our definition of “transmission cost” corresponds to the notion of rate as described in [7], [12], and [15]. Consequently, as the number of bits downloaded by the user increases, the transmission cost also increases accordingly.

B. Coded Caching Problem

In a centralized setup described in [7], a server stores $N$ independent files denoted as $W_0, \ldots, W_{N-1}$, all of the same size. There are $K$ users, each capable of storing $M$ files. The system operates in two phases. During the placement phase, when the network is uncongested, the server preloads the caches with file contents. In the context of file storage within caches, when files are partitioned into smaller units and placed in cache storage, this approach is referred to as uncoded prefetching or uncoded placement. On the other hand, if the files undergo encoding before being stored in the caches, this method is termed as coded placement. Then, in the delivery phase, all users intend to retrieve certain files from the server. User $k$ aims to retrieve file $d_k \in [0 : N - 1]$. Each user communicates their desired file index to the server. Upon receiving users’ requests, the server broadcasts coded transmissions $\mathbf{X}$ of size equivalent to $R_{CC}^L$ file. These transmissions $\mathbf{X}$ are generated based on the server’s stored files and user demands. Upon receiving the coded transmission $\mathbf{X}$, all users should be able to recover their desired files using their cache contents. The parameter $R_{CC}$ represents the rate of the coded caching system, indicating the size of the server’s transmissions.

In the work [7], a placement and delivery strategy called the MAN scheme was introduced for $M = tN/K$, where $t \in [0 : K]$, achieving a rate denoted as $R_{CC}(M) = K\left(\frac{(1 - \frac{4}{K})}{1 + \frac{KM}{N}}\right)$. For other cache sizes, the lower convex envelope of points $(tN/K, R_{CC}(tN/K))_{t\in[0;K]})$ can be attained through memory sharing.

Let $R_{CC}^L(M)$ be the minimum achievable rate for a coded caching problem with $K$ users, $N$ files, and cache size $M$. It has been proven in [8] that the rate achieved by the MAN scheme is optimal up to a multiplicative factor of 4, i.e., $R_{CC}^*(M) \geq R_{CC}(M)/4$. Moreover, it is demonstrated in [9] that for worst-case demands, the rate $R_{CC}(M)$ is optimal when $N \geq K$, assuming uncoded placement. Denoting the optimal rate for a coded caching problem with uncoded placement as $R_{CC}^u(M)$, we have $R_{CC}(M) = R_{CC}^u(M)$.

C. Multi-Access Coded Caching

In a multi-access coded caching scheme, a server has $N$ files $\{W_n\}_{n\in[N]}$ each of unit size and there are $K$ users connected via an error-free shared link to the server. There are $C$ helper caches, each capable of storing $M$ units of files. Each user can access a subset of helper caches. Let $Z_k \subseteq [C]$ be the indices of the helper caches, user $k$ has access to. These cache-user connectivities define the network topology. The system operates in two phases.

Placement Phase: In this phase, all $C$ helper caches are filled without the knowledge of the future demands of the

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
users. Let $Z_c$ denote the content stored in helper cache $c \in [C]$. The content stored in these helper caches is a function of the files $W_{[1,N]}$, and all the servers know the content stored at each of the helper caches.

**Delivery Phase**: In this phase, each user wishes to retrieve a file from the server. User $k$ chooses $d_k \in [N]$ and wants to retrieve $W_{d_k}$. Users convey their demands to the server, and the server performs coded transmissions of size $R$ units such that users get their demanded file using the transmissions and the cache content they have access to.

The quantity $R$ is known as rate, and the goal of the coded caching problem is to minimise the rate. We consider that the servers are broadcasting to the users, and the users are downloading all the transmissions performed by the servers. Therefore the download cost for the users is $KR$, and reducing the download cost for the users is the same as reducing $R$. The rate of multi-access systems depends on the parameters $K$, $C$, $M$ and $N$, and the cache-user connectivities. In this paper, we consider the following two cache-user connectivities:

1) **Generalized Combinatorial Topology** [13]: In generalized combinatorial topology [13], any set of $L$ cache nodes is uniquely assigned to $K_L$ users for all $L \in [0 : C]$. This implies that the total number of users in the system is

$$K = \sum_{i=0}^{C} K_i \binom{C}{i}. \quad (2)$$

In generalized combinatorial topology, there are $K_L \binom{C}{L}$ users that are connected to exactly $L$ cache nodes and $K_L$ users are served by any one set of $L$ caches. This generalized combinatorial topology can be described using a $C + 1$ length vector of integers $K_{comb} = (K_0, \ldots, K_C)$. For example, in a generalised combinatorial topology with $C = 4$ cache nodes and $K_{comb} = (0, 0, 2, 3, 0)$, every set of 2 caches is connected to $K_2 = 2$ users and every set of 3 caches is connected to $K_3 = 3$ users. No user is connected to all four cache nodes and there is no user that isn’t connected to any cache and no user is connected to only one cache node as shown in Figure 1. There are $2\binom{4}{2} + 3\binom{4}{3} = 24$ users in this setting.

2) **Multi-Access System of [12]**: The multi-access setup of [12] is a special case of generalised combinatorial topology where $K_L = 1$ for some $L \in [C]$ and $K_c = 0, \forall c \in [C] \setminus \{L\}$. In this setup, there are $\binom{C}{L}$ users and every user is connected to a unique set of $L$ cache nodes where $L$ is called the cache access degree.

A placement and delivery scheme is given in [12] for the multi-access setup described in it. This placement and delivery scheme generalizes the well known MAN scheme for dedicated cache system, i.e. for $L = 1$ and $C = K$ case. Every file is divided into $\binom{C}{L}$ non-overlapping subfiles of equal size and the server transmits $\binom{C}{L+1}$ such subfiles, where $t = CM/N \in \mathbb{Z}$. The rate achieved in this scheme is $\frac{C}{\binom{C}{L}}$. In [13] this rate is shown to be optimal under the assumption of uncoded cache placement. We denote this rate by

$$R^*_{nPIR}(t) = \frac{C}{\binom{C}{L}}. \quad (3)$$

where the subscript $nPIR$ is to indicate that the rate is for multi-access setup with no PIR constraints.

Also in [13] multi-access system with generalised combinatorial topology is considered. As stated earlier, generalised combinatorial topology generalize the multi-access system of [12] for the special case of $K_{comb} = (0, \ldots, 0, 1, \ldots, 0)$. For any general $K_{comb}$, it is shown in [13] that a basic TDMA-like approach for treating groups of users with different cache-connectivity capabilities is optimal. For instance, for any fixed $L \in [0 : C]$ and $K \in [K_L]$, the achievable scheme given in [13] propose the same transmissions as described in [12] for given $C$, $N$ and $L$. Therefore, for generalised combinatorial topology characterized by some $K_{comb} = (K_0, \ldots, K_C) \in \mathbb{Z}^{C+1}_+$ with $C$ cache nodes, $N$ files, and $K = \sum_{L \in [0 : C]} K_L \binom{C}{L}$ users, such that $N \geq K$, each cache node capable of storing $M = \frac{tN}{C}$ files for some $t \in [0 : C]$ and assuming uncoded cache placement, the optimal rate is given by

$$R^*_{get}(t) = \sum_{L=0}^{C-t} K_L \frac{C}{\binom{C}{L}}. \quad (4)$$

Multi-User Private Information Retrieval (MuPIR) problem has been recently studied with coded caching setup [10]. MuPIR has multiple users, each equipped with a cache and multiple servers. Each user wants to retrieve a file from the servers but does not want the servers to get any information about the demand.

**D. Our Contributions**

In this paper, we develop a PIR scheme in which multiple users aided with multi-access cache nodes privately retrieve data from multiple servers. We consider the multi-access setup of [12] but with multiple non-colluding servers with all the messages replicated across the servers. Each server is connected to all the users via broadcast links. The organisation and our contributions are as follows:

- In Section II we describe the system model in detail and provide a formal description of privacy and correctness constraints.
In Section III we present our main result in Theorem 1 that gives an achievable rate for the multiuser PIR problem with generalised combinatorial topology multi-access setup of [13]. It is shown in Theorem 2 that the rate achieved in Theorem 1 is order optimal within a multiplicative factor of 2 assuming uncoded storage. In Corollary 1 we present an achievable rate for the multiuser PIR problem with multi-access setup of [12].

