# Fundamental Limits of Decentralized Data Shuffling

Kai Wan, Daniela Tuninetti, Mingyue Ji, Giuseppe Caire, Pablo Piantanida

# Fundamental Limits of Decentralized Data Shuffling

Kai Wan, *Member, IEEE,* Daniela Tuninetti, *Senior Member, IEEE,* Mingyue Ji, *Member, IEEE,* Giuseppe Caire, *Fellow, IEEE,* and Pablo Piantanida, *Senior Member, IEEE*

***Abstract*—Data shuffling of training data among different computing nodes (workers) has been identified as a core element to improve the statistical performance of modern large-scale machine learning algorithms. Data shuffling is often considered as one of the most significant bottlenecks in such systems due to the heavy communication load. Under a master-worker architecture (where a master has access to the entire dataset and only communication between the master and the workers is allowed) coding has been recently proved to considerably reduce the communication load. This work considers a different communication paradigm referred to as *decentralized data shuffling*, where workers are allowed to communicate with one another via a shared link. The decentralized data shuffling problem has two phases: workers communicate with each other during the *data shuffling* phase, and then workers update their stored content during the *storage* phase. The main challenge is to derive novel converse bounds and achievable schemes for decentralized data shuffling by considering the asymmetry of the workers' storages (i.e., workers are constrained to store different files in their storages based on the problem setting), in order to characterize the fundamental limits of this problem.**

**For the case of uncoded storage (i.e., each worker directly stores a subset of bits of the dataset), this paper proposes converse and achievable bounds (based on distributed interference alignment and distributed clique-covering strategies) that are within a factor of $3/2$ of one another. The proposed schemes are also exactly optimal under the constraint of uncoded storage for either large storage size or at most four workers in the system.**

***Index Terms*—Decentralized Data shuffling, uncoded storage, distributed clique covering.**

## I. Introduction

RECENT years have witnessed the emergence of big data and machine learning with wide applications in both business and consumer worlds. To cope with such a large size/dimension of data and the complexity of machine learning algorithms, it is increasingly popular to use distributed computing platforms such as Amazon Web Services Cloud, Google Cloud, and Microsoft Azure services, where large-scale distributed machine learning algorithms can be implemented. The approach of data shuffling has been identified as one of the core elements to improve the statistical performance of modern large-scale machine learning algorithms [1], [2]. In particular, data shuffling consists of re-shuffling the training data among all computing nodes (workers) once every few iterations, according to some given learning algorithms. However, due to the huge communication cost, data shuffling may become one of the main system bottlenecks.

To tackle this communication bottleneck problem, under a master-worker setup where the master has access to the entire dataset, coded data shuffling has been recently proposed to significantly reduce the communication load between master and workers [3]. However, when the whole dataset is stored across the workers, data shuffling can be implemented in a distributed fashion by allowing direct communication between the workers[1]. In this way, the communication bottleneck between a master and the workers can be considerably alleviated. This can be advantageous if the transmission capacity among workers is much higher than that between the master and workers, and the communication load between this two setups are similar.

In this work, we consider such a *decentralized data shuffling* framework, where workers, connected by the same communication bus (common shared link), are allowed to communicate[2]. Although a master node may be present for the initial data distribution and/or for collecting the results of the training phase in a machine learning application, it is not involved in the data shuffling process which is entirely managed by the worker nodes in a distributed manner. In the following, we will review the literature of coded data shuffling (which we shall refer to as centralized data shuffling) and introduce the decentralized data shuffling framework studied in this paper.

### A. Centralized Data Shuffling

The coded data shuffling problem was originally proposed in [3] in a *master-worker centralized model* as illustrated in Fig. 1a. In this setup, a master, with the access to the whole

A short version of this paper was presented the 56th Annual Allerton Conference (2018) on Communication, Control, and Computing in Monticello, USA.

K. Wan and G. Caire are with the Electrical Engineering and Computer Science Department, Technische Universität Berlin, 10587 Berlin, Germany (e-mail: kai.wan@tu-berlin.de; caire@tu-berlin.de). The work of K. Wan and G. Caire was partially funded by the European Research Council under the ERC Advanced Grant N. 789190, CARENET.

D. Tuninetti is with the Electrical and Computer Engineering Department, University of Illinois at Chicago, Chicago, IL 60607, USA (e-mail: danielat@uic.edu). The work of D. Tuninetti was supported in parts by NSF 1527059 and NSF 1910309.

M. Ji is with the Electrical and Computer Engineering Department, University of Utah, Salt Lake City, UT 84112, USA (e-mail: mingyue.ji@utah.edu). The work of M. Ji was supported by NSF 1817154 and NSF 1824558.

P. Piantanida is with CentraleSupélec–French National Center for Scientific Research (CNRS)–Université Paris-Sud, 91192 Gif-sur-Yvette, France, and with Montreal Institute for Learning Algorithms (MILA) at Université de Montréal, QC H3T 1N8, Canada (e-mail: pablo.piantanida@centralesupelec.fr). This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 792464.

[1] In practice, workers communicate with each other as described in [1].

[2] Notice that putting all nodes on the same bus (typical terminology in Compute Science) is very common and practically relevant since this is what happens for example with Ethernet, or with the Peripheral Component Interconnect Express (PCI Express) bus inside a multi-core computer, where all cores share a common bus for intercommunication. The access of such bus is regulated by some collision avoidance protocol such as Carrier Sense Multiple Access (CSMA) [4] or Token ring [5], such that once one node talks at a time, and all other listen. Therefore, this architecture is relevant in practice.

dataset containing $\mathsf{N}$ data units, is connected to $\mathsf{K} = \mathsf{N}/\mathsf{q}$ workers, where $\mathsf{q} := \mathsf{N}/\mathsf{K}$ is a positive integer. Each shuffling epoch is divided into *data shuffling* and *storage update* phases. In the data shuffling phase, a subset of the data units is assigned to each worker and each worker must recover these data units from the broadcasted packets of the master and its own stored content from the previous epoch. In the storage update phase, each worker must store the newly assigned data units and, in addition, some information about other data units that can be retrieved from the storage content and master transmission in the current epoch. Such additional information should be strategically designed in order to help the coded delivery of the required data units in the following epochs. Each worker can store up to $\mathsf{M}$ data units in its local storage. If each worker directly copies some bits of the data units in its storage, the storage update phase is said to be *uncoded*. On the other hand, if the workers store functions (e.g., linear combinations) of the data objects' bits, the storage update is said to be *coded*. The goal is, for a given $(\mathsf{M}, \mathsf{N}, \mathsf{q})$, to find the best two-phase strategy that minimizes the communication load during the data shuffling phase regardless of the shuffle.

The scheme proposed in [3] uses a random uncoded storage (to fill users' extra memories independently when $\mathsf{M} > \mathsf{q}$) and a coded multicast transmission from the master to the workers, and yields a gain of a factor of $O(\mathsf{K})$ in terms of communication load with respect to the naive scheme for which the master simply broadcasts the missing, but required data bits to the workers.

The centralized coded data shuffling scheme with coordinated (i.e., deterministic) uncoded storage update phase was originally proposed in [6], [7], in order to minimize the worst-case communication load $\mathsf{R}$ among all the possible shuffles, i.e., $\mathsf{R}$ is smallest possible such that any shuffle can be realized. The proposed schemes in [6], [7] are optimal under the constraint of uncoded storage for the cases where there is no extra storage for each worker (i.e., $\mathsf{M} = \mathsf{q}$) or there are less than or equal to three workers in the systems. Inspired by the achievable and converse bounds for the single-bottleneck-link caching problem in [8]–[10], the authors in [11] then proposed a general coded data shuffling scheme, which was shown to be order optimal to within a factor of 2 under the constraint of uncoded storage. Also in [11], the authors improved the performance of the general coded shuffling scheme by introducing an aligned coded delivery, which was shown to be optimal under the constraint of uncoded storage for either $\mathsf{M} = \mathsf{q}$ or $\mathsf{M} \geq (\mathsf{K} - 2)\mathsf{q}$.

Recently, inspired by the improved data shuffling scheme in [11], the authors in [12] proposed a linear coding scheme based on interference alignment, which achieves the optimal worst-case communication load under the constraint of uncoded storage for all system parameters. In addition, under the constraint of uncoded storage, the proposed coded data shuffling scheme in [12] was shown to be optimal for any shuffles (not just for the worst-case shuffles) when $\mathsf{q} = 1$.

### B. Decentralized Data Shuffling

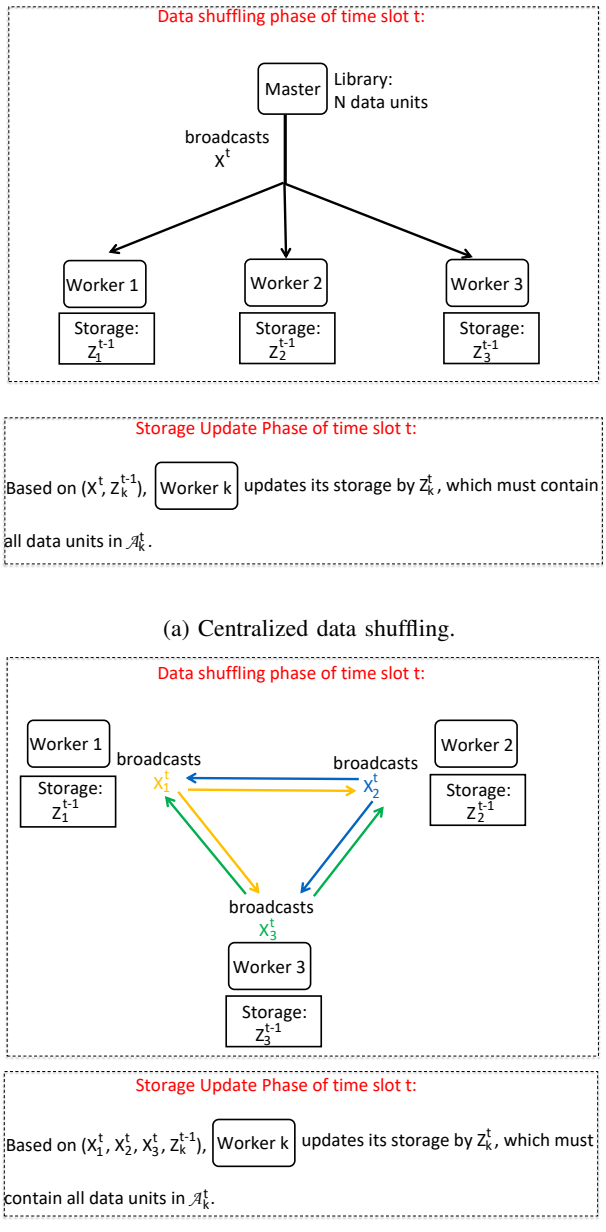
An important limitation of the centralized framework is the assumption that workers can only receive packets from the master. Since the entire dataset is stored in a decentralized fashion across the workers at each epoch of the distributed learning algorithm, the master may not be needed in the data shuffling phase if workers can communicate with each other (e.g., [1]). In addition, the communication among workers can be much more efficient compared to the communication from the master node to the workers [1], e.g., the connection between the master node and workers is via a single-ported interface, where only one message can be passed for a given time/frequency slot. In this paper, we propose the *decentralized data shuffling* problem as illustrated in Fig. 1b, where only communications among workers are allowed during the shuffling phase. This means that in the data shuffling phase, each worker broadcasts well designed coded packets (i.e., representations of the data) based on its stored content in the previous epoch. Workers take turn in transmitting, and transmissions are received error-free by all other workers through the common communication bus. The objective is to design the data shuffling and storage update phases in order to minimize the total communication load across all the workers in the worst-case shuffling scenario.

*Importance of decentralized data shuffling in practice:* In order to make the decentralized topology work in practice, we need to firstly guarantee that all the data units are already stored across the nodes so that the communication among computing nodes is sufficient. This condition is automatically satisfied from the definition of the decentralized data shuffling problem. Although the decentralized coded data shuffling incurs a larger load compared to its centralized counterpart, in practice, we may prefer the decentralized coded shuffling framework. This is due to the fact that the transmission delay/latency of the data transmission in real distributed computing system may depend on other system properties besides the total communication load, and the decentralized topology may achieve a better transmission delay/latency. This could be due to that 1) the connection between the master node and the worker clusters is normally via a single-ported interference, where only one message can be transmitted per time/frequency slot [1]; 2) computing nodes are normally connected (e.g., via grid, or ring topologies) and the link bandwidth is generally much faster, in addition, computing nodes can transmit in parallel.

### C. Relation to Device-to-device (D2D) Caching and Distributed Computing

The coded decentralized data shuffling problem considered in this paper is related to the *coded device-to-device (D2D) caching problem* [13] and the *coded distributed computing problem* [14] – see also Remark 1 next.

The coded caching problem was originally proposed in [8] for a shared-link broadcast model. The authors in [13] extended the coded caching model to the case of D2D networks under the so-called protocol model. By choosing the communication radius of the protocol model such that each node can broadcast messages to all other nodes in the network, the delivery phase of D2D coded caching is resemblant (as far as the topology of communication between the nodes is




(a) Centralized data shuffling.




(b) Decentralized data shuffling.

Fig. 1: The system models of the 3-worker centralized and decentralized data shuffling problems in time slot $t$. The data units in $\mathcal{A}_k^t$ are assigned to worker $k$, where $k \in \{1, 2, 3\}$ at time $t$.

concerned) to the shuffling phase of our decentralized data shuffling problem.

Recently, the scheme for coded D2D caching in [13] has been extended to the coded distributed computing problem [14], which consists of two stages named *Map* and *Reduce*. In the Map stage, workers compute a fraction of intermediate computation values using local input data according to the designed Map functions. In the Reduce stage, according to the designed Reduce functions, workers exchange among each other a set of well designed (coded) intermediate computation values, in order to compute the final output results. The coded distributed computing problem can be seen as a coded D2D caching problem under the constraint of uncoded and symmetric cache placement, where the symmetry means that each worker uses the same cache function for each file. A converse bound was proposed in [14] to show that the proposed coded distributed computing scheme is optimal in terms of communication load. This coded distributed computing framework was extended to cases such as computing only necessary intermediate values [15], [16], reducing file partitions and number of output functions [16], [17], and considering random network topologies [18], random connection graphs [19], [20], stragglers [21], storage cost [22], and heterogeneous computing power, function assignment and storage space [23], [24].

Compared to coded D2D caching and coded distributed computing, the decentralized data shuffling problem differs as follows. On the one hand, an *asymmetric* constraint on the stored contents for the workers is present (because each worker must store all bits of each assigned data unit in the previous epoch, which breaks the symmetry of the stored contents across data units of the other settings). On the other hand, each worker also needs to *dynamically update its storage* based on the received packets and its own stored content in the previous epoch. Therefore the decentralized data shuffling problem over multiple data assignment epochs is indeed a dynamic system where the evolution across the epochs of the node stored content plays a key role, while in the other problems reviewed above the cache content is static and determined at a single initial placement setup phase.

### *D. Relation to Centralized, Distributed, and Embedded Index Codings*

In a *distributed index coding* problem [25], [26], there are multiple senders connected to several receivers, where each sender or receiver can access to a subset of messages in the library. Each receiver demands one message and according to the users' demands and side informations, the senders cooperatively broadcast packets to all users to satisfy the users' demands. The difference in a *centralized index coding* problem [27] compared to the distributed one is that only one sender exists and this sender can access the whole library. Very recently, the authors in [28] considered a special case of distributed index coding, referred to as *embedded index coding*, where each node acts as both a sender and a receiver in the system. It was shown in [28] that a linear code for this embedded index coding problem can be obtained from a linear index code for the centralized version of the problem by doubling the communication load.

The centralized and decentralized data shuffling phases with uncoded storage are special cases of centralized and embedded index coding problems, respectively. By using the construction in [28] we could thus design a code for the decentralized data shuffling problem by using the optimal (linear) code for the centralized case [12]; this would give a decentralized data shuffling scheme with a load twice that of [12]. It will be clarified later (in Remark 2) that the proposed decentralized data shuffling schemes are strictly better than the

those derived with the construction in [28]. This is so because the construction in [28] is general, while our design is for the specific topology considered.

### E. Contributions

In this paper, we study the decentralized data shuffling problem, for which we propose converse and achievable bounds as follows.

1) *Novel converse bound under the constraint of uncoded storage.* Inspired by the induction method in [14, Thm.1] for the distributed computing problem, we derive a converse bound under the constraint of uncoded storage. Different from the converse bound for the distributed computing problem, in our proof we propose a novel approach to account for the additional constraint on the "asymmetric" stored content.
2) *Scheme A: General scheme for any* $\mathsf{M}$. By extending the general centralized data shuffling scheme from [11] to our decentralized model, we propose a general decentralized data shuffling scheme, where the analysis holds for any system parameters.
3) *Scheme B: Improved scheme for* $\mathsf{M} \geq (\mathsf{K}-2)\mathsf{q}$. It can be seen later that Scheme A does not fully leverage the workers' stored content. With the storage update phase inspired by the converse bound and also used in the improved centralized data shuffling scheme in [11], we propose a two-step scheme for decentralized data shuffling to improve on Scheme A. In the first step we generate multicast messages as in [8], and in the second step we encode these multicast messages by a linear code based on distributed interference alignment (see Remark 3).
   By comparing our proposed converse bound and Scheme B, we prove that Scheme B is exactly optimal under the constraint of uncoded storage for $\mathsf{M} \geq (\mathsf{K}-2)\mathsf{q}$. Based on this result, we can also characterize the exact optimality under the constraint of uncoded storage when the number of workers satisfies $\mathsf{K} \leq 4$.
4) *Scheme C: Improved scheme for* $\mathsf{M} = 2\mathsf{q}$. The delivery schemes proposed in [8], [11], [13] for coded caching with a shared-link, D2D caching, and centralized data shuffling, all belong to the class of *clique-covering* method from a graph theoretic viewpoint. By a non-trivial extension from a *distributed clique-covering approach* for the two-sender distributed index coding problems [29] to our decentralized data shuffling problem for the case $\mathsf{M} = 2\mathsf{q}$, we propose a novel decentralized data shuffling scheme. The resulting scheme outperforms the previous two schemes for this specific storage size.
5) *Order optimality under the constraint of uncoded storage.* By combing the three proposed schemes and comparing with the proposed converse bound, we prove the order optimality of the combined scheme within a factor of $3/2$ under the constraint of uncoded storage.

### F. Paper Organization

The rest of the paper is organized as follows. The system model and problem formulation for the decentralized data shuffling problem are given in Section II. Results from decentralized data shuffling related to our work are compiled in Section III. Our main results are summarized in Section IV. The proof of the proposed converse bound can be found in Section V, while the analysis of the proposed achievable schemes is in Section VI. Section VII concludes the paper. The proofs of some auxiliary results can be found in the Appendix.

### G. Notation Convention

We use the following notation convention. Calligraphic symbols denote sets, bold symbols denote vectors, and sans-serif symbols denote system parameters. We use $|\cdot|$ to represent the cardinality of a set or the length of a vector; $[a:b] := \{a, a+1, \ldots, b\}$ and $[n] := \{1, 2, \ldots, n\}$; $\oplus$ represents bit-wise XOR; $\mathbb{N}$ denotes the set of all positive integers.

## II. System Model

The $(\mathsf{K}, \mathsf{q}, \mathsf{M})$ decentralized data shuffling problem illustrated in Fig. 1b is defined as follows. There are $\mathsf{K} \in \mathbb{N}$ workers, each of which is charged to process and store $\mathsf{q} \in \mathbb{N}$ data units from a dataset of $\mathsf{N} := \mathsf{K}\mathsf{q}$ data units. Data units are denoted as $(F_1, F_2, \ldots, F_{\mathsf{N}})$ and each data unit is a binary vector containing $\mathsf{B}$ i.i.d. bits. Each worker has a local storage of $\mathsf{M}\mathsf{B}$ bits, where $\mathsf{q} \leq \mathsf{M} \leq \mathsf{K}\mathsf{q} = \mathsf{N}$. The workers are interconnected through a noiseless multicast network.

The computation process occurs over $\mathsf{T}$ time slots/epochs. At the end of time slot $t-1$, $t \in [\mathsf{T}]$, the content of the local storage of worker $k \in [\mathsf{K}]$ is denoted by $Z_k^{t-1}$; the content of all storages is denoted by $Z^{t-1} := (Z_1^{t-1}, Z_2^{t-1}, \ldots, Z_{\mathsf{K}}^{t-1})$. At the beginning of time slot $t \in [\mathsf{T}]$, the $\mathsf{N}$ data units are partitioned into $\mathsf{K}$ disjoint batches, each containing $\mathsf{q}$ data units. The data units indexed by $\mathcal{A}_k^t \subseteq [N]$ are assigned to worker $k \in [\mathsf{K}]$ who must store them in its local storage by the end of time slot $t \in [\mathsf{T}]$. The dataset partition (i.e., data shuffle) in time slot $t \in [\mathsf{T}]$ is denoted by $\mathcal{A}^t = (\mathcal{A}_1^t, \mathcal{A}_2^t, \ldots, \mathcal{A}_{\mathsf{K}}^t)$ and must satisfy

$$|\mathcal{A}_k^t| = \mathsf{q}, \ \forall k \in [\mathsf{K}], \tag{1a}$$

$$\mathcal{A}_{k_1}^t \cap \mathcal{A}_{k_2}^t = \emptyset, \ \forall (k_1, k_2) \in [\mathsf{K}]^2 : k_1 \neq k_2, \tag{1b}$$

$$\cup_{k \in [\mathsf{K}]} \mathcal{A}_k^t = [\mathsf{N}] \quad \text{(dataset partition)}. \tag{1c}$$

If $\mathsf{q} = 1$, we let $\mathcal{A}_k^t = \{d_k^t\}$ for each $k \in [\mathsf{K}]$.

We denote the worker who must store data unit $F_i$ at the end of time slot $t$ by $\mathsf{u}_i^t$, where

$$\mathsf{u}_i^t = k \text{ if and only if } i \in \mathcal{A}_k^t. \tag{2}$$

The following two-phase scheme allows workers to store the requested data units.

*Initialization:* We first focus on the initial time slot $t = 0$, where a master node broadcasts to all the workers. Given partition $\mathcal{A}^0$, worker $k \in [\mathsf{K}]$ must store all the data units $F_i$ where $i \in \mathcal{A}_k^0$; if there is excess storage, that is, if $\mathsf{M} > \mathsf{q}$, worker $k \in [\mathsf{K}]$ can store in its local storage parts of the data units indexed by $[\mathsf{N}] \setminus \mathcal{A}_k^0$. The storage function for worker $k \in [\mathsf{K}]$ in time slot $t = 0$ is denoted by $\psi_k^0$, where

$$Z_k^0 := \psi_k^0\Big(\mathcal{A}^0, (F_i : i \in \mathsf{N})\Big) \quad \text{(initial storage placement)}: \tag{3a}$$

$$H\Big(Z_k^0\Big) \le \mathsf{MB},\ \forall k \in [\mathsf{K}] \quad \text{(initial storage size constraint)}, \tag{3b}$$

$$H\Big(\big(F_i : i \in \mathcal{A}_k^0\big)|Z_k^0\Big) = 0 \quad \text{(initial storage content constraint)}. \tag{3c}$$

Notice that the storage initialization and the storage update phase (which will be described later) are without knowledge of later shuffles. In subsequent time slots $t \in [\mathsf{T}]$, the master is not needed and the workers communicate with one another.