Also in Section III, we compare the per-user rate of our setup with dedicated cache setup of [10] in the following settings:

1) Firstly, we consider the same number of caches and same cache sizes in both dedicated cache and multi-access system. In this setting, we see that the multi-access system performs better than the dedicated cache system in terms of per-user rate as the multi-access system supports a higher number of users, and each user of the multi-access system accesses more cache memory than the dedicated cache system.

2) Then we consider systems with the same number of caches and equal amount of memory accessed by the users in both the systems. In this setup, the multi-access system is at a disadvantage as all users in a dedicated cache setup access independent cache content, whereas users of multi-access setup access dependent cache contents. Coupled with smaller cache sizes, this results in poor performance for the multi-access system. But we have highlighted a class of parameters for which the multi-access system is performing similarly to the dedicated cache system in terms of per-user rate and, at some points performing better than the dedicated cache setup.

3) Lastly we consider systems with equal number of users and same total system memory: In this setting, the multi-access system performs better than the dedicated cache system because for supporting the same number of users as a multi-access setup, using the same total system memory, the size of the individual cache in a dedicated cache system has to be reduced, and every user gets a smaller share of total memory compared to the multi-access system.

Section IV contains our achievable scheme, along with proof of privacy, achieving rates mentioned in Theorem 1 and Corollary 1. The achievable scheme of Theorem 1 is given in Section IV-B which applies to Corollary 1 also as a special case. Also, the scheme described in Section IV incurs lower upload cost per user than the scheme proposed in dedicated cache setup of [10].

II. SYSTEM MODEL

The system model considered in this paper is shown in Figure 2. There are \( N \) independent files, \( \{W_n\}_{n \in [N]} \) each of unit size, replicated across \( S \geq 1 \) non-colluding servers and \( C \) cache nodes. Each cache node is capable of storing equivalent to \( M \) files, where \( M \) can take any real number value between 0 and \( N \). There are \( K \) users, and each user is connected to a subset of these caches, and these connections define the network topology and connectivity. Let the cache nodes to which user \( k \) is connected be indexed by the set \( \mathcal{L}_k \subseteq [C], \forall k \in [K] \).

As an example, a multi-access setup with \( S = 2 \) servers \( C = 4 \) cache nodes and \( L = 2 \) cache nodes per user is shown in Figure 2. This multi-access system can be characterized by \( K_{\text{same}} = (0, 0, 1, 0, 0, 0) \).

The coded caching setup works in the following two phases,

1) Placement Phase: In this phase, all \( C \) cache nodes are filled. Let \( Z_c \) denote the contents stored in cache \( c \in [C] \) which is a function of the files \( W_{[1:N]} \), i.e.,

\[
H(Z_c|W_1,\ldots,W_N) = 0, \forall c \in [C],
\]

and all the servers know the contents stored at each of the helper caches. In Corollary 1 we present an achievable rate for the multi-access system with generalised combinatorial topology multi-user PIR problem with dedicated cache setup of [10]. Also in Section III, we compare the per-user rate of our scheme proposed in dedicated cache setup of [10].

The achievable scheme of Theorem 1 is given in Section IV-B which applies to Corollary 1. The achievable scheme of Theorem 1 is order optimal within a multiplicative factor of 2 assuming uncoded storage. The rate achieved in Theorem 1 is order optimal within a multiplicative factor of 2 assuming uncoded storage. It is shown in Theorem 2 that the rate achieved in Theorem 1 is order optimal within a multiplicative factor of 2 assuming uncoded storage. It is shown in Theorem 2 that the rate achieved in Theorem 1 is order optimal within a multiplicative factor of 2 assuming uncoded storage. In this setup, we see that the multi-access system performs better than the dedicated cache system because for supporting the same number of users as a multi-access setup, using the same total system memory, the size of the individual cache in a dedicated cache system has to be reduced, and every user gets a smaller share of total memory compared to the multi-access system.

Note that the rate in PIR must remain unaffected by user demands. Any variation in the rate \( R \) based on user demands could reveal information about the demands. Our goal is to design cache placement schemes and private delivery schemes that satisfy the privacy constraint (5) and the correctness constraint (6) that minimizes the rate \( R \).

Sending the queries \( Q_{[S]}^d \) to the servers also incurs an upload cost for the users. The upload cost should also be reduced.
Section I-C.1, generalised combinatorial topology is characterized by a \([C + 1]\) length vector \(K_{comb} = (K_0, \ldots, K_C)\) where \(K_L \in \mathbb{N}, \forall L \in \{0 : C\}\). For every \(L\), there are \(K_L\) sets, each containing \(\binom{C}{L}\) users, and in each set, every user is served by a unique set of \(L\) cache nodes. Consequently, the number of users in generalised combinatorial topology is given by \(K = \sum_{L=0}^{C} K_L\binom{C}{L}\).

2) Multi-Access Setup of [12]: As described in Section I-C.2, multi-access setup of [12] is a special case of generalised combinatorial topology. This multi-access setup consists of \(\binom{C}{L}\) users for some \(L \in \{C\}\). Each user is connected to a unique set of \(L\) cache nodes.

### III. Main Results

In this section, we present the main results of the paper. For a given multi-access cache aided PIR problem with generalized combinatorial topology, we provide a scheme to achieve the rate described in Theorem 1. The scheme is described in Section IV.

**Theorem 1:** For a multi-access coded caching setup, with \(S\) servers, \(N\) files, \(C\) helper caches and generalized combinatorial topology with \(K = (K_0, K_1, \ldots, K_C)\), and each cache can store \(M\) files and \(t = \frac{CM}{N}\) is an integer, the users can retrieve their required file privately, i.e. without revealing their demand to any of the servers, with rate \(R(t) = R_{gct}^*(\frac{MC}{N}) \times R_{PIR}(S, N)\) where \(R_{gct}^*(\frac{MC}{N})\) is given by (4) and \(R_{PIR}(S, N)\) is given by (1).

**Proof:** A scheme, along with the proof of privacy for the scheme, is given in Section IV-B that achieves the rate stated above. \(\square\)

Theorem 1 gives achievable rate in multi-access setup where cache memory \(M\) is integer multiple of \(N/C\). For intermediate memory points lower convex envelope of points \(\{(t, R(t))\}_{t \in [0:C]}\) can be achieved using memory sharing.

Next, we show that the rate achieved in Theorem 1 is order optimal within a factor of 2 under the assumption of uncoded cache placement.

**Theorem 2:** Under the assumption of uncoded cache placement, and considering \(N \geq K\) and \(S \geq 2\), the rate achieved in Theorem 1 is less than or equal to twice the optimal worst-case rate \(R^*(t)\) i.e. \(R(t) \leq 2R^*(t)\).

**Proof:** The optimal rate achieved in multi-access coded caching without PIR constraint can only be as high as the optimal rate achievable in a multi-access coded caching setup with PIR constraint. So using (4) we have,

\[
R^*(t) \geq R_{gct}^*(t) \Rightarrow \frac{R^*(t)}{R(t)} \geq \frac{R_{gct}^*(t)}{R(t)} \Rightarrow R(t) \leq 2R^*(t) \leq 1 + \frac{1}{S} + \ldots + \frac{1}{S^{C-1}}.
\]

As \(1 + \frac{1}{S} + \ldots + \frac{1}{S^{C-1}} \leq 2\) for all \(S \geq 2\) we have \(R(t) \leq 2R^*(t)\). \(\square\)

**Remark I:** In \(R(t) \leq 2R^*(t)\), note that the upper bound of \(2R^*(t)\) is met only when \(S = 2\) and \(N \to \infty\). For most other cases we have that \(R(t) < 2R^*(t)\). For instance, if \(S = 10\) and \(N = 100\) then \(R(t) \approx 1.1 R_{comb}(t)\) which is only
at most 10% higher than optimal rate. For the special case when \( S \to \infty \), optimal rate is achieved. For the trivial case of \( S = 1 \) server, we have \( R(t) \leq NR^t(t) \).