*Data Shuffling Phase:* Given global knowledge of the stored content $Z^{t-1}$ at all workers, and of the data shuffle from $\mathcal{A}^{t-1}$ to $\mathcal{A}^t$ (indicated as $\mathcal{A}^{t-1} \to \mathcal{A}^t$) worker $k \in [\mathsf{K}]$ broadcasts a message $X_k^t$ to all other workers, where $X_k^t$ is based only on the its local storage content $Z_k^{t-1}$, that is,

$$H\Big(X_k^t|Z_k^{t-1}\Big) = 0 \quad \text{(encoding)}. \tag{4}$$

The collection of all sent messages is denoted by $X^t := (X_1^t, X_2^t, \ldots, X_{\mathsf{K}}^t)$. Each worker $k \in [\mathsf{K}]$ must recover all data units indexed by $\mathcal{A}_k^t$ from the sent messages $X^t$ and its local storage content $Z_k^{t-1}$, that is,

$$H\Big(\big(F_i : i \in \mathcal{A}_k^t\big)|Z_k^{t-1}, X^t\Big) = 0 \quad \text{(decoding)}. \tag{5}$$

The rate $\mathsf{K}$-tuple $(\mathsf{R}_1^{\mathcal{A}^{t-1}\to\mathcal{A}^t}, \ldots, \mathsf{R}_{\mathsf{K}}^{\mathcal{A}^{t-1}\to\mathcal{A}^t})$ is said to be *feasible* if there exist delivery functions $\phi_k^t : X_k^t = \phi_k^t(Z_k^{t-1})$ for all $t \in [\mathsf{T}]$ and $k \in [\mathsf{K}]$ satisfying the constraints (4) and (5), and such that

$$H\Big(X_k^t\Big) \le \mathsf{BR}_k^{\mathcal{A}^{t-1}\to\mathcal{A}^t} \quad \text{(load)}. \tag{6}$$

*Storage Update Phase:* After the data shuffling phase in time slot $t$, we have the storage update phase in time slot $t \in [\mathsf{T}]$. Each worker $k \in [\mathsf{K}]$ must update its local storage based on the sent messages $X^t$ and its local stored content $Z_k^{t-1}$, that is,

$$H\Big(Z_k^t|Z_k^{t-1}, X^t\Big) = 0 \quad \text{(storage update)}, \tag{7}$$

by placing in it all the recovered data units, that is,

$$H\Big(\big(F_i : i \in \mathcal{A}_k^t\big)|Z_k^t\Big) = 0, \quad \text{(stored content)}. \tag{8}$$

Moreover, the local storage has limited size bounded by

$$H\Big(Z_k^t\Big) \le \mathsf{MB},\ \forall k \in [\mathsf{K}], \quad \text{(storage size)}. \tag{9}$$

A storage update for worker $k \in [\mathsf{K}]$ is said to be feasible if there exist functions $\psi_k^t : Z_k^t = \psi_k^t(\mathcal{A}_k^t, Z_k^{t-1}, X^t)$ for all $t \in [\mathsf{T}]$ and $k \in [\mathsf{K}]$ satisfying the constraints in (7), (8) and (9).

Note: if for any $k_1, k_2 \in [\mathsf{K}]$ and $t_1, t_2 \in [\mathsf{T}]$ we have $\Psi_{k_1}^{t_1} \equiv \Psi_{k_2}^{t_2}$ (i.e., $\Psi_{k_1}^{t_1}$ is equivalent to $\Psi_{k_2}^{t_2}$), the storage phase is called *structural invariant*.

*Objective:* The objective is to minimize the *worst-case total communication load*, or just load for short in the following, among all possible consecutive data shuffles, that is we aim to characterized $\mathsf{R}^\star$ defined as

$$\mathsf{R}^\star := \lim_{\mathsf{T}\to\infty} \min_{\substack{\psi_k^{t'}, \phi_k^{t'}: \\ t' \in [\mathsf{T}], k \in [\mathsf{K}]}} \max_{(\mathcal{A}^0, \ldots, \mathcal{A}^{\mathsf{T}})} \left\{ \max_{t\in[\mathsf{T}]} \sum_{k\in[\mathsf{K}]} \mathsf{R}_k^{\mathcal{A}^{t-1}\to\mathcal{A}^t} : \text{the rate } \mathsf{K}\text{-tuple and the storage are feasible} \right\}. \tag{10}$$

The minimum load under the constraint of uncoded storage is denoted by $\mathsf{R}_{\mathrm{u}}^\star$. In general, $\mathsf{R}_{\mathrm{u}}^\star \ge \mathsf{R}^\star$, because the set of all general data shuffling schemes is a superset of all data shuffling schemes with uncoded storage.

**Remark 1** (Decentralized Data Shuffling vs D2D Caching)**.** *The D2D caching problem studied in [13] differs from our setting as follows:*

1) *in the decentralized data shuffling problem one has the constraint on the stored content in* (8) *that imposes that each worker stores the whole requested files, which is not present in the D2D caching problem; and*
2) *in the D2D caching problem each worker fills its local cache by accessing the whole library of files, while in the decentralized data shuffling problem each worker updates its local storage based on the received packets in the current time slot and its stored content in the previous time slot as in* (7)*.*

*Because of these differences, achievable and converse bounds for the decentralized data shuffling problem can not be obtained by trivial renaming of variables in the D2D caching problem.*

□

## III. Relevant Results for Centralized Data Shuffling

Data shuffling was originally proposed in [3] for the centralized scenario, where communications only exists between the master and the workers, that is, the $\mathsf{K}$ decentralized encoding conditions in (4) are replaced by $H(X^t|F_1, \ldots, F_{\mathsf{N}}) = 0$ where $X^t$ is broadcasted by the master to all the workers. We summarize next some key results from [11], which will be used in the following sections. We shall use the subscripts "u,cen,conv" and "u,cen,ach" for converse (conv) and achievable (ach) bounds, respectively, for the centralized problem (cen) with uncoded storage (u). We have

1) *Converse for centralized data shuffling:* For a $(\mathsf{K}, \mathsf{q}, \mathsf{M})$ centralized data shuffling system, the worst-case communication load under the constraint of uncoded storage is lower bounded by the lower convex envelope of the following storage-load pairs [11, Thm.2]

$$\left(\frac{\mathsf{M}}{\mathsf{q}} = m,\ \frac{\mathsf{R}}{\mathsf{q}} = \frac{\mathsf{K}-m}{m}\right)_{\text{u,cen,conv}},\ \forall m \in [\mathsf{K}]. \tag{11}$$

2) *Achievability for centralized data shuffling:* In [11] it was also shown that the lower convex envelope of the following storage-load pairs is achievable with uncoded storage [11, Thm.1]

$$\left(\frac{\mathsf{M}}{\mathsf{q}} = 1 + g\frac{\mathsf{K}-1}{\mathsf{K}},\ \frac{\mathsf{R}}{\mathsf{q}} = \frac{\mathsf{K}-g}{g+1}\right)_{\text{u,cen,ach}},\ \forall g \in [0:\mathsf{K}]. \tag{12}$$

The achievable bound in (12) was shown to be within a factor $\frac{\mathsf{K}}{\mathsf{K}-1} \le 2$ of the converse bound in (11) under the constraint of uncoded storage [11, Thm.3].

3) *Optimality for centralized data shuffling:* It was shown in [12, Thm.4] that the converse bound in (11) can be achieved by a scheme that uses linear network coding and interference alignement/elimination. An optimality result similar to [12, Thm.4] was shown in [11, Thm.4], but only for $m \in \{1, \mathsf{K}-2, \mathsf{K}-1\}$; note that $m = \mathsf{K}$ is trivial.

Although the scheme that achieves the load in (12) is not optimal in general, we shall next describe its inner workings as we will generalize it to the case of decentralized data shuffling.

*Structural Invariant Data Partitioning and Storage:* Fix $g \in [0:\mathsf{K}]$ and divide each data unit into $\binom{\mathsf{K}}{g}$ non-overlapping and equal-length sub-blocks of length $\mathsf{B}/\binom{\mathsf{K}}{g}$ bits. Let each data unit be $F_i = (G_{i,\mathcal{W}} : \mathcal{W} \subseteq [\mathsf{K}] : |\mathcal{W}| = g),\ \forall i \in [\mathsf{N}]$. The storage of worker $k \in [\mathsf{K}]$ at the end of time slot $t$ is as follows,[3]

$$Z_k^t = \Big(\underbrace{(G_{i,\mathcal{W}} : \forall \mathcal{W}, \forall i \in \mathcal{A}_k^t)}_{\text{required data units}} \cup \underbrace{(G_{i,\mathcal{W}} : k \in \mathcal{W}, \forall i \in [\mathsf{N}] \setminus \mathcal{A}_k^t)}_{\text{other data units}}\Big) \tag{13}$$

$$= \Big(\underbrace{(G_{i,\mathcal{W}} : k \notin \mathcal{W}, \forall i \in \mathcal{A}_k^t)}_{\text{variable part of the storage}} \cup \underbrace{(G_{i,\mathcal{W}} : k \in \mathcal{W}, \forall i \in [\mathsf{N}])}_{\text{fixed part of the storage}}\Big). \tag{14}$$

Worker $k \in [\mathsf{K}]$ stores all the $\binom{\mathsf{K}}{g}$ sub-blocks of the required $\mathsf{q}$ data units indexed by $\mathcal{A}_k^t$, and also $\binom{\mathsf{K}-1}{g-1}$ sub-blocks of each data unit indexed by $[\mathsf{N}] \setminus \mathcal{A}_k^t$ (see (13)), thus the required storage space is

$$\mathsf{M} = \mathsf{q} + (\mathsf{N} - \mathsf{q})\frac{\binom{\mathsf{K}-1}{g-1}}{\binom{\mathsf{K}}{g}} = \Big(1 + g\frac{\mathsf{K}-1}{\mathsf{K}}\Big)\mathsf{q}. \tag{15}$$

It can be seen (see (14) and also Table I) that the storage of worker $k \in [\mathsf{K}]$ at time $t \in [\mathsf{T}]$ is partitioned in two parts: (i) the "fixed part" contains all the sub-blocks of all data points that have the index $k$ in the second subscript; this part of the storage will not be changed over time; and (ii) the "variable part" contains all the sub-blocks of all required data points at time $t$ that do not have the index $k$ in the second subscript; this part of the storage will be updated over time.

*Initialization (for the achievable bound in* (12)*):* The master directly transmits all data units. The storage is as in (14) given $\mathcal{A}^0$.

*Data Shuffling Phase of time slot $t \in [T]$ (for the achievable bound in* (12)*):* After the end of storage update phase at time $t-1$, the new assignment $\mathcal{A}^t$ is revealed. For notation convenience, let

$$G'_{k,\mathcal{W}} = \Big(G_{i,\mathcal{W}} : i \in \mathcal{A}_k^t \setminus \mathcal{A}_k^{t-1}\Big), \tag{16}$$

for all $k \in [\mathsf{K}]$ and all $\mathcal{W} \subseteq [\mathsf{K}]$, where $|\mathcal{W}| = g$ and $k \notin \mathcal{W}$. Note that in (16) we have $|G'_{k,\mathcal{W}}| \le \mathsf{B}\frac{\mathsf{q}}{\binom{\mathsf{K}}{g}}$, with equality (i.e., worst-case scenario) if and only if $\mathcal{A}_k^t \cap \mathcal{A}_k^{t-1} = \emptyset$. To allow the workers to recover their missing sub-blocks, the central server broadcasts $X^t$ defined as

$$X^t = (W_{\mathcal{J}}^t : \mathcal{J} \subseteq [\mathsf{K}] : |\mathcal{J}| = g+1), \tag{17}$$

$$\text{where } W_{\mathcal{J}}^t = \oplus_{k \in \mathcal{J}} G'_{k,\mathcal{J}\setminus\{k\}}, \tag{18}$$

where in the multicast message $W_{\mathcal{J}}^t$ in (18) the sub-blocks $G'_{k,\mathcal{W}}$ involved in the sum are zero-padded to meet the length of the longest one. Since worker $k \in \mathcal{J}$ requests $G'_{k,\mathcal{J}\setminus\{k\}}$ and has stored all the remaining sub-blocks in $W_{\mathcal{J}}^t$ defined in (18), it can recover $G'_{k,\mathcal{J}\setminus\{k\}}$ from $W_{\mathcal{J}}^t$, and thus all its missing sub-blocks from $X^t$.

*Storage Update Phase of time slot $t \in [T]$ (for the achievable bound in* (12)*):* Worker $k \in [\mathsf{K}]$ evicts from the (variable part of its) storage the sub-blocks $(G_{i,\mathcal{W}} : k \notin \mathcal{W}, \forall i \in \mathcal{A}_k^{t-1} \setminus \mathcal{A}_k^t)$ and replaces them with the sub-blocks $(G_{i,\mathcal{W}} : k \notin \mathcal{W}, \forall i \in \mathcal{A}_k^t \setminus \mathcal{A}_k^{t-1})$. This procedure maintains the structural invariant storage structure of the storage in (14).

*Performance Analysis (for the achievable bound in* (12)*):* The total worst-case communication load satisfies

$$\mathsf{R} \le \mathsf{q}\frac{\binom{\mathsf{K}}{g+1}}{\binom{\mathsf{K}}{g}} = \mathsf{q}\frac{\mathsf{K}-g}{g+1}, \tag{19}$$

with equality (i.e., worst-case scenario) if and only if $\mathcal{A}_k^t \cap \mathcal{A}_k^{t-1} = \emptyset$ for all $k \in [\mathsf{K}]$.

## IV. Main Results

In this section, we summarize our main results for the decentralized data shuffling problem. We shall use the subscripts "u,dec,conv" and "u,dec,ach" for converse (conv) and achievable (ach) bounds, respectively, for the decentralized problem (dec) with uncoded storage (u). We have:

1) *Converse:* We start with a converse bound for the decentralized data shuffling problem under the constraint of uncoded storage.

**Theorem 1** (Converse)**.** *For a $(\mathsf{K}, \mathsf{q}, \mathsf{M})$ decentralized data shuffling system, the worst-case load under the constraint of uncoded storage is lower bounded by the lower convex envelope of the following storage-load pairs*

$$\Big(\frac{\mathsf{M}}{\mathsf{q}} = m, \frac{\mathsf{R}}{\mathsf{q}} = \frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}\Big)_{\text{u,dec,conv}},\ \forall m \in [\mathsf{K}]. \tag{20}$$

Notice that the proposed converse bound is a piecewise linear curve with the corner points in (20) and these corner points are successively convex.

The proof of Theorem 1 can be found in Section V and is inspired by the induction method proposed in [14, Thm.1] for the distributed computing problem. However, there are two main differences in our proof compared to [14, Thm.1]: (i) we need to account for the additional constraint on the stored content in (8), (ii) our storage update phase is by problem definition in (8) asymmetric across data units, while it is symmetric in the distributed computing problem.

[3] Notice that here each sub-block $G_{i,\mathcal{W}}$ is stored by workers $\{\mathsf{u}_i^t\} \cup \mathcal{W}$. In addition, later in our proofs of the converse bound and proposed achievable schemes for decentralized data shuffling, the notation $F_{i,\mathcal{W}}$ denotes the sub-block of $F_i$, which is stored by workers in $\mathcal{W}$.

TABLE I: Example of file partitioning and storage in (14) at the end of time slot $t$ for the decentralized data shuffling problem with $(\mathsf{K},\mathsf{q},\mathsf{M})=(3,1,7/3)$ and $\mathcal{A}^t=(3,1,2)$ where $g=2$.

| Workers | Sub-blocks of $F_1$ | Sub-blocks of $F_2$ | Sub-blocks of $F_3$ |
|---|---|---|---|
| Worker 1 stores | $G_{1,\{1,2\}}$, $G_{1,\{1,3\}}$ | $G_{2,\{1,2\}}$, $G_{2,\{1,3\}}$ | $G_{3,\{1,2\}}$, $G_{3,\{1,3\}}$, $G_{3,\{2,3\}}$ |
| Worker 2 stores | $G_{1,\{1,2\}}$, $G_{1,\{1,3\}}$, $G_{1,\{2,3\}}$ | $G_{2,\{1,2\}}$, $G_{2,\{2,3\}}$ | $G_{3,\{1,2\}}$, $G_{3,\{2,3\}}$ |
| Worker 3 stores | $G_{1,\{1,3\}}$, $G_{1,\{2,3\}}$ | $G_{2,\{1,2\}}$, $G_{2,\{1,3\}}$, $G_{2,\{2,3\}}$ | $G_{3,\{1,3\}}$, $G_{3,\{2,3\}}$ |

2) *Achievability:* We next extend the centralized data shuffling scheme in Section III to our decentralized setting.

**Theorem 2** (Scheme A)**.** *For a* $(\mathsf{K},\mathsf{q},\mathsf{M})$ *decentralized data shuffling system, the worst-case load under the constraint of uncoded storage is upper bounded by the lower convex envelope of the following storage-load pairs*

$$\left(\frac{\mathsf{M}}{\mathsf{q}}=1+g\frac{\mathsf{K}-1}{\mathsf{K}},\frac{\mathsf{R}}{\mathsf{q}}=\frac{\mathsf{K}-g}{g}\right)_{\text{u,dec,ach}},\ \forall g\in[\mathsf{K}-1]. \tag{21}$$

$$\textit{and (smallest storage)}\ \left(\frac{\mathsf{M}}{\mathsf{q}}=1,\frac{\mathsf{R}}{\mathsf{q}}=\mathsf{K}\right)_{\text{u,dec,ach}}, \tag{22}$$

$$\textit{and (largest storage)}\ \left(\frac{\mathsf{M}}{\mathsf{q}}=\mathsf{K},\frac{\mathsf{R}}{\mathsf{q}}=0\right)_{\text{u,dec,ach}}. \tag{23}$$

The proof is given in Section VI-A.

A limitation of Scheme A in Theorem 2 is that, in time slot $t\in[\mathsf{T}]$ worker $k\in[\mathsf{K}]$ does not fully leverage all its stored content. We overcome this limitation by developing Scheme B described in Section VI-B.

**Theorem 3** (Scheme B)**.** *For a* $(\mathsf{K},\mathsf{q},\mathsf{M})$ *decentralized data shuffling system, the worst-case load under the constraint of uncoded storage for* $\mathsf{M}\geq(\mathsf{K}-2)\mathsf{q}$ *is upper bounded by the lower convex envelope of the following storage-load pairs*

$$\left(\frac{\mathsf{M}}{\mathsf{q}}=m,\frac{\mathsf{R}}{\mathsf{q}}=\frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}\right)_{\text{u,dec,ach}},\quad \forall m\in\{\mathsf{K}-2,\mathsf{K}-1,\mathsf{K}\}. \tag{24}$$

We note that Scheme B is neither a direct extension of [11, Thm.4] nor of [12, Thm.4] from the centralized to the decentralized setting. As it will become clear from the details in Section VI-B, our scheme works with a rather simple way to generate the multicast messages transmitted by the workers, and it applies to any shuffle, not just to the worst-case one. In Remark 4, we also extend this scheme for the general storage size regime.

Scheme B in Theorem 3 uses a distributed clique-covering method to generate multicast messages similar to what is done for D2D caching [8], where distributed clique cover is for the side information graph (more details in Section V-A). Each multicast message corresponds to one distributed clique and includes one linear combination of all nodes in this clique. However, due to the asymmetry of the decentralized data shuffling problem (not present in D2D coded caching), the lengths of most distributed cliques are small and thus the multicast messages based on cliques and sent by a worker in general include only a small number of messages (i.e., small multicast gain). To overcome this limitation, the key idea of Scheme C for $\mathsf{M}/\mathsf{q}=2$ (described in Section VI-C) is to augment some of the cliques and send them in $\mathsf{M}/\mathsf{q}=2$ linear combinations.

**Theorem 4** (Scheme C)**.** *For a* $(\mathsf{K},\mathsf{q},\mathsf{M})$ *decentralized data shuffling system, the worst-case load under the constraint of uncoded storage for* $\mathsf{M}/\mathsf{q}=2$ *is upper bounded by*

$$\left(\frac{\mathsf{M}}{\mathsf{q}}=2,\frac{\mathsf{R}}{\mathsf{q}}=\frac{2\mathsf{K}(\mathsf{K}-2)}{3(\mathsf{K}-1)}\right)_{\text{u,dec,ach}}. \tag{25}$$

It will be seen later that the proposed schemes only use binary codes, and only XOR operations are needed for the decoding procedure.

Finally, we combine the proposed three schemes (by considering the one among Schemes A, B or C that attains the lowest load for each storage size).

**Corollary 1** (Combined Scheme)**.** *For a* $(\mathsf{K},\mathsf{q},\mathsf{M})$ *decentralized data shuffling system, the achieved storage-load tradeoff of the combined scheme is the lower convex envelope of the corner points is as follows:*

- $\mathsf{M}=\mathsf{q}$. *With Scheme A, the worst-case load is* $\mathsf{q}\frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}$.
- $\mathsf{M}=2\mathsf{q}$. *With Scheme C, the worst-case load is* $\mathsf{q}\frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}\frac{4}{3}$.
- $\mathsf{M}=\big(1+g\frac{\mathsf{K}-1}{\mathsf{K}}\big)\mathsf{q}$ *where* $g\in[2:\mathsf{K}-3]$. *With Scheme A, the worst-case load is* $\mathsf{q}\frac{\mathsf{K}-g}{g}$.
- $\mathsf{M}=m\mathsf{q}$ *where* $m\in[\mathsf{K}-2:\mathsf{K}]$. *With Scheme B, the worst-case load is* $\mathsf{q}\frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}$.

3) *Optimality:* By comparing our achievable and converse bounds, we have the following exact optimality results.

**Theorem 5** (Exact Optimality for $\mathsf{M}/\mathsf{q}\geq\mathsf{K}-2$)**.** *For a* $(\mathsf{K},\mathsf{q},\mathsf{M})$ *decentralized data shuffling system, the optimal worst-case load under the constraint of uncoded storage for* $\mathsf{M}/\mathsf{q}\in[\mathsf{K}-2,\mathsf{K}]$ *is given in Theorem 1 and is attained by Scheme B in Theorem 3.*

Note that the converse bound on the load for the case $\mathsf{M}/\mathsf{q}=1$ is trivially achieved by Scheme A in Theorem 2.

From Theorem 5 we can immediately conclude the following.

**Corollary 2** (Exact Optimality for $\mathsf{K}\leq 4$)**.** *For a* $(\mathsf{K},\mathsf{q},\mathsf{M})$ *decentralized data shuffling system, the optimal worst-case load under the constraint of uncoded storage is given by Theorem 1 for* $\mathsf{K}\leq 4$.

Finally, by combining the three proposed achievable schemes, we have the following order optimality result proved in Section VI-D.

**Theorem 6** (Order Optimality for $\mathsf{K} > 4$)**.** *For a $(\mathsf{K}, \mathsf{q}, \mathsf{M})$ decentralized data shuffling system under the constraint of uncoded storage, for the cases not covered by Theorem 5, the combined scheme in Corollary 1 achieves the converse bound in Theorem 1 within a factor of $3/2$. More precisely, when $m\mathsf{q} \leq \mathsf{M} \leq (m+1)\mathsf{q}$, the multiplicative gap between the achievable load in Corollary 1 and the converse bound in Theorem 1 is upper bounded by*

- $4/3$, *if* $m = 1$;
- $1 - \frac{1}{\mathsf{K}} + \frac{1}{2}$, *if* $m = 2$;
- $1 - \frac{1}{\mathsf{K}} + \frac{1}{m-1}$, *if* $m \in [3 : \mathsf{K} - 3]$;
- $1$, *if* $m \in \{\mathsf{K} - 2, \mathsf{K} - 1\}$.

4) Finally, by directly comparing the minimum load for the centralized data shuffling system (the master-worker framework) in (12) with the load achieved by the combined scheme in Corollary 1, we can quantify the communication cost of peer-to-peer operations (i.e., the multiplicative gap on the minimum worst-case load under the constraint of uncoded storage between decentralized and centralized data shufflings), which will be proved in Section VI-D.

**Corollary 3.** *For a $(\mathsf{K}, \mathsf{q}, \mathsf{M})$ decentralized data shuffling system under the constraint of uncoded storage, the communication cost of peer-to-peer operations is no more than a factor of 2. More precisely, when $\mathsf{K} \leq 4$, this cost is $\frac{\mathsf{K}}{\mathsf{K}-1}$; when $\mathsf{K} \geq 5$ and $m\mathsf{q} \leq \mathsf{M} \leq (m+1)\mathsf{q}$, this cost is upper bounded by*

- $\frac{4\mathsf{K}}{3(\mathsf{K}-1)}$, *if* $m = 1$;
- $1 + \frac{\mathsf{K}}{2(\mathsf{K}-1)}$, *if* $m = 2$;
- $1 + \frac{\mathsf{K}}{(m-1)(\mathsf{K}-1)}$, *if* $m \in [3 : \mathsf{K} - 3]$;
- $\frac{\mathsf{K}}{\mathsf{K}-1}$, *if* $m \in \{\mathsf{K} - 2, \mathsf{K} - 1\}$.