As the coded caching system of [12] is a special case of generalised combinatorial topology with \( K_{comb} = (0, \ldots, 0, 1, 0, \ldots) \), the order optimal rate for this setting is provided in Corollary 1.

**Corollary 1:** For a multi-access coded caching setup, with \( S \geq 2 \) servers, \( N \) files, \( C \) helper caches and \( K = (\frac{S}{2}, \ldots, \frac{S}{2}) \) users, where each user is accessing a unique set of \( L \) helper cache and each cache can store \( M \) files, and \( t = \frac{CM}{N} \) is an integer, the users can retrieve their required file privately, i.e. without revealing their demand to any of the servers, with rate \( R(t) = R_{nPIR}^*(\frac{SN}{C}) R^*_{PIR}(S, N) \), where \( R_{nPIR}^*(\frac{SN}{C}) \) is given by (3). Under the assumption of uncoded placement and considering \( N \geq K \), this rate is also order-optimal within a multiplicative factor of 2.

**Proof:** The corollary is straightforward as the multi-access system is a special case of generalised combinatorial topology whose rate is characterized in Theorem 1. \( \square \)

### A. Comparison With the Dedicated Cache Setup of [10]

We compare our scheme with the Product Design given in [10]. First, we will provide a brief summary of the Product Design.

1) **Product Design [10]:** In the dedicated cache setup of [10] there are \( N \) files \( \{W_n\}_{n \in [N]} \) replicated across \( S \) servers and \( K \) users, each equipped with a dedicated cache capable of storing \( M \) files. Users want to retrieve their desired files from the servers. The system works in two phases. In the Placement Phase, each user’s cache is filled with some content. This cache content is a function of the files stored across the servers and is independent of the future demands of the users. Then in the Private Delivery Phase, each user will choose a file independently and wish to retrieve its respective file from the servers privately. For that, users will cooperatively generate \( S \) queries and send them to the servers. Servers, after receiving their respective queries, will respond with answers. Specifically, server \( s \) will broadcast \( A_s \) for all \( s \in [S] \). In [10] an achievable scheme called Product Design is given. The product design will achieve the rate \( R_{PD} \) given by

\[
R_{PD} = \frac{K - t}{t + 1} \left( 1 + \frac{1}{S} + \ldots + \frac{1}{SN - 1} \right) \text{ where } t = \frac{KM}{N}.
\]

Note that the rate achieved by the product design is the same as the rate achieved in Corollary 1 for the special case of \( L = 1 \), i.e. when every user is accessing only one file.

2) **Rate Per User or Per-User-Rate:** In a coded caching system, the number of cache nodes and each cache node’s storage capacity are important parameters. In dedicated cache systems, the number of cache nodes and the number of users supported in the networks are the same. In contrast, multi-access coded caching systems can support more users for the same number of cache nodes. So, in multi-access systems, even if the number of transmissions is more than that of a dedicated cache system, one transmission may benefit more users. So, the appropriate metric is the rate per user or per-user-rate given by \( R_L(N) \). This performance metric was first introduced in [12]. Under this metric two coded caching systems with different number of users can be compared [12]. For distinction, quantities related to the dedicated cache system will have subscript \( DC \) and multi-access setup quantities will have subscript \( MA \).

We compare our scheme with the product design in the following four settings with \( S = 2 \) servers and \( N = 100 \) files. Note that, for given a \( S \) and \( N \) we have \( R_{PIR}^*(2, 100) \approx 2 \).

1) First we compare the dedicated cache system with \( K = 4 \) users (and hence 4 cache nodes) and with two multi-user PIR system utilizing generalised combinatorial topology characterized by \( K_{comb1} = (0, 0, 3, 2, 0) \) and \( K_{comb2} = (0, 0, 1, 0, 0) \). The first multi-access system has 4 cache nodes and \( 3(\frac{S}{2}) + 2(\frac{S}{3}) = 26 \) users. There are 3 sets, each containing 6 users, and each user of the set is connected to a unique subset of 2 cache nodes, and there are 2 sets of 4 users each where all 4 users in every set are connected to a unique set of 3 caches. The multi-access system characterized by \( K_{comb2} = (0, 0, 1, 0, 0) \) consists of \( \binom{4}{3} = 6 \) users, each accessing a unique subset of 2 cache nodes. We will keep the same cache sizes in all three settings, and as there are 4 cache nodes, we will have total system memory equal for all three systems, i.e. \( t_{MA} = t_{DC} = t \).

To compare a dedicated cache system with multi-access systems with different cache access degrees, we will consider the multi-access systems characterized by \( K_{comb} \) of the form \((0, \ldots, 0, U, 0, \ldots, 0)\) for some \( U \in \mathbb{N} \). But, as we are considering per user rate, the per user rate of a multi-access system with \( K_L = U \) for some \( L \in [C] \) and \( K_c = 0 \) for \( c \not\in L \) is equal to (Theorem 1) \( \frac{U}{V(U)} \frac{1}{C} R_{PIR}^*(S, N) = \frac{V(U)}{V(U)} \frac{1}{C(N)} R_{PIR}^*(S, N) \), which is independent of \( U \). Therefore we can consider multi-access systems described in [12], which are special cases of above mentioned generalised combinatorial topology with \( U = 1 \). To compare multi-access systems with dedicated cache systems with varying cache access degree \( L \), we consider the following three settings:

2) Both dedicated cache system and multi-access system have the same number of caches, i.e. \( C \) cache nodes in both settings, and cache size is also the same in both settings, i.e. \( M_{DC} = M_{MA} = M \). In this case, there will be \( C \) users in the dedicated cache setup and \( \binom{C}{C} \) users in the multi-access setup.

3) Both the dedicated cache and multi-access systems have the same number of cache nodes, i.e. \( C \) cache nodes in both settings, but each user is accessing the same amount of memory. As users in the multi-access system are accessing \( L \) cache nodes each of size \( M_{MA} \) and in the dedicated cache system, each user is accessing only one cache of size \( M_{DC} \). We will set \( M_{DC} = L \times M_{MA} \). In this case, also, there will be \( C \) users in the dedicated cache setup and \( \binom{C}{C} \) users in the multi-access setup.

4) Number of users in both systems is the same, i.e. \( K_{MA} = K_{DC} \) and total system memory is also the
We will consider These cache-aided MuPIR systems can be characterized by a dedicated cache system and two multi-access systems. In each system there are 4 cache nodes.

same. Considering C cache nodes in multi-access system we have \( K_{MA} = \binom{C}{L} = K_{DC} \). And as the number of cache nodes in the dedicated cache system is the same as the number of users, there will be \( \binom{C}{L} \) cache nodes in the dedicated cache system. For same total memory in both settings, we want \( M_{DC} \times \binom{C}{L} = M_{MA} \times C \).

Also, note that the parameter \( t = \frac{CM}{N} \) in multi-access setup and \( t = \frac{KM}{N} \) in dedicated cache setup denote how many times the entire set of \( N \) files can be replicated across the cache. For instance, if \( t = 2 \) then cache nodes are capable of storing \( 2N \) units. Also, the system’s total memory is \( tN \) units, equal to \( KM \) for the dedicated cache system and \( CM \) for the multi-access setup. For a dedicated cache system, the number of cache nodes is always equal to the number of users \( K \).

3) Dedicated Cache System Compared to Generalised Combinatorial TopologyAided MuPIR Systems: We will consider a dedicated cache system and two multi-access systems. These cache-aided MuPIR systems can be characterized by \((0,1,0,0),(0,0,1,0),(0,0,3,2)\) and \((0,0,3,2)\) and we will use subscripts \( DC, MA1 \) and \( MA2 \) for these systems respectively.

Firstly note that \( C_{MA1} = C_{MA2} = K_{DC} = C = 4 \) i.e. 4 cache nodes in all three setups. We will also consider total memory of all the systems to be the same i.e. \( t_{MA1} = t_{MA2} = t_{DC} = t \). The number of users in first multi-access system, \( K_{MA1} = \binom{4}{3} = 6 \), and the number of users in the second multi-access system is \( K_{MA2} = 3 \binom{4}{3} + 2 \binom{4}{0} = 26 \), i.e. the number of users in the multi-access setups are higher than the dedicated cache system. So with the same amount of memory, multi-access setups can support a higher number of users than the dedicated cache setup. We will compare the rate per user for these three setups. Note that although \( t_{MA1} = t_{MA2} = t_{DC} \), users in multi-access setups are accessing either two or three cache nodes. Therefore they have access to more cache memory than users of dedicated cache settings.

From (9), the per user rate achieved in dedicated cache setup is
\[
R_{DC/K_{DC}}\left(\frac{M}{N}\right) = \frac{2}{K_{DC} M_{DC}} M_{DC} = 2 - \frac{M}{N + 1}.
\]

And from Corollary 1, the per user rate achieved in the multi-access setup characterized by \((0,0,1,0),(0,0,3,2)\) is
\[
R_{MA1/K_{MA1}}\left(\frac{M}{N}\right) = \frac{2}{K_{MA1} M_{MA1}} M_{MA1} = \frac{3}{3} \left(\frac{2+3M}{4}\right) = \frac{1}{3} \left(\frac{3+2M}{12}\right).
\]

The rate per user for all three systems is plotted in Figure 4 for different values of \( M/N \). We can see that the per-user rate of a multi-access setup is better than that of a dedicated cache setup. Due to the large number of users in a multi-access setup, a single transmission from a server can benefit more users than the number of users benefited by a single transmission in a dedicated cache setup. Also, per user rate is decreasing as the caching ratio \( M/N \) is increasing for all cache-aided setups. This is obvious as users can access more content from their caches, and servers have to transmit fewer data.

4) Same Number of Cache and Same Amount of Memory: As the number of cache nodes in a multi-access system is \( C_{MA} \) and in a dedicated cache system, the number of cache nodes is the same as the number of users \( K_{DC} \), we will consider \( C_{MA} = K_{DC} = 8 \), i.e. 8 cache nodes in both, dedicated cache and multi-access setup. We will also consider total memory of both systems are also same i.e. \( t_{MA} = t_{DC} = t \). Let the cache access degree for the multi-access system is \( L \). Note that \( K_{MA} = \binom{C}{L} \), i.e. the number of users in the multi-access setup will be higher than the dedicated cache system except for \( L = 1 \), where both systems are the same. So with the same amount of memory, a multi-access setup can support a higher number of users. We will compare the rate per user for the two setups. Note that although \( t_{MA} = t_{DC} \), users in multi-access setup are accessing \( L \) cache nodes. Therefore they have access to more cache memory than users of dedicated cache settings.
From (9), the per user rate achieved in dedicated cache setup is
$$\frac{R_{DC}}{K_{DC}} \left( \frac{M}{N} \right) = \frac{2}{K_{DC}} \frac{K_{DC} - K_{DC} M_{DC}}{K_{DC} + 1} = 2 \left( 1 - \frac{M_{DC}}{K_{DC}} \right).$$
We will also consider a multi-access setup characterized by $K_{comb}(0, 0, 3, 2, 0, 0, 0, 0, 0)$. In this system, there are $K_{MA2} = 3(3) + 2(2) = 196$ users. From Theorem 1, the per user rate achieved in this multi-access setup is
$$\frac{R_{MA2}}{K_{MA2}} \left( \frac{M}{N} \right) = \frac{2}{K_{MA2}} \left( \frac{C_{MA2}}{\left(\begin{array}{c} C_{MA2} \\ C_{MA2} \end{array}\right)} \right) + 2 \left( \frac{C_{MA2}}{\left(\begin{array}{c} C_{MA2} \\ C_{MA2} \end{array}\right)} \right).$$

The per user rate for all the systems is plotted in Figure 8 for 8 cache nodes in each setting for different values of $M/N$ and $L$. We can see that the per-user rate of the multi-access setup is better than that of the dedicated cache setup. The ratio of per-user rates is plotted in Figure 6. Due to the large number of users in the multi-access setup, a single transmission from a server can benefit more users than the number of users benefited by a single transmission in the dedicated cache setup. We will see that in the multi-access system of [12], a single transmission is simultaneously used by $(L+t)$ users, compared to $t+1$ in the dedicated cache setup.

5) Same Number of Cache and Same Memory per User:
As we saw, users in multi-access setup are accessing more memory than users of dedicated cache setting if $C_{MA} = K_{DC}$ and $t_{MA} = t_{DC}$. Now, we will reduce the storage capacity in the multi-access setup so that amount of memory accessed by users of the dedicated cache setup and multi-access setup is the same. We consider that $C_{MA} = K_{DC} = 8$, but now the storage capacity of cache nodes in the multi-access setup is smaller than that of the dedicated cache system so that each user is accessing the same amount of memory. For that we set $M_{DC} = L \times M_{MA}$ because every user in multi-access setup is accessing $L$ cache nodes. Therefore we have $t_{DC} = L \times M_{MA} \Rightarrow t_{MA} = t_{DC}/L$ i.e. the total system memory in a multi-access setup is $L$ times lesser than that of a dedicated cache system. Now, although each user has access to the same amount of memory, each user of the dedicated cache system has access to a cache whose content is independent of content stored in the cache of other users. Whereas cache content accessed by the users of multi-access setup is not independent of cache content accessed by other users, therefore the rate of dedicated cache setup will be better than the rate of multi-access setup.

Again, from (9), the per user rate achieved in dedicated cache setup is
$$\frac{R_{DC}}{K_{DC}} \left( \frac{M}{N} \right) = \frac{2}{K_{DC}} \frac{K_{DC} - K_{DC} M_{DC}}{K_{DC} + 1} = 2 \left( 1 - \frac{M_{DC}}{K_{DC}} \right).$$
And from Corollary 1, the per user rate achieved in the multi-access setup with cache access degree $L$ will be
$$\frac{R_{MA}}{K_{MA}} \left( \frac{M_{DC}}{N} \right) = \frac{R_{MA}}{K_{MA}} \left( \frac{M_{DC}}{L \times N} \right) = 2 \left( \frac{L^{C_{MA}}}{} \right).$$

In Figure 7, we compare the per user rate for cache-aided systems for various cache access degrees. Note that when $L = 1$, we have a dedicated cache system of [10]. Although even for the per-user rate, we see that the dedicated system is performing better than the multi-access setup for most of the cases, we see that for cases when $M_{DC}/N = 4$ and $L = 2$, we have $M_{DC}/N = 0.5$ and $R_{DC}/K_{DC} = \frac{1}{10}$ whereas $R_{MA}/K_{MA} = \frac{1}{11.2}$. Therefore, there also exist some points where the per user rate of the multi-access setup is better than that of the dedicated cache setup. For instance, in Figure 8, we plot the ratio of per user rates for the dedicated cache system and multi-access systems with different cache access degrees. When $t_{DC} = 4$ and $L = 2$ we have $M_{DC}/N = 0.5$ and $R_{DC}/K_{DC} = \frac{1}{10}$ whereas $R_{MA}/K_{MA} = \frac{1}{11.2}$.

6) Same Number of Users and Same Total System Memory:
We consider a multi-access systems with $C_{MA} = 8 = C$ cache nodes and different cache access degrees $L$. We keep the number of users in the dedicated cache setup be same as
the number of users in the multi-access setup. So \( K_{DC} = K_{MA} = \binom{C_{MA}}{L} \) i.e. we will consider a different dedicated cache setup for every \( L \). Also we will keep total memory in all the systems same i.e. \( t_{MA} = t_{DC} = t \). Note that the number of cache nodes in the dedicated cache system is \( K_{DC} = \binom{C_{MA}}{L} \). For the same total memory, we want that \( C_{MA}M_{MA} = K_{DC}M_{DC} \implies M_{DC} = \frac{C_{MA}}{C_{MA}}M_{MA} \). So, the size of individual cache nodes in the dedicated cache system is now smaller than that of the multi-access system. Then again, users of a multi-access setup can access more than one cache node. Therefore, we see that the dedicated cache system has two disadvantages. For supporting the same number of users as a multi-access setup, using the same total system memory, the size of individual cache have to be reduced, and every user gets a smaller share of total memory compared to the multi-access system.