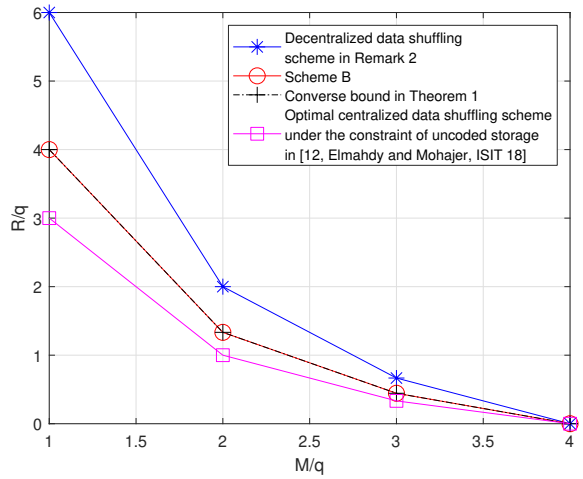
**Remark 2** (Comparison to the direct extension from [28])**.** *As mentioned in Section I-D, the result in [28] guarantees that from the optimal (linear) centralized data shuffling scheme in [12] one can derive a linear scheme for the decentralized setting with twice the number of transmissions (by the construction given in [28, Proof of Theorem 4]), that is, the following storage-load corner points can be achieved,*

$$\left(\frac{\mathsf{M}}{\mathsf{q}} = m,\ \frac{\mathsf{R}}{\mathsf{q}} = 2\frac{\mathsf{K} - m}{m}\right),\ \forall m \in [\mathsf{K}]. \tag{26}$$

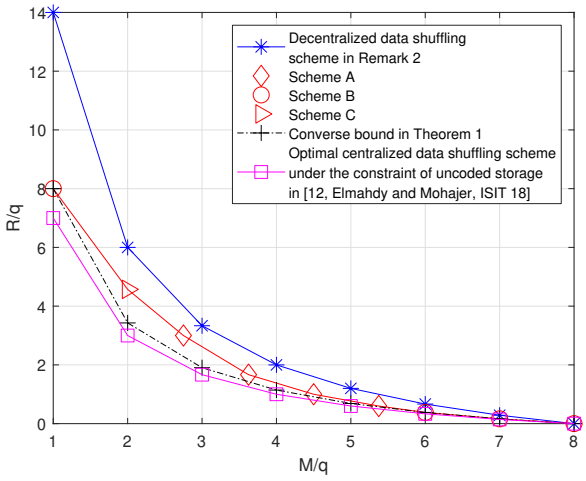
*The multiplicative gap between the data shuffling scheme in* (26) *and the proposed converse bound in Theorem 1, is $\frac{2(\mathsf{K}-1)}{\mathsf{K}}$, which is close to 2 when $\mathsf{K}$ is large. Our proposed combined scheme in Corollary 1 does better: for $\mathsf{K} \leq 4$, it exactly matches the proposed converse bound in Theorem 1, while for $\mathsf{K} > 4$ it is order optimal to within a factor of $3/2$.*

*In addition, the multiplicative gap $\frac{2(\mathsf{K}-1)}{\mathsf{K}}$ is independent of the storage size $\mathsf{M}$. It is shown in Theorem 6 that the multiplicative gap between the combined scheme and the converse decreases towards 1 when $\mathsf{M}$ increases.*

*Similar observation can be obtained for the communication cost of peer-to-peer operations. With the data shuffling scheme in* (26)*, we can only prove this cost upper is bounded by 2, which is independent of $\mathsf{M}$ and $\mathsf{K}$. With the combined scheme, it is shown in Corollary 3 that this cost decreases towards to 1 when $\mathsf{M}$ and $\mathsf{K}$ increase.*




(a) $\mathsf{K} = 4$.




(b) $\mathsf{K} = 8$.

Fig. 2: The storage-load tradeoff for the decentralized data shuffling problem.

We conclude this section by providing some numerical results. Fig. 2 plots our converse bound and the best convex combination of the proposed achievable bounds on the worst-case load under the constraint of uncoded storage for the decentralized data shuffling systems with $\mathsf{K} = 4$ (Fig. 2a) and $\mathsf{K} = 8$ (Fig. 2b) workers. For comparison, we also plot the achieved load by the decentralized data shuffling scheme in Remark 2, and the optimal load for the corresponding centralized system in (11) under the constraint of uncoded storage. For the case of $\mathsf{K} = 4$ workers, Theorem 1 is tight under the constraint of uncoded storage. For the case of $\mathsf{K} = 8$ workers, Scheme B meets our converse bound when $\mathsf{M}/\mathsf{q} \in [6, 8]$, and also trivially when $\mathsf{M}/\mathsf{q} = 1$.

## V. Proof of Theorem 1: Converse Bound under the Constraint of Uncoded Storage

We want to lower bound $\max_{\mathcal{A}^t} \sum_{k\in[\mathsf{K}]} \mathsf{R}_k^{\mathcal{A}^{t-1}\to\mathcal{A}^t}$ for a fixed $t \in [\mathsf{T}]$ and a fixed $\mathcal{A}^{t-1}$. It can be also checked that this lower bound is also a lower bound on the worst-case total communication load in (10) among all $t \in [\mathsf{T}]$ and all possible $(\mathcal{A}^0, \ldots, \mathcal{A}^{\mathsf{T}})$. Recall that the excess storage is said to be *uncoded* if each worker simply copies bits from the data units in its local storage. When the storage update phase is uncoded, we can divide each data unit into sub-blocks depending on the set of workers who store them.

### A. Sub-block Division of the Data Shuffling Phase under Uncoded Storage

Because of the data shuffling constraint in (1), all the bits of all data units are stored by at least one worker at the end of any time slot. Recall that the worker who must store data unit $F_i$ at the end of time slot $t$ is denoted by $\mathsf{u}_i^t$ in (2). In the case of excess storage, some bits of some files may be stored by multiple workers. We denote by $F_{i,\mathcal{W}}$ the sub-block of bits of data unit $F_i$ exclusively stored by workers in $\mathcal{W}$ where $i \in [\mathsf{N}]$ and $\mathcal{W} \subseteq [\mathsf{K}]$. By definition, at the end of step $t-1$, we have that $\mathsf{u}_i^{t-1}$ must be in $\mathcal{W}$ for all sub-blocks $F_{i,\mathcal{W}}$ of data unit $F_i$; we also let $F_{i,\mathcal{W}} = \emptyset$ for all $\mathcal{W} \subseteq [\mathsf{K}]$ if $\mathsf{u}_i^{t-1} \notin \mathcal{W}$. Hence, at the end of step $t-1$, each data unit $F_i$ can be written as

$$F_i = \{F_{i,\mathcal{W}} : \mathcal{W} \subseteq [\mathsf{K}], \mathsf{u}_i^{t-1} \in \mathcal{W}\}, \tag{27}$$

and the storage content as

$$\begin{aligned} Z_k^{t-1} &= \{F_{i,\mathcal{W}} : \mathcal{W} \subseteq [\mathsf{K}], \{\mathsf{u}_i^{t-1}, k\} \subseteq \mathcal{W}, i \in [\mathsf{N}]\} \\ &= \underbrace{\{F_i : i \in \mathcal{A}_k^{t-1}\}}_{\text{required data units}} \cup \underbrace{\{F_{i,\mathcal{W}} : i \notin \mathcal{A}_k^{t-1}, \{\mathsf{u}_i^{t-1}, k\} \subseteq \mathcal{W}\}}_{\text{other data units}}. \end{aligned} \tag{28}$$

We note that the sub-blocks $F_{i,\mathcal{W}}$ have different content at different times (as the partition in (27) is a function of $\mathcal{A}^{t-1}$ through $(\mathsf{u}_1^{t-1}, \ldots, \mathsf{u}_{\mathsf{N}}^{t-1})$); however, in order not to clutter the notation, we will not explicitly denote the dependance of $F_{i,\mathcal{W}}$ on time. Finally, please note that the definition of sub-block $F_{i,\mathcal{W}}$, as defined here for the converse bound, is not the same as $G_{i,\mathcal{W}}$ defined in Section VI for the achievable scheme (see Footnote 3).

### B. Proof of Theorem 1

We are interested in deriving an information theoretic lower bound on the worst-case communication load. We will first obtain a number of lower bounds on the load for some carefully chosen shuffles. Since the load of any shuffle is at most as large as the worst-case shuffle, the obtained lower bounds are valid lower bounds for the worst-case load as well. We will then average the obtained lower bounds.

In particular, the shuffles are chosen as follows. Consider a permutation of $[\mathsf{K}]$ denoted by $\mathbf{d} = (d_1, \ldots, d_{\mathsf{K}})$ where $d_k \neq k$ for each $k \in [\mathsf{K}]$ and consider the shuffle

$$\mathcal{A}_k^t = \mathcal{A}_{d_k}^{t-1}, \ \forall k \in [\mathsf{K}]. \tag{29}$$

We define $X_{\mathcal{S}}^t$ as the messages sent by the workers in $\mathcal{S}$ during time slot $t$, that is,

$$X_{\mathcal{S}}^t := \Big\{X_k^t : k \in \mathcal{S}\Big\}. \tag{30}$$

From Lemma 1 in the Appendix with $\mathcal{S} = [\mathsf{K}]$, which is the key novel contribution of our proof and that was inspired by the induction argument in [14], we have

$$\begin{aligned} \frac{\mathsf{R}_{\mathsf{u}}^\star}{\mathsf{B}} &\geq H\Big(X_{[\mathsf{K}]}^t\Big) \\ &\geq \sum_{m=1}^{\mathsf{K}} \sum_{k\in[\mathsf{K}]} \sum_{i\in\mathcal{A}_k^t} \sum_{\mathcal{W}\subseteq[\mathsf{K}]\setminus\{k\}:\ \mathsf{u}_i^{t-1}\in\mathcal{W},\ |\mathcal{W}|=m} \frac{|F_{i,\mathcal{W}}|}{m}. \end{aligned} \tag{31}$$

To briefly illustrate the main ingredients on the derivation of (31), we provide the following example.

**Example 1** ($\mathsf{K} = \mathsf{N} = 3$)**.** We focus on the decentralized data shuffling problem with $\mathsf{K} = \mathsf{N} = 3$. Without loss of generality, we assume

$$\mathcal{A}_1^{t-1} = \{3\},\ \mathcal{A}_2^{t-1} = \{1\},\ \mathcal{A}_3^{t-1} = \{2\}. \tag{32}$$

Based on $(\mathcal{A}_1^{t-1}, \mathcal{A}_2^{t-1}, \mathcal{A}_3^{t-1})$, we can divide each data unit into sub-blocks as in (27). More precisely, we have

$$\begin{aligned} F_1 &= \{F_{1,\{2\}}, F_{1,\{1,2\}}, F_{1,\{2,3\}}, F_{1,\{1,2,3\}}\}; \\ F_2 &= \{F_{2,\{3\}}, F_{2,\{1,3\}}, F_{2,\{2,3\}}, F_{2,\{1,2,3\}}\}; \\ F_3 &= \{F_{3,\{1\}}, F_{3,\{1,2\}}, F_{3,\{1,3\}}, F_{3,\{1,2,3\}}\}. \end{aligned}$$

At the end of time slot $t-1$, each worker $k \in [3]$ stores $F_{i,\mathcal{W}}$ if $k \in \mathcal{W}$. Hence, we have

$$\begin{aligned} Z_1^{t-1} =& \{F_{1,\{1,2\}}, F_{1,\{1,2,3\}}, F_{2,\{1,3\}}, F_{2,\{1,2,3\}}, \\ & F_{3,\{1\}}, F_{3,\{1,2\}}, F_{3,\{1,3\}}, F_{3,\{1,2,3\}}\}; \\ Z_2^{t-1} =& \{F_{1,\{2\}}, F_{1,\{1,2\}}, F_{1,\{2,3\}}, F_{1,\{1,2,3\}}, F_{2,\{2,3\}}, \\ & F_{2,\{1,2,3\}}, F_{3,\{1,2\}}, F_{3,\{1,2,3\}}\}; \\ Z_3^{t-1} =& \{F_{1,\{1,3\}}, F_{1,\{1,2,3\}}, F_{2,\{3\}}, F_{2,\{1,3\}}, F_{2,\{2,3\}}, \\ & F_{2,\{1,2,3\}}, F_{3,\{1,3\}}, F_{3,\{1,2,3\}}\}. \end{aligned}$$

Now we consider a permutation of $[3]$ denoted by $\mathbf{d} = (d_1, d_2, d_3)$ where $d_k \neq k$ for each $k \in [3]$ and assume the considered permutation is $(2, 3, 1)$. Based on $\mathbf{d}$, from (29), we consider the shuffle

$$\mathcal{A}_1^t = \mathcal{A}_2^{t-1} = \{1\},\ \mathcal{A}_2^t = \mathcal{A}_3^{t-1} = \{2\},\ \mathcal{A}_3^t = \mathcal{A}_1^{t-1} = \{3\}. \tag{33}$$

We first prove

$$H(X_{[2]}^t | Z_3^{t-1}, F_3) \geq |F_{1,\{2\}}|. \tag{34}$$

More precisely, by the decoding constraint in (5), we have

$$H(F_{1,\{2\}} | Z_1^{t-1}, X_{[3]}^t) = 0, \tag{35}$$

which implies

$$H(F_{1,\{2\}} | Z_1^{t-1}, X_{[3]}^t, Z_3^{t-1}, F_3) = 0. \tag{36}$$

Since $F_{1,\{2\}}$ is not stored by workers 1 and 3, we have

$$|F_{1,\{2\}}| \leq H(F_{1,\{2\}}, X_{[2]}^t | Z_1^{t-1}, X_3^t, Z_3^{t-1}, F_3) \tag{37a}$$

$$= H(X_{[2]}^t|Z_1^{t-1}, X_3^t, Z_3^{t-1}, F_3)+$$
$$H(F_{1,\{2\}}|Z_1^{t-1}, X_{[3]}^t, Z_3^{t-1}, F_3) \quad (37b)$$
$$= H(X_{[2]}^t|Z_1^{t-1}, X_3^t, Z_3^{t-1}, F_3) \quad (37c)$$
$$\leq H(X_{[2]}^t|X_3^t, Z_3^{t-1}, F_3) \quad (37d)$$
$$= H(X_{[2]}^t|Z_3^{t-1}, F_3), \quad (37e)$$

where (37c) comes from (36) and (37e) comes from the fact that $X_3^t$ is a function of $Z_3^{t-1}$. Hence, we prove (34).