Therefore, the per user rate achieved in a dedicated cache systems with \( \binom{C}{L} \) users will be

\[
R_{DC}\left( \frac{M_{MA}}{N}, L \right) = \frac{2\binom{C_{MA}}{N}}{L\binom{C}{L} + 1} = \frac{2}{L}\frac{8M_{MA}}{N} + 1.
\]

For a multi-access system with cache access degree \( L \), the per user rate is

\[
R_{MA}\left( \frac{M_{MA}}{N}, L \right) = \frac{2\binom{C_{MA}}{L} + \frac{C_{MA}}{L}}{2\binom{C_{MA}}{L} + \frac{8M_{MA}}{N}}.
\]

The comparison of the per-user rate for this setting is shown in Figure 9. For every value of \( L \), we’ve considered a dedicated cache setup with \( n \) users and a multi-access setup with cache access degree \( L \). The number of users in both systems is the same. Then we considered \( M_{DC} = M_{MA}C_{MA}/L \) and plotted per user rates for both the settings for \( M_{MA}/N \in [0,1] \). Note that because the number of users in both settings is kept equal, the ratio of the per user rate will also be the same as the rate in both settings. The ratio of per-user rates is plotted in Figure 10.

We can see that the multi-access system is performing better than the dedicated cache setup while utilizing the same amount of cache memory and serving the same number of users. For instance, in Figure 9, consider \( L = 4 \). The number of users in the dedicated cache system as well as in the multi-access system is \( \binom{70}{4} \) users. When \( M_{MA}/N = 0.25 \) we have \( t = 2 \). In the dedicated cache system, each cache is of size \( M_{DC}/N = 1/35 \) whereas in the multi-access setup each cache is of size \( M_{MA}/N = 0.25 \) and every user is accessing 4 caches. As \( M_{MA}/N \) is increasing, the rate for both dedicated cache setup and multi-access setup is increasing. But, we can see even for \( M_{MA}/N = 1 \), we have \( M_{DC}/N = 8/70 \), thus dedicated cache setup is incurring a very high rate for supporting the same number of users with the same amount of total memory.

7) Upload Cost: We also consider the per-user upload cost. As uploading random queries to the servers incurs some upload cost, we compare the per-user upload cost incurred in product design proposed in [10] and the achievable scheme proposed in Section IV-B for generalised combinatorial topology with arbitrary \( K_{comb} \).
Firstly, in the product design proposed in [10], every user generates \( (K-1) \) queries, where the size of each query is the same as the query in single-user PIR. Therefore, the upload cost incurred by every user is given by \( N\hat{S}(K-1)\log_2 \left( \frac{S}{N^n} \right) \) bits. Whereas, in the scheme provided in Section IV-B, every user incurs an upload cost of \( N\hat{S}(K-1)\log_2 \left( \frac{S^{K-1}}{N^n} \right) \) bits which is same as the upload cost for the user in single user PIR of [2]. Therefore, in the query construction provided in Section IV-B users need to upload \( (K-1) \) times lower data in comparison to the product design proposed in [10].

IV. SCHEME DESCRIPTION

A. Example

We will first understand the proposed scheme with an example. Consider a multi-access setup with \( S = 2 \) non-colluding servers storing \( N = 3 \) files \( W_1, W_2 \) and \( W_3 \) each of unit size. There are \( C = 5 \) cache nodes, each capable of storing \( M/N = 2/5 \) units. The caching system is characterized by \( K_{\text{comb}} = (0, 0, 1, 1, 0, 0) \). This means there is a set of 10 users each connected to a unique set of 2 cache nodes, and there is a set of 5 users each connected to a unique set of 3 cache nodes. We will index the users with the indices of cache nodes they are connected to; for example, user \( [2, 4] \) is connected to cache node 2 and cache node 4 and user \( [1, 2, 3] \) is the user connected to cache nodes 1, 2 and 3. The caches will be filled in the placement phase as follows.

1) Placement Phase: Let \( t = \frac{C\log N}{M/N} = 2 \). Divide each file into \( \binom{C}{t} = \binom{5}{2} = 10 \) subfiles as \( W_n = \{W_{n,T} | T \in [5] \} \). Then fill cache node \( c \in [5] \) as \( Z_c = \{W_{n,T} | n \in [N], \forall T \in [5], \text{such that } c \in T \} \).

2) Delivery Phase: Now, every user will choose a file index and want to retrieve the file from the servers privately. Let demand of user \( K \in \{0, 2\} \cup \{3\} \) is \( d_K \). Then the demand vector is \( d = (d_K)_{K \in \{0, 2\} \cup \{3\}} \) and users want to hide this demand vector from the servers. For this, each subfile will be further divided into \( S^{N} = 8 \) sub-subfiles, and the users will generate 2 queries \( Q^1 \) and \( Q^2 \) one for each server as follows.

Every user generates three random permutations of the set \( \{S^{N}\} = [8] \), one for each file. Using these random permutations, users will generate queries \( Q^K, d_K, \forall K \in [5] \), one for each server where \( Q^K, d_K \) is query sent to server \( s \) in single user PIR setup, with \( S \) servers and \( N \) files if demand of the user is \( d_K \). For instance, consider user \( K = \{1, 2, 3\} \) (the user connected to cache nodes 1, 2 and 3) and let \( d_{\{1, 2, 3\}} = 1 \) then this user generates three random permutations of \( \{S^{N}\} = [8] \), one corresponding to each file. Let these permutations be \( \{a_1, a_2, b_1, b_2, c_1, c_2\} \) for files \( W_1, W_2 \) & \( W_3 \) respectively. Now, as user \( \{1, 2, 3\} \) wants file \( W_1 \) then the query \( Q^1_{\{1, 2, 3\}, 1} \) will be a list of integers as given in Table II.

An independently random permutation of \( [1 : 8] \) is chosen by every user \( K \in \{0, 2\} \cup \{3\} \) and using this permutation, every user generates \( Q^K, d_K \) for every server \( s \). Therefore, query \( Q^1 \) is \( \{Q^1_{K, d_K} | K \in \{0, 2\} \cup \{3\} \} \) is sent to server 1 and \( Q^2 \) is \( \{Q^2_{K, d_K} | K \in \{0, 2\} \cup \{3\} \} \) is sent to server 2.

![Table I](image1)

| Server 1 | Server 2 |
|----------|----------|
| \( a_1, b_1, c_3 \) | \( a_2, b_2, c_2 \) |
| \( a_3, b_2 \) | \( a_4, b_1 \) |
| \( a_6, c_1 \) | \( b_3, c_4 \) |
| \( a_7, b_4, c_4 \) | \( a_8, b_3, c_3 \) |

![Table II](image2)

| Server 1 | Server 2 |
|----------|----------|
| \( W_{a_1}, (4, 5), W_{a_2}, (4, 5), W_{c_1}, (4, 5) \) | \( W_{a_3}, (4, 5), W_{b_2}, (4, 5), W_{c_2}, (4, 5) \) |
| \( W_{b_3}, (4, 5), W_{c_3}, (4, 5) \) | \( W_{a_4}, (4, 5), W_{b_1}, (4, 5) \) |
| \( W_{a_5}, (4, 5), W_{c_1}, (4, 5), W_{a_6}, (4, 5), W_{b_3}, (4, 5) \) | \( W_{a_7}, (4, 5), W_{b_2}, (4, 5), W_{c_4}, (4, 5) \) |

receiving these queries server \( s \), for \( s \in [2] \), will transmit \( A^d_c(Q^K, d_K, W_{[s]}, S) \) for \( S = \{1, 2, 3, 4, 5\} \); \( K \in \{3\} \) and \( A^d_c(Q^K, d_K, W_{[s]}, S) \) for \( \forall S \in \{5\} \). Where \( A^d_c(Q^K, d_K, W_{[s]}, S) \) is answer of server \( s \) in single-user PIR setup if query is \( Q^K, d_K \) and set of files is \( W_{[s]} \).

Again, considering \( S = \{1, 2, 3, 4, 5\} \) and \( K = \{1, 2, 3\} \), answers from the servers, \( A^d_1(Q^1_{1, 2, 3}, W_{[3]}, [4, 5]), A^d_1(Q^1_{1, 2, 3}, W_{[3]}, [4, 5]) \), is given in Table II. After receiving the broadcast from the servers, every user will be able to decode their desired subfile. Again considering the case of user \( \{1, 2, 3\} \), it has access to all subfiles of all files indexed by \( \{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\} \) and \( \{3, 4\} \). The only missing subfiles are those indexed by \( \{4, 5\} \). Now consider again the transmission of the server for \( S = \{1, 2, 3, 4, 5\} \); \( K \in \{3\} \) and \( A^d_3(Q^3_{1, 2, 3}, W_{[3]}, S) \) is answer of \( s \) in single-user PIR setup if query is \( Q^3_{1, 2, 3} \) and set of files is \( W_{[3]} \).

User will get all sub-subfiles of \( W_{1, 4, 5} \) from these remaining terms. Similarly, all users will get the missing subfiles of the demanded file. Also, to generate \( Q^K, d_K \), for every user \( K \), independent random permutations of \( [8] \) is chosen. Then, from the privacy of the single-user PIR scheme, servers will get no information about the user demands from the queries they get.

Also note that, as each server is transmitting 7 sub-subfiles each of size \( \frac{1}{108} \) units for every \( s \in \{5\} \cup \{\{5\} \} \), the rate in this example is \( R = \frac{42}{50} = 0.85 \) units, and the subpacketization level is 80. Also, note that every user sends 12 random integers to each server, each integer from the set \([8]\).

B. General Description: Generalised combinatorial topology

Consider \( N \) unit size files \( \{W_n\}_{n \in [N]} \) replicated across \( S \) servers. There are \( C \) cache nodes, each capable of storing
M units such that $CM/N$ is an integer. There are $K$ users each accessing the caches using generalised combinatorial topology characterized by $K_{comb} = (K_0, K_1, \ldots, K_C)$. Therefore, $K = \sum_{l=0}^{C} K_l \binom{C}{l}$. There are $\binom{C}{l}$ sets of $K_l$ users in which each user is accessing a unique set of $L$ cache nodes. So, we will index each user with an ordered pair $(u, K)$ where $u \in \mathbb{Z}$ and $K \subseteq [C]$. Specifically, user $(u, K)$, where $u \in [K_l]$ and $K \in \binom{C}{l}$ for some $L \in [0 : C]$, is $u$-th user connected to cache nodes indexed by $K$. For every $u \in [K_l]$, let $\mathcal{U}_{u,l}$ be the $u$-th set of those $\binom{C}{l}$ users that are connected to $L$ cache nodes for some $L \in [0 : C]$. Let $\mathcal{U}_{L}$ be the set of users connected to exactly $L$ caches, and $\mathcal{U}$ be the set of all users. That is, $\mathcal{U}_{u,l} \equiv \{(u, K)|K \in \binom{C}{l}\}$, $\mathcal{U}_{L} \equiv \bigcup_{u \in [K_l]} \mathcal{U}_{u,L}$ and $\mathcal{U} \equiv \bigcup_{L \in [0:C]} \mathcal{U}_{L}$.

1) Placement Phase: Let $t = \frac{CM}{d}$ be an integer. Then divide each file into $\binom{C}{l}$ subfiles, each indexed by a $t$ sized subset of $[C]$ as follows: $W_c = \{W_{n,T}|T \in \binom{C}{l}\}$. Then fill cache node $c$ with $Z_c = \{W_{n,T}|c \in T, T \in \binom{C}{l}\}$.

2) Delivery Phase: In this phase, every user will choose one of the file indexes independently and randomly. Let user $(u, K) \in \mathcal{U}$ choose index $i(u, K) \in [N]$. User $(u, K)$ will then wish to retrieve file $W_{d(u,K)}$ from the servers without revealing the index of the demanded file to the servers. Let, $d_{u} \equiv (d(u,K), i(u, K)) \in \{\binom{C}{l}\}$ be the demands of all users connected to exactly $L$ cache nodes, then $d = (d_0, d_1, \ldots, d_C)$ will be the demand vector.

Users do not want the servers to get any information about the demand vector. For privately retrieving the files, users will cooperatively generate $S$ queries $Q^d_s, \forall s \in [S]$ as follows.

Each user generates $N$ random permutations of integers in $[SN]$, one random permutation for every file. Each user generates $S$ sub-queries using these random permutations, one for every server. Specifically, user $(u, K) \in \mathcal{U}$ generates sub-queries $Q^d_s(u,K), \forall s \in [S]$. Here, $Q^d_s(u,K)_{d(u,K)}$ is the query sent by the user of single-user PIR setup of [2] to server $s \in [S]$ if the demanded file is $d(u,K) \in [N]$. After every user constructs these sub-queries, the query $Q^d$ will be sent to server $s$ for every $s \in [S]$ where, $Q^d \equiv \{Q^d_s(u,K)_{d(u,K)}|(u, K) \in \mathcal{U}\}$. For every $L \in [0 : C]$ and $u \in [K_l]$, we define $d^s_{u,L} \equiv (d(u,K))_{K \in \binom{C}{l}}$ and $Q^d_{u,L} \equiv \{Q^d_s(u,K)_{d(u,K)}|K \in \binom{C}{l}\}$. After receiving their respective queries, servers will construct answers using these queries and the files. For every $L \in [0 : C]$ and every $u \in [K_l]$, server $s \in [S]$ computes the following quantity:

$$A^d_{u,s,L} \equiv \bigoplus_{K \in \binom{C}{l}} A^d_{u,K} \left(Q^d_s(u,K)_{d(u,K)}, W_{[N],S\setminus K}\right)$$  \(12\)

where $A^d_{u,K} \left(Q^d_s(u,K)_{d(u,K)}, W_{[N],S\setminus K}\right)$ is the answer computed by the server $s$ in single-user PIR setup of [2] if the query received is $Q^d_s(u,K)_{d(u,K)}$ and the set of files is $W_{[N],S\setminus K}$. Note that in order to compute this quantity, each subfile needs to be further divided into $SN$ subpackets. Defining $A^d_{s,L} \equiv \{A^d_{u,s,L} | S \in \binom{C}{l}\}$, server $s$ broadcasts: $A^d_s \equiv \bigcup_{L \in [0:C]} \bigcup_{u \in [K_l]} A^d_{u,s,L}$.

Decoding: Firstly, users who access more than $C - t$ cache nodes have access to the entire set of files from the caches themselves. So, no decoding is required for these users. Now consider user $(u, K) \in \mathcal{U}_{u,L}$ for $L < C - t$ (i.e. $u$-th user connected to $L$ cache nodes indexed by $K$) and subfile $T \in \binom{C}{k}$, if $K \cap T \neq \emptyset$ then subfile $W_{d(u,K)\cup T}$ is available to user from the cache. If $K \cap T = \emptyset$, the subfile must be decoded from the transmissions. As $K \cup T \subseteq \binom{C}{l}$, consider the quantity $A^d_{u,s,K\cup T}(u) \in A^d_s$. From (12):

$$A^d_{u,s,K\cup T}(u) = \bigoplus_{K' \subseteq (K \cup T)\setminus K} \left(\bigoplus_{K'' \subseteq K'} A^d_{s,K''} \left(Q^d_s(u,K'')_{d(u,K'')}, W_{[N],K''\cup T}\right)\right)$$

User $(u, K)$ has access to all subfiles in the second term of RHS above, and therefore it can recover the first term from the above expression. After getting $A^d_{s,K\cup T}(u) \left(Q^d_s(u,K)_{d(u,K)}, W_{[N],T}\right)$ for all $s \in [S]$, user $(u, K)$ can recover subfile $W_{d(u,K)\cup T}$ from the transmissions as guaranteed from the correctness of single user PIR scheme.