Similarly, we can also prove

$$H(X_{\{1,3\}}^t|Z_2^{t-1}, F_2) \geq |F_{3,\{1\}}|. \quad (38)$$
$$H(X_{\{2,3\}}^t|Z_1^{t-1}, F_1) \geq |F_{2,\{3\}}|. \quad (39)$$

In addition, we have

$$H(X_{[3]}^t) = \frac{1}{3}\sum_{k\in[3]}\Big(H(X_k^t) + H(X_{[3]\setminus\{k\}}^t|X_k^t)\Big) \quad (40a)$$
$$\geq \frac{1}{3}\left(H(X_{[3]}^t) + \sum_{k\in[3]} H(X_{[3]\setminus\{k\}}^t|X_k^t)\right), \quad (40b)$$

and thus

$$2H(X_{[3]}^t) \geq \sum_{k\in[3]} H(X_{[3]\setminus\{k\}}^t|X_k^t) \quad (40c)$$
$$\geq \sum_{k\in[3]} H(X_{[3]\setminus\{k\}}^t|Z_k^{t-1}) \quad (40d)$$
$$= \sum_{k\in[3]} H(X_{[3]\setminus\{k\}}^t, F_k|Z_k^{t-1}) \quad (40e)$$
$$= \sum_{k\in[3]} H(F_k|Z_k^{t-1}) + H(X_{[3]\setminus\{k\}}^t|F_k, Z_k^{t-1}), \quad (40f)$$

where (40d) comes from the fact that $X_k^t$ is a function of $Z_k^{t-1}$ and conditioning cannot increase entropy, and (40e) comes from the decoding constraint for worker $k$ in (5).

Let us focus on worker 1 and we have

$$H(F_1|Z_1^{t-1}) = |F_{1,\{2\}}| + |F_{1,\{2,3\}}|. \quad (41)$$

In addition, we have $H(X_{\{2,3\}}^t|Z_1^{t-1}, F_1) \geq |F_{2,\{3\}}|$ from (39). Hence,

$$H(F_1|Z_1^{t-1}) + H(X_{\{2,3\}}^t|Z_1^{t-1}, F_1)$$
$$\geq |F_{1,\{2\}}| + |F_{1,\{2,3\}}| + |F_{2,\{3\}}|. \quad (42)$$

Similarly, we have

$$H(F_2|Z_2^{t-1}) + H(X_{\{1,3\}}^t|Z_2^{t-1}, F_2)$$
$$\geq |F_{2,\{3\}}| + |F_{2,\{1,3\}}| + |F_{3,\{1\}}|; \quad (43)$$
$$H(F_3|Z_3^{t-1}) + H(X_{\{1,2\}}^t|Z_3^{t-1}, F_3)$$
$$\geq |F_{3,\{1\}}| + |F_{3,\{1,2\}}| + |F_{1,\{2\}}|. \quad (44)$$

By taking (42)-(44) to (40f), we have

$$H(X_{[3]}^t) \geq \frac{1}{2}\sum_{k\in[3]} H(F_k|Z_k^{t-1}) + H(X_{[3]\setminus\{k\}}^t|F_k, Z_k^{t-1}) \quad (45a)$$
$$= |F_{1,\{2\}}| + |F_{2,\{3\}}| + |F_{3,\{1\}}| + \frac{|F_{1,\{2,3\}}|}{2} + \frac{|F_{2,\{1,3\}}|}{2}$$
$$+ \frac{|F_{3,\{1,2\}}|}{2}, \quad (45b)$$

coinciding with (31). □

We now go back to the general proof of Theorem 1. We next consider all the permutations $\mathbf{d} = (d_1, \ldots, d_{\mathsf{K}})$ of $[\mathsf{K}]$ where $d_k \neq k$ for each $k \in [\mathsf{K}]$, and sum together the inequalities in the form of (31). For an integer $m \in [\mathsf{K}]$, by the symmetry of the problem, the sub-blocks $F_{i,\mathcal{W}}$ where $i \in [\mathsf{N}]$, $\mathsf{u}_i^{t-1} \in \mathcal{W}$ and $|\mathcal{W}| = m$ appear the same number of times in the final sum. In addition, the total number of these sub-blocks in general is $\mathsf{N}\binom{\mathsf{K}-1}{m-1}$ and the total number of such sub-blocks in each inequality in the form of (31) is $\mathsf{N}\binom{\mathsf{K}-2}{m-1}$. So we obtain

$$\mathsf{R}_{\mathrm{u}}^\star \geq \sum_{m=1}^{\mathsf{K}} \sum_{i\in[\mathsf{N}]} \sum_{\mathcal{W}\subseteq[\mathsf{K}]:\ \mathsf{u}_i^{t-1}\in\mathcal{W},\ |\mathcal{W}|=m} \frac{\binom{\mathsf{K}-2}{m-1}}{m\binom{\mathsf{K}-1}{m-1}} |F_{i,\mathcal{W}}| \frac{\mathsf{qK}}{\mathsf{NB}} \quad (46)$$
$$= \sum_{m=1}^{\mathsf{K}} \mathsf{qK}\ x_m \frac{1-(m-1)/(\mathsf{K}-1)}{m} \quad (47)$$
$$= \sum_{m=1}^{\mathsf{K}} \mathsf{q}\ x_m \frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}, \quad (48)$$

where we defined $x_m$ as the total number of bits in the sub-blocks stored by $m$ workers at the end of time slot $t-1$ normalized by the total number of bits $\mathsf{NB}$, i.e.,

$$0 \leq x_m := \sum_{i\in[\mathsf{N}]} \sum_{\mathcal{W}\subseteq[\mathsf{K}]:\ \mathsf{u}_i^{t-1}\in\mathcal{W},\ |\mathcal{W}|=m} \frac{|F_{i,\mathcal{W}}|}{\mathsf{NB}}, \quad (49)$$

which must satisfy

$$\sum_{m\in[\mathsf{K}]} x_m = 1 \text{ (total size of all data units)}, \quad (50)$$
$$\sum_{m\in[\mathsf{K}]} m\, x_m \leq \frac{\mathsf{KM}}{\mathsf{N}} = \frac{\mathsf{M}}{\mathsf{q}} \text{ (total storage size)}. \quad (51)$$

We then use a method based on Fourier-Motzkin elimination [30, Appendix D] for to bound $\mathsf{R}_{\mathrm{u}}^\star$ from (47) under the constraints in (50) and (51), as we did in [9] for coded caching with uncoded cache placement. In particular, for each integer $p \in [\mathsf{K}]$, we multiply (50) by $\frac{-\mathsf{N}(2\mathsf{K}p-p^2+\mathsf{K}-p)}{p(p+1)}$ to obtain

$$\frac{-\mathsf{N}(2\mathsf{K}p-p^2+\mathsf{K}-p)}{p(p+1)} = \sum_{m=1}^{\mathsf{K}} \frac{-\mathsf{N}(2\mathsf{K}p-p^2+\mathsf{K}-p)}{p(p+1)} x_m, \quad (52)$$

and we multiply (51) by $\frac{\mathsf{NK}}{(\mathsf{K}-1)p(p+1)}$ to have

$$\frac{\mathsf{NK}}{(\mathsf{K}-1)p(p+1)}\frac{\mathsf{KM}}{\mathsf{N}} \geq \sum_{m=1}^{\mathsf{K}} \frac{\mathsf{NK}}{(\mathsf{K}-1)p(p+1)} m x_m. \quad (53)$$

We then add (52), (53), and (47) to obtain,

$$\mathsf{R}_{\mathrm{u}}^\star \geq \sum_{m=1}^{\mathsf{K}} \frac{\mathsf{NK}(p-m)(p+1-m)}{m(\mathsf{K}-1)p(p+1)} x_m - \frac{\mathsf{NK}}{(\mathsf{K}-1)p(p+1)}\frac{\mathsf{KM}}{\mathsf{N}}$$
$$+ \frac{\mathsf{N}(2\mathsf{K}p-p^2+\mathsf{K}-p)}{p(p+1)} \quad (54)$$

$$\geq -\frac{\mathsf{NK}}{(\mathsf{K}-1)p(p+1)}\frac{\mathsf{KM}}{\mathsf{N}}+\frac{\mathsf{N}(2\mathsf{K}p-p^2+\mathsf{K}-p)}{p(p+1)}. \tag{55}$$

Hence, for each integer $p \in [\mathsf{K}]$, the bound in (55) becomes a linear function in $\mathsf{M}$. When $\mathsf{M}=\mathsf{q}p$, from (55) we have $\mathsf{R}_{\mathrm{u}}^{\star} \geq \frac{\mathsf{N}(\mathsf{K}-p)}{(\mathsf{K}-1)p}$. When $\mathsf{M}=\mathsf{q}(p+1)$, from (55) we have $\mathsf{R}_{\mathrm{u}}^{\star} \geq \frac{\mathsf{N}(\mathsf{K}-p-1)}{(\mathsf{K}-1)(p+1)}$. In conclusion, we prove that $\mathsf{R}_{\mathrm{u}}^{\star}$ is lower bounded by the lower convex envelope (also referred to as "memeory sharing") of the points $\left(\mathsf{M}=\mathsf{q}m, \mathsf{R}=\frac{\mathsf{N}(\mathsf{K}-m)}{(\mathsf{K}-1)m}\right)$, where $m \in [\mathsf{K}]$.

This concludes the proof of Theorem 1.