Rate: Each server is performing $\binom{C}{l+1}$ transmissions each of size $\frac{1}{|\phi_s|}\left(\frac{1}{S} + \frac{1}{S^2} +\ldots + \frac{1}{S^{N-1}}\right)$ units for every $Q^d_{s,K} \forall u \in [K_l], \forall L \in [0 : C - t]$. So the rate of the achievable scheme is

$$R(t) = \sum_{L \in [0:C-t]} K_L \left(\frac{C_L}{t^L}\right) \left(1 + \frac{1}{S} + \frac{1}{S^2} + \ldots + \frac{1}{S^{N-1}}\right).$$

Subpacketization: As we can see, each file is divided into $\binom{C}{l}$ subfiles, each of which has to be further divided into $SN$ sub-subfiles. So subpacketization level is $\binom{C}{l} \times SN$.

Upload Cost: User $(u, K) \in \mathcal{U}$ is sending $Q^d_s(u,K)_{d(u,K)}$ to server $s, s \in [S]$. Each $Q^d_s(u,K)_{d(u,K)}$ is the same query (upto the random permutations of $[SN]$) sent by the user of single-user PIR problem. Therefore the upload cost for each user is the same as the upload cost for the user of single user PIR scheme of [2], which is $SN \log_2 \left(\frac{SN+1}{SN-1}\right)$ bits.

Proof of Privacy: Now we will prove that none of the servers will get any information about the demand vector. We will show that given any realization of query sent to server $s$ say $Q^d_s = q_s$, all possible demand vectors are equally likely. Consider

$$\mathbb{P}(d = d_{[K]}|Q^d_s = q_s) = \frac{\mathbb{P}(Q^d_s = q_s|d = d_{[K]}))\mathbb{P}(d = d_{[K]}))}{\mathbb{P}(Q^d_s = q_s)}$$  \(13\)

for some $d_{[K]} \in [N]^K$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Now \( q_s = \left\{ q_s(u,K) \mid (u,K) \in U \right\} \). Here \( q_s(u,K) \) is realization of query \( Q^t_s(u,K) \). In each \( q_s(u,K) \), there are \( S^{N-1} \) unique integers (subpacket indices) corresponding to each file. The indices corresponding to user demand never repeat across the servers, and a given server only sees \( S^{N-1} \) unique indices corresponding to each file in every \( q_s(u,K) \). There are \( (S^N - S^{N-1})! \) permutations that’ll result in same \( q_s(u,K) \). As all \( q_s(u,K) \) are generated independently therefore \[ P(Q^d_s = q_s | d = d_s[K]) = \frac{(S^N-S^{N-1})! N^{\sum_{L=2}^{K+1} L \choose L}}{S^{N !}} \] which does not depend on the demand vector. Substituting this value in (13), we get \[ P(d = d_s[K] | Q^d_s = q_s) = P(d = d_s[K]) = 1/N^{K!} \]. Therefore, every possible demand vector is equiprobable given the query to a server. Hence, servers will not get any information about the demand vector.

C. General Description: Multi-Access System of [12]

Consider \( N \) unit size files \( \{W_n\}_{n \in [N]} \) replicated across \( S \) servers. There are \( C \) cache nodes, each capable of storing \( M \) files and \( K \) users, each connected to a unique set of \( L \) cache nodes. We will consider a system with \( \left( \begin{array}{c} C \end{array} \right) \) users. As each user is connected to a unique set of \( L \) cache, we will index that user with a \( L \) sized subset of \( \{C\} \). Specifically, user \( K \), where \( K \in \left( \begin{array}{c} C \end{array} \right) \), is a user connected to cache nodes indexed by \( K \).

1) Placement Phase: Let \( t = \frac{SM}{C} \) be an integer. Then divide each file into \( \left( \begin{array}{c} C \end{array} \right) \) subfiles, each indexed by a \( t \) sized subset of \( \{C\} \) as \( W_n = \left\{ W_{n,T} \mid T \in \left( \begin{array}{c} C \end{array} \right) \right\} \). Then fill cache node \( c \) with \( \{W_{n,T} | c \in T, T \in \left( \begin{array}{c} C \end{array} \right) \} \).

2) Delivery Phase: Every user will choose one of the file indexes in this phase. Let user \( K \), \( \forall K \in \left( \begin{array}{c} C \end{array} \right) \) choose index \( d_K \) of the \( \left( \begin{array}{c} C \end{array} \right) \) subfiles from the server without revealing the index of the demanded file to the servers. Let \( d = (d_K)_{K \in \left( \begin{array}{c} C \end{array} \right)} \) be the demand vector. Users do not want the servers to get any information about the demand vector. For privately retrieving the files, users will cooperatively generate \( S \) queries \( Q_d^s \) as follows.

User \( K \) will generate query \( Q^t_s(K,d_K) \) for server \( s \), where \( Q^t_s(K,d_K) \) is the query sent to server \( s \) in single user PIR setup of [2] if user demand is \( d_K \). Query sent to server \( s \) is \( Q^t_s = \{Q^t_s(K,d_K)\} \). Now for every \( S \in \left( \begin{array}{c} C \end{array} \right) \), server \( s \) will transmit \( \bigoplus_{K \in \left( \begin{array}{c} S \end{array} \right)} A^s_{dc}(Q^t_s(K,d_K),W_{[N],S,K}) \) which is the answer of server \( s \) in single user PIR setup if received query is \( Q^t_s \) and set of indices is \( \{W_{[N],S,K}\} \).

Now we will see that all the users will be able to decode their required files from these transmissions and the caches they have access to.

Decoding: Consider user \( K \in \left( \begin{array}{c} C \end{array} \right) \) (i.e. the user connected to cache nodes indexed by \( K \) and subfile index \( T \in \left( \begin{array}{c} C \end{array} \right) \)). If \( K \cap T \neq \phi \), then subfile \( W_{dc,L,T} \) is available to the user from the cache. If \( K \cap T = \phi \), then the subfile has to be decoded from the transmissions. Consider transmissions corresponding to \( S = K \cup T \). The answer of server \( s \) to \( Q^t_s(K,d_K) \) is \( A^s_{dc}(Q^t_s(K,d_K),W_{[N],(K,T)\cup K'}) = A^s_{dc}(Q^t_s(K,d_K),W_{[N],T}) \bigoplus_{K' \in \left( \begin{array}{c} C \end{array} \right) \setminus T} A^s_{dc}(Q^t_s(K,d_K),W_{[N],(K,T)\setminus K'}) \) user \( K \) has access to all subfiles in the second term of RHS above, and therefore it can recover the first term from the above expression. After getting \( A^s_{dc}(Q^t_s(K,d_K),W_{[N],T}) \) for all \( s \in S \), user \( K \) can recover subfile \( W_{dc,T} \) from the transmissions.

Rate: Each server is performing \( \left( \begin{array}{c} C + L \end{array} \right) \) transmissions each of size \( \frac{1}{t} \left( \frac{N}{2} + \frac{1}{2^{t2}} + \ldots + \frac{1}{2^{tN}} \right) \) units. So rate is \( R(t) = \left( \begin{array}{c} C + L \end{array} \right) \left( \frac{1}{t} \left( \frac{N}{2} + \frac{1}{2^{t2}} + \ldots + \frac{1}{2^{tN}} \right) \right) \).

Subpacketization: As we see, each file is divided into \( \left( \begin{array}{c} C \end{array} \right) \) subfiles, each of which has to be further divided into \( S^N \) sub-subfiles. So the subpacketization level is \( \left( \begin{array}{c} C \end{array} \right) \times S^N \).

Coding Gain: We can see that transmission corresponding to each \( S \in \left( \begin{array}{c} C \end{array} \right) \) is beneficial to user \( K \) if \( K \in \left( \begin{array}{c} C \end{array} \right) \). So every transmission, from each server, is used by \( \left( \begin{array}{c} L \end{array} \right) \) users.

**Proof of Privacy:** Now we will prove that none of the servers will get any information about the demand vector. We will show that given any realisation of the query sent to the server \( s \) say \( Q^t_s = q_s \) all possible demand vectors are equally likely. Consider \( P(d = d_s[K] | Q^t_s = q_s) = \frac{P(Q^t_s = q_s | d_s[K]) P(d_s[K])}{P(Q^t_s = q_s)} \) for some \( d_s[K] \in N^{K!} \). Query \( q_s \) sent to server \( s \) consists of \( q_s = \left\{ q_{K}^s | K \in \left( \begin{array}{c} C \end{array} \right) \right\} \). Each \( q_{K}^s \) is made using an independent random permutation of \( \left( \begin{array}{c} S \end{array} \right) \). Each sub-query has \( S^{N-1} \) unique subpacket indices corresponding to each file. Now, the indices corresponding to user demand never repeat across the servers, and a given server only sees \( S^{N-1} \) unique indices corresponding to each file in every list. There are \( (S^N - S^{N-1})! \) permutations that’ll result in the same list of indices. As all \( \left( \begin{array}{c} C \end{array} \right) \) sub-queries \( q_{K}^s \) are generated independently therefore \( P(Q^t_s = q_s | d_s[K]) = (S^N - S^{N-1})! / S^{N !} \) which does not depend on the demand vector. Therefore \( P(d = d_s[K] | Q^t_s = q_s) = 1/N^{K!} = P(d = d_s[K]) \) and every possible demand vector is equiprobable given the query to a server. Hence, servers will not get any information about the demand vector.

V. CONCLUSION

In this paper, we considered the problem of multi-access cache-aided multi-user private information retrieval (MuPIR). The multi-access setup discussed in this paper, also known as the generalized combinatorial topology, generalizes the dedicated cache-aided multi-user PIR setup. We proposed an achievable scheme for the multi-access setup. The proposed scheme’s download cost (rate) is order optimal within a factor of 2, considering uncoded placement at the caches. Although the rate incurred by the multi-access cache-aided setup is higher than the rate incurred by the dedicated cache-aided setup, the multi-access setup supports a larger number of users than the dedicated cache setup for the same number of cache nodes. Therefore, we compared the rate per user of these two setups in which the performance of the multi-access setup is better than that of the dedicated cache setup.

Although the order optimal achievable scheme is provided, optimality results for multi-access cache-aided MuPIR problems remain open. Furthermore, the upload cost, the subpacketization problem and other cache-user connectivities are of interest. Conveying randomly chosen queries to the
servers incurs upload cost, which can be ignored for large file sizes. However, if upload cost is of concern, then further work is required to optimize it as well. The subpacketization problem is studied separately in the context of coded caching (without privacy constraints) and PIR. Further work is required to tackle this problem when PIR and coded caching are studied in conjunction. MuPIR schemes that simultaneously improve these parameters (upload cost, download cost and subpacketization) are still unknown.

Generalized combinatorial topology generalizes various cache-user connectivities. But other connectivities not covered under the generalized combinatorial topology are an exciting research area. Better achievable schemes and optimality results remain open for general cache-user connectivities and arbitrary system parameters.

REFERENCES

[1] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, “Private information retrieval,” in Proc. 36th IEEE Symp. Foundations Comput. Sci., Milwaukee, WI, USA, Oct. 1995, pp. 41–50, doi: 10.1109/SFCS.1995.492461.

[2] H. Sun and S. A. Jafar, “The capacity of private information retrieval,” IEEE Trans. Inf. Theory, vol. 63, no. 7, pp. 4075–4088, Jul. 2017, doi: 10.1109/TIT.2016.260029.

[3] H. Sun and S. A. Jafar, “The capacity of robust private information retrieval with colluding databases,” IEEE Trans. Inf. Theory, vol. 64, no. 4, pp. 2361–2370, Apr. 2018, doi: 10.1109/TIT.2017.2777490.

[4] H.-Y. Lin, S. Kumar, E. Rosnes, A. G. I. Amat, and E. Yaakobi, “Multi-server weakly-private information retrieval,” IEEE Trans. Inf. Theory, vol. 68, no. 2, pp. 1197–1219, Feb. 2022, doi: 10.1109/TIT.2021.3126865.

[5] Z. Chen, Z. Wang, and S. A. Jafar, “The capacity of T-private information retrieval with private side information,” IEEE Trans. Inf. Theory, vol. 66, no. 8, pp. 4761–4773, Aug. 2020, doi: 10.1109/TIT.2020.2977919.

[6] K. Banawan and S. Ulukus, “The capacity of private information retrieval from coded databases,” IEEE Trans. Inf. Theory, vol. 64, no. 3, pp. 1945–1956, Mar. 2018, doi: 10.1109/TIT.2017.2919949.

[7] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” IEEE Trans. Inf. Theory, vol. 60, no. 5, pp. 2856–2867, May 2014, doi: 10.1109/TIT.2014.2306938.

[8] H. Ghasemi and A. Ramamoorthy, “Improved lower bounds for coded caching,” IEEE Trans. Inf. Theory, vol. 63, no. 7, pp. 4388–4413, Jul. 2017, doi: 10.1109/TIT.2017.2705166.

[9] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “The exact rate-memory tradeoff for caching with uncoded prefetching,” IEEE Trans. Inf. Theory, vol. 64, no. 2, pp. 1281–1296, Feb. 2018, doi: 10.1109/TIT.2017.2785237.

[10] X. Zhang, K. Wan, H. Sun, M. Ji, and G. Caire, “On the fundamental limits of cache-aided multiuser private information retrieval,” IEEE Trans. Commun., vol. 69, no. 9, pp. 5828–5842, Sep. 2021, doi: 10.1109/TCOMM.2020.3091612.

[11] K. Vaidya and B. S. Rajan, “Cache-aided multi-access multi-user private information retrieval,” in Proc. 20th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt), Sep. 2022, pp. 246–253, doi: 10.23919/WiOpt56218.2022.9930597.

Kanishak Vaidya (Student Member, IEEE) was born in Himachal Pradesh, India. He received the B.Tech. degree in electronics and communications engineering from the Jawaharlal Nehru Government Engineering College, Sundar Nagar, Himachal Pradesh, in 2017. He is currently pursuing the integrated M.Tech. and Ph.D. degrees with the Department of Electrical Communications Engineering, IISc, Bengaluru, Karnataka, India. His primary research interests include distributed computation, private information retrieval, private information delivery, coded caching, and index coding. He was a recipient of the Prime Minister’s Research Fellowship (2020–2023).

B. Sundar Rajan (Life Fellow, IEEE) was born in Tamil Nadu, India. He received the B.Sc. degree in mathematics from Madras University, Madras, India, in 1979, the B.Tech. degree in electronics from Madras Institute of Technology, Madras, in 1982, and the M.Tech. and Ph.D. degrees in electrical engineering from IIT Kanpur, Kanpur, in 1984 and 1989, respectively. He was a Faculty Member with the Department of Electrical Engineering, IIT Delhi, New Delhi, from 1990 to 1997. He has been a Professor with the Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru, since 1998. His primary research interests include space-time coding for MIMO channels, distributed space-time coding and cooperative communication, coding for multiple-access and relay channels, and network coding.

Dr. Rajan is also a J. C. Bose National Fellow (2016–2025) and a member of the American Mathematical Society. He is a fellow of the Indian National Academy of Engineering, the Indian National Science Academy, the Indian Academy of Sciences, and the National Academy of Sciences, India. He has been a recipient of Prof. Rustum Choksi Award by IISc for Excellence in Research in Engineering in 2009, the IETE Pune Center’s S. V. C. Aiyar Award for Telecom Education in 2004, and the Best Academic Paper Award at IEEE WCNC in 2011. He served as the Technical Program Co-Chair for IEEE Information Theory Workshop (ITW’02) held in Bengaluru, in 2002. He was an Editor of IEEE WIRELESS COMMUNICATIONS LETTERS (2012–2015) and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (2007–2011) and an Associate Editor of Coding Theory for IEEE TRANSACTIONS ON INFORMATION THEORY (2008–2011) and (2013–2015).