### C. Discussion

We conclude this session the following remarks:

1) The corner points from the converse bound are of the form $\left(\mathsf{M}/\mathsf{q}=m, \mathsf{R}/\mathsf{q}=\frac{\mathsf{K}\binom{\mathsf{K}-2}{m-1}}{m\binom{\mathsf{K}-1}{m-1}}\right)$, which may suggest the following placement.
   At the end of time slot $t-1$, each data unit is partitioned into $\binom{\mathsf{K}-1}{m-1}$ equal-length sub-blocks of length $\mathsf{B}/\binom{\mathsf{K}-1}{m-1}$ bits as $F_i=(F_{i,\mathcal{W}}: \mathcal{W}\subseteq[\mathsf{K}], |\mathcal{W}|=m, \mathrm{u}_i^{t-1}\in\mathcal{W})$; by definition $F_{i,\mathcal{W}}=\emptyset$ if either $\mathrm{u}_i^{t-1}\notin\mathcal{W}$ or $|\mathcal{W}|\neq m$. Each worker $k\in[\mathsf{K}]$ stores all the sub-blocks $F_{i,\mathcal{W}}$ if $k\in\mathcal{W}$; in other words, worker $k\in[\mathsf{K}]$ stores all the $\binom{\mathsf{K}-1}{m-1}$ sub-blocks of the desired data units, and $\binom{\mathsf{K}-2}{m-2}$ sub-blocks of the remaining data units.
   In the data shuffling phase of time slot $t$, worker $k\in[\mathsf{K}]$ must decode the missing $\binom{\mathsf{K}-1}{m-1}-\binom{\mathsf{K}-2}{m-2}=\binom{\mathsf{K}-2}{m-1}$ sub-blocks of data unit $F_j$ for all $j\in\mathcal{A}_k^t\backslash\mathcal{A}_k^{t-1}$. An interpretation of the converse bound is that, in the worst case, the total number of transmissions is equivalent to at least $\frac{\mathsf{qK}}{m}\binom{\mathsf{K}-2}{m-1}$ sub-blocks.
   We will use this interpretation to design the storage update phase our proposed Schemes B and C.
2) The converse bound is derived for the objective of minimizing the "sum load" $\sum_{k\in[\mathsf{K}]}\mathsf{R}_k^{\mathcal{A}^{t-1}\to\mathcal{A}^t}$, see (10). The same derivation would give a converse bound for the "largest individual load" $\max_{k\in[\mathsf{K}]}\mathsf{R}_k^{\mathcal{A}^{t-1}\to\mathcal{A}^t}$. In the latter case, the corner points from converse bound are of the form $\left(\mathsf{M}/\mathsf{q}=m, \mathsf{R}/\mathsf{q}=\frac{\binom{\mathsf{K}-2}{m-1}}{m\binom{\mathsf{K}-1}{m-1}}\right)$. This view point may suggest that, in the worst case, all the individual loads $\mathsf{R}_k^{\mathcal{A}^{t-1}\to\mathcal{A}^t}$ are the same, i.e., the burden of communicating missing data units is equally shared by all the workers.
   Our proof technique for Theorem 1 could also be directly extended to derive a converse bound on the *average load* (as opposed to the worst-case load) for all the possible shuffles in the decentralized data shuffling problem when $\mathsf{N}=\mathsf{K}$.

## VI. Achievable Schemes for Decentralized Data Shuffling

In this section, we propose three schemes for the decentralized data shuffling problem, and analyze their performances.

### A. Scheme A in Theorem 2

Scheme A extends the general centralized data shuffling scheme in Section III to the distributed model. Scheme A achieves the load in Theorem 2 for each storage size $\mathsf{M}=\left(1+g\frac{\mathsf{K}-1}{\mathsf{K}}\right)\mathsf{q}$, where $g\in[\mathsf{K}-1]$; the whole storage-load tradeoff piecewise curve is achieved by memory-sharing[4] between these points (given in (21)) and the (trivially achievable) points in (22)-(23).

*Structural Invariant Data Partitioning and Storage:* This is the same as the one in Section III for the centralized case.

*Initialization:* The master directly transmits all data units. The storage is as in (14) given $\mathcal{A}^0$.

*Data Shuffling Phase of time slot $t\in[T]$:* The data shuffling phase is inspired by the delivery in D2D caching [13]. Recall the definition of sub-block $G'_{k,\mathcal{W}}$ in (16), where each sub-block is known by $|\mathcal{W}|=g$ workers and needed by worker $k$. Partition $G'_{k,\mathcal{W}}$ into $g$ non-overlapping and equal-length pieces $G'_{k,\mathcal{W}}=\{G'_{k,\mathcal{W}}(j): j\in\mathcal{W}\}$. Worker $j\in\mathcal{J}$ broadcasts

$$W_{j,\mathcal{J}}^t=\underset{k\in\mathcal{J}\backslash\{j\}}{\oplus}G'_{k,\mathcal{J}\backslash\{k\}}(j),\quad \forall\mathcal{J}\subseteq[\mathsf{K}] \text{ where } |\mathcal{J}|=g+1, \tag{56}$$

in other words, one linear combination $W_{\mathcal{J}}^t$ in (17) for the centralized setting becomes $g+1$ linear combinations $W_{j,\mathcal{J}}^t$ in (56) for the decentralized setting, but of size reduced by a factor $g$. Evidently, each sub-block in $W_{j,\mathcal{J}}^t$ is stored in the storage of worker $j$ at the end of time slot $t-1$. In addition, each worker $k\in\mathcal{J}\backslash\{j\}$ knows $G'_{k_1,\mathcal{J}\backslash\{k_1\}}(j)$ where $k_1\in\mathcal{J}\backslash\{k,j\}$ such that it can recover its desired block $G'_{k,\mathcal{J}\backslash\{k\}}(j)$.

Since $|G'_{k,\mathcal{W}}|\leq\mathsf{qB}/\binom{\mathsf{K}}{g}$, the worst-case load is

$$\mathsf{R}\leq\mathsf{q}\frac{(g+1)\binom{\mathsf{K}}{g+1}}{g\binom{\mathsf{K}}{g}}=\mathsf{q}\frac{\mathsf{K}-g}{g}=:\mathsf{R}_{\text{Ach.A}}, \tag{57}$$

as claimed in Theorem 2, where the subscript "Ach.A" in (57) denotes the worst-case load achieved by Scheme A.

*Storage Update Phase of time slot $t\in[T]$:* The storage update phase is the same as the general centralized data shuffling scheme in Section III, and thus is not repeated here.

### B. Scheme B in Theorem 3

During the data shuffling phase in time slot $t$ of Scheme A, we treat some sub-blocks known by $g+1$ workers as if they were only known by $g$ workers (for example, if $\mathrm{u}_i^{t-1}\notin\mathcal{W}$, $G_{i,\mathcal{W}}$ is stored by workers $\{\mathrm{u}_i^{t-1}\}\cup\mathcal{W}$, but Scheme A treats $G_{i,\mathcal{W}}$ as if it was only stored by workers in $\mathcal{W}$), which may be suboptimal as more multicasting opportunities may be leveraged. In the following, we propose Scheme B to remedy for this shortcoming for $\mathsf{M}=m\mathsf{q}$ for $m\in\{\mathsf{K}-2,\mathsf{K}-1\}$.

[4] Memory-sharing is an achievability technique originally proposed by Maddah-Ali and Niesen in [8] for coded caching systems, which is used to extend achievability in between discrete memory points. More precisely, focus one storage size $\mathsf{M}'=\alpha\mathsf{M}_1+(1-\alpha)\mathsf{M}_2$, where $\alpha\in[0,1]$, $\mathsf{M}_1=\left(1+g\frac{\mathsf{K}-1}{\mathsf{K}}\right)\mathsf{q}$, $\mathsf{M}_2=\left(1+(g+1)\frac{\mathsf{K}-1}{\mathsf{K}}\right)\mathsf{q}$, and $g\in[\mathsf{K}-1]$. We can divide each data unit into two non-overlapping parts,with $\alpha\mathsf{B}$ and $(1-\alpha)\mathsf{B}$ bits, respectively. The first and second parts of the $\mathsf{N}$ data units are stored and transmitted based on the proposed data shuffling scheme for $\mathsf{M}_1$ and $\mathsf{M}_2$, respectively.

*Structural Invariant Data Partitioning and Storage:* Data units are partitions as inspired by the converse bound (see discussion in Section V-C), which is as in the improved centralized data shuffling scheme in [11]. Fix $m \in [\mathsf{K}]$. Partition each data unit into $\binom{\mathsf{K}-1}{m-1}$ non-overlapping equal-length sub-block of length $\mathsf{B}/\binom{\mathsf{K}-1}{m-1}$ bits. Write $F_i = (F_{i,\mathcal{W}} : \mathcal{W} \subseteq [\mathsf{K}], |\mathcal{W}| = m, \mathsf{u}_i^t \in \mathcal{W})$, and set $F_{i,\mathcal{W}} = \emptyset$ if either $\mathsf{u}_i^t \notin \mathcal{W}$ or $|\mathcal{W}| \neq m$. The storage of worker $k \in [\mathsf{K}]$ at the end of time slot $t$ is as follows,

$$Z_k^t = \Big( \underbrace{(F_{i,\mathcal{W}} : i \in \mathcal{A}_k^t, \forall \mathcal{W})}_{\text{required data units}} \cup \underbrace{(F_{i,\mathcal{W}} : i \notin \mathcal{A}_k^t, k \in \mathcal{W})}_{\text{other data units}} \Big), \tag{58}$$

that is, worker $k \in [\mathsf{K}]$ stores all the $\binom{\mathsf{K}-1}{m-1}$ sub-blocks of data unit $F_i$ if $i \in \mathcal{A}_k^t$, and $\binom{\mathsf{K}-2}{m-2}$ sub-blocks of data unit $F_j$ if $j \notin \mathcal{A}_k^t$ (the sub-blocks stored are such that $k \in \mathcal{W}$), thus the required storage space is

$$\mathsf{M} = \mathsf{q} + (\mathsf{N} - \mathsf{q}) \frac{\binom{\mathsf{K}-2}{m-2}}{\binom{\mathsf{K}-1}{m-1}} = \mathsf{q} + (m-1)\frac{\mathsf{N}-\mathsf{q}}{\mathsf{K}-1} = m\mathsf{q}. \tag{59}$$

In the following, we shall see that it is possible to maintain the storage structure in (58) after the shuffling phase.

*Initialization:* The master directly transmits all data units. The storage is as in (58) given $\mathcal{A}^0$.

*Data Shuffling Phase of time slot $t \in [T]$ for $m = \mathsf{K} - 1$:* Each data unit has been partitioned into $\binom{\mathsf{K}-1}{m-1} = \mathsf{K} - 1$ sub-blocks and each sub-block is stored by $m = \mathsf{K} - 1$ workers. Similarly to Scheme A, define the set of sub-blocks needed by worker $k \in [\mathsf{K}]$ at time slot $t$ and not previously stored as

$$F'_{k,[\mathsf{K}]\setminus\{k\}} = \Big(F_{i,\mathcal{W}} : i \in \mathcal{A}_k^t \setminus \mathcal{A}_k^{t-1}, \mathcal{W} = [\mathsf{K}] \setminus \{k\}\Big), \forall k \in [\mathsf{K}]. \tag{60}$$

Since $F'_{k,[\mathsf{K}]\setminus\{k\}}$ (of length $\mathsf{q}\mathsf{B}/(\mathsf{K}-1)$ bits in the worst case) is desired by worker $k$ and known by all the remaining $m = \mathsf{K} - 1$ workers, we partition $F'_{k,[\mathsf{K}]\setminus\{k\}}$ into $m = \mathsf{K} - 1$ pieces (of length $\mathsf{q}\mathsf{B}/(\mathsf{K}-1)^2$ bits in the worst case), and write $F'_{k,[\mathsf{K}]\setminus\{k\}} = \big(F'_{k,[\mathsf{K}]\setminus\{k\}}(j) : j \in [\mathsf{K}] \setminus \{k\}\big)$. Worker $j \in [\mathsf{K}]$ broadcasts the single linear combination (of length $\mathsf{q}\mathsf{B}/(\mathsf{K}-1)^2$ bits in the worst case) given by

$$W_j^t = \underset{k \neq j}{\oplus} F'_{k,[\mathsf{K}]\setminus\{k\}}(j). \tag{61}$$

Therefore, the worst-case satisfies

$$\mathsf{R} \leq \frac{\mathsf{K}}{(\mathsf{K}-1)^2}\mathsf{q} = \frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}\mathsf{q}\Big|_{m=\mathsf{K}-1} =: \mathsf{R}_{\text{Ach.B}}|_{\mathsf{M}=(\mathsf{K}-1)\mathsf{q}} \tag{62}$$

which coincides with the converse bound.

*Storage Upadte Phase of time slot $t \in [T]$ for $m = \mathsf{K} - 1$:* In time slot $t - 1 > 0$, we assume that the above storage configuration of each worker $k \in [\mathsf{K}]$ can be done with $Z_k^{t-2}$ and $X_j^{t-1}$ where $j \in [\mathsf{K}] \setminus \{k\}$. We will show next that at the end of time slot $t$, we can re-create the same configuration of storage, but with permuted data units. Thus by the induction method, we prove the above storage update phase is also structural invariant.

For each worker $k \in [\mathsf{K}]$ and each data unit $F_i$ where $i \in \mathcal{A}_k^t \setminus \mathcal{A}_k^{t-1}$, worker $k$ stores the whole data unit $F_i$ in its storage. For each data unit $F_i$ where $i \in \mathcal{A}_k^{t-1} \setminus \mathcal{A}_k^t$, instead of storing the whole data unit $F_i$, worker $k$ only stores the bits of $F_i$ which was stored at the end of time slot $t-1$ by worker $\mathsf{u}_i^t$. For other data units, worker $k$ does not change the stored bits. Hence, after the storage phase in time slot $t$, we can re-create the same configuration of storage as the end of time slot $t-1$ but with permuted data units.

*Data Shuffling Phase of time slot $t \in [T]$ for $m = \mathsf{K} - 2$:* We partition the $\mathsf{N}$ data units into $\mathsf{q}$ groups as $[\mathsf{N}] = \underset{i \in [\mathsf{q}]}{\cup} \mathcal{H}_i$, where each group contains $\mathsf{K}$ data units, and such that for each group $\mathcal{H}_i, i \in [\mathsf{q}]$, and each worker $k \in [\mathsf{K}]$ we have $|\mathcal{H}_i \cap \mathcal{A}_k^t| = 1$ and $|\mathcal{H}_i \cap \mathcal{A}_k^{t-1}| = 1$. In other words, the partition is such that, during the data shuffling phase of time slot $t$, among all the $\mathsf{K}$ data units in each group, each worker requests exactly one data unit and knows exactly one data unit. Such a partition can always be found [12, Lemma 7]. The dependance of $\mathcal{H}_i$ on $t$ is not specified so not to clutter the notation. For group $\mathcal{H}_i, i \in [\mathsf{q}]$, we define

$$\mathcal{U}(\mathcal{H}_i) := \{k \in [\mathsf{K}] : \mathcal{H}_i \cap \mathcal{A}_k^t \subseteq \mathcal{A}_k^{t-1}\}, \quad \forall i \in [\mathsf{q}], \tag{63}$$

as the set of workers in the group who already have stored the needed data point (i.e., who do not need to shuffle). Since each worker has to recover at most one data unit in each group, the delivery in each group is as if $\mathsf{q} = 1$. Hence, to simplify the description, we focus on the case $\mathsf{q} = 1$, in which case there is only one group and thus we simplify the notation $\mathcal{U}(\mathcal{H}_i)$ to just $\mathcal{U}$. We first use the following example to illustrate the main idea.

**Example 2.** Consider the $(\mathsf{K}, \mathsf{q}, \mathsf{M}) = (5, 1, 3)$ decentralized data shuffling problem, where $m = \mathsf{M}/\mathsf{q} = 3$. Let $\mathcal{A}^{t-1} = (5, 1, 2, 3, 4)$. During the storage update phase in time slot $t-1$, we partition each data unit into 6 equal-length sub-blocks, each of which has $\mathsf{B}/6$ bits, as

$$\begin{aligned}
F_1 =& (F_{1,\{1,2,3\}}, F_{1,\{1,2,4\}}, F_{1,\{1,2,5\}}, F_{1,\{2,3,4\}}, F_{1,\{2,3,5\}}, \\
& F_{1,\{2,4,5\}}\}, && \text{(64a)} \\
F_2 =& (F_{2,\{1,2,3\}}, F_{2,\{1,3,4\}}, F_{2,\{1,3,5\}}, F_{2,\{2,3,4\}}, F_{2,\{2,3,5\}}, \\
& F_{2,\{3,4,5\}}\}, && \text{(64b)} \\
F_3 =& (F_{3,\{1,2,4\}}, F_{3,\{1,3,4\}}, F_{3,\{1,4,5\}}, F_{3,\{2,3,4\}}, F_{3,\{2,4,5\}}, \\
& F_{3,\{3,4,5\}}\}, && \text{(64c)} \\
F_4 =& (F_{4,\{1,2,5\}}, F_{4,\{1,3,5\}}, F_{4,\{1,4,5\}}, F_{4,\{2,3,5\}}, F_{4,\{2,4,5\}}, \\
& F_{4,\{3,4,5\}}\}, && \text{(64d)} \\
F_5 =& (F_{5,\{1,2,3\}}, F_{5,\{1,2,4\}}, F_{5,\{1,2,5\}}, F_{5,\{1,3,4\}}, F_{5,\{1,3,5\}}, \\
& F_{5,\{1,4,5\}}\}, && \text{(64e)}
\end{aligned}$$

and each worker $k$ stores $F_{i,\mathcal{W}}$ if $k \in \mathcal{W}$.

In the following, we consider various shuffles in time slot $t$. If one sub-block is stored by some worker in $\mathcal{U}$, we let this worker transmit it and the transmission is equivalent to centralized data shuffling; otherwise, we will introduce the proposed scheme B to transmit it.

We first consider $\mathcal{A}^t = (1,2,3,4,5)$. For each set $\mathcal{J} \subseteq [\mathsf{K}]$ of size $|\mathcal{J}| = m+1 = \mathsf{K}-1 = 4$, we generate

$$V^t_{\mathcal{J}} = \underset{k\in\mathcal{J}}{\oplus} F_{d^t_k,\mathcal{J}\setminus\{k\}}, \tag{65}$$

where $d^t_k$ represents the demanded data unit of worker $k$ in time slot $t$ if $\mathsf{q}=1$. The details are illustrated in Table II. For example, when $\mathcal{J}=\{1,2,3,4\}$, we have $V^t_{\{1,2,3,4\}} = F_{1,\{2,3,4\}} + F_{2,\{1,3,4\}} + F_{3,\{1,2,4\}} + F_{4,\{1,2,3\}}$ where $F_{4,\{1,2,3\}}$ is empty and we replace $F_{4,\{1,2,3\}}$ by $\mathsf{B}/6$ zero bits. Since $F_{1,\{2,3,4\}}$, $F_{2,\{1,3,4\}}$, and $F_{3,\{1,2,4\}}$ are all stored by worker 4, $V^t_{\{1,2,3,4\}}$ can be transmitted by worker 4. Similarly, we let worker 3 transmit $V^t_{\{1,2,3,5\}}$, worker 2 transmit $V^t_{\{1,2,4,5\}}$, worker 1 transmit $V^t_{\{1,3,4,5\}}$, and worker 5 transmit $V^t_{\{2,3,4,5\}}$. It can be checked that each worker can recover its desired sub-blocks and the achieved load is $5/6$, which coincides with the proposed converse bound.

Let us then focus on $\mathcal{A}^t = (5,2,3,4,1)$. For this shuffle, $\mathcal{A}^{t-1}_1 = \mathcal{A}^t_1$ such that worker 1 needs not to decode anything from what the other workers transmit. We divide all desired sub-blocks into two sets, stored and not stored by worker 1 as follows

$$\begin{aligned}\mathcal{S}_{\{1\}} &= \{F_{1,\{1,2,3\}}, F_{1,\{1,2,4\}}, F_{2,\{1,3,4\}}, F_{2,\{1,3,5\}}, F_{3,\{1,2,4\}},\\ &\quad F_{3,\{1,4,5\}}, F_{4,\{1,2,5\}}, F_{4,\{1,3,5\}}\},\\ \mathcal{S}_{\emptyset} &= \{F_{1,\{2,3,4\}}, F_{2,\{3,4,5\}}, F_{3,\{2,4,5\}}, F_{4,\{2,3,5\}}\}.\end{aligned}$$

Since the sub-blocks in $\mathcal{S}_{\{1\}}$ are all stored by worker 1, we can treat worker 1 as a central server and the transmission of $\mathcal{S}_{\{1\}}$ is equivalent to centralized data shuffling with $\mathsf{K}_{\text{eq}}=4$, $\mathsf{M}_{\text{eq}}=2$ and $\mathsf{q}_{\text{eq}}=1$, where $\mathcal{U}_{\text{eq}}=\emptyset$. For this centralized problem, the data shuffling schemes in [11], [12] are optimal under the constraint of uncoded storage. Alternatively, we can also use the following simplified scheme. By generating $V^t_{\{1,2,3,4\}}$ as in (65), and it can be seen that $V^t_{\{1,2,3,4\}}$ is known by workers 1 and 4. Similarly, $V^t_{\{1,2,3,5\}}$ is known by workers 1 and 3, $V^t_{\{1,2,4,5\}}$ is known by workers 1 and 2, and $V^t_{\{1,3,4,5\}}$ is known by workers 1 and 5. Hence, we can let worker 1 transmit $V^t_{\{1,2,3,4\}} \oplus V^t_{\{1,2,3,5\}}$, $V^t_{\{1,2,3,4\}} \oplus V^t_{\{1,2,4,5\}}$, and $V^t_{\{1,2,3,4\}} \oplus V^t_{\{1,3,4,5\}}$. Hence, each worker can recover $V^t_{\{1,2,3,4\}}$, $V^t_{\{1,2,3,5\}}$, $V^t_{\{1,2,4,5\}}$, and $V^t_{\{1,3,4,5\}}$. We then consider the transmission for $\mathcal{S}_{\emptyset} = \{F_{1,\{2,3,4\}}, F_{2,\{3,4,5\}}, F_{3,\{2,4,5\}}, F_{4,\{2,3,5\}}\}$, which is equivalent to decentralized data shuffling with $\mathsf{K}_{\text{eq}}=4$, $\mathsf{M}_{\text{eq}}=3$ and $\mathsf{q}_{\text{eq}}=1$, where $\mathcal{U}_{\text{eq}}=\emptyset$ defined in (63). Hence, we can use the proposed Scheme B for $m=\mathsf{K}-1$. More precisely, we split each sub-block in $V^t_{\{2,3,4,5\}}$ into 3 non-overlapping and equal-length sub-pieces, e.g., $F_{2,\{3,4,5\}} = \{F_{2,\{3,4,5\}}(3), F_{2,\{3,4,5\}}(4), F_{2,\{3,4,5\}}(5)\}$. We then let

worker 2 transmit $F_{3,\{2,4,5\}}(2) \oplus F_{4,\{2,3,5\}}(2) \oplus F_{1,\{2,3,4\}}(2)$;
worker 3 transmit $F_{2,\{3,4,5\}}(3) \oplus F_{4,\{2,3,5\}}(3) \oplus F_{1,\{2,3,4\}}(3)$;
worker 4 transmit $F_{2,\{3,4,5\}}(4) \oplus F_{3,\{2,4,5\}}(4) \oplus F_{1,\{2,3,4\}}(4)$;
worker 5 transmit $F_{2,\{3,4,5\}}(5) \oplus F_{3,\{2,4,5\}}(5) \oplus F_{4,\{2,3,5\}}(5)$.

In conclusion, the total load for $\mathcal{A}^t=(5,2,3,4,1)$ is $\frac{3}{6}+\frac{2}{9} = \frac{13}{18}$.

TABLE II: Multicast messages for Example 2. Empty sub-blocks are colored in magenta.

| For $\mathcal{A}^t = (1,2,3,4,5) = (F_1, F_2, F_3, F_4, F_5)$ |
|---|
| Worker 1 wants $(F_{1,\{2,3,4\}}, F_{1,\{2,3,5\}}, F_{1,\{2,4,5\}}, F_{1,\{3,4,5\}} = \emptyset)$ |
| Worker 2 wants $(F_{2,\{1,3,4\}}, F_{2,\{1,3,5\}}, F_{2,\{3,4,5\}}, F_{2,\{1,4,5\}} = \emptyset)$ |
| Worker 3 wants $(F_{3,\{1,2,4\}}, F_{3,\{1,4,5\}}, F_{3,\{2,4,5\}}, F_{3,\{1,2,5\}} = \emptyset)$ |
| Worker 4 wants $(F_{4,\{1,2,5\}}, F_{4,\{1,3,5\}}, F_{4,\{2,3,5\}}, F_{4,\{1,2,3\}} = \emptyset)$ |
| Worker 5 wants $(F_{5,\{1,2,3\}}, F_{5,\{1,2,4\}}, F_{5,\{1,3,4\}}, F_{5,\{2,3,4\}} = \emptyset)$ |
| $V^t_{\{1,2,3,4\}} = F_{1,\{2,3,4\}} + F_{2,\{1,3,4\}} + F_{3,\{1,2,4\}} + F_{4,\{1,2,3\}}$ |
| $V^t_{\{1,2,3,5\}} = F_{1,\{2,3,5\}} + F_{2,\{1,3,5\}} + F_{3,\{1,2,5\}} + F_{5,\{1,2,3\}}$ |
| $V^t_{\{1,2,4,5\}} = F_{1,\{2,4,5\}} + F_{2,\{1,4,5\}} + F_{4,\{1,2,5\}} + F_{5,\{1,2,4\}}$ |
| $V^t_{\{1,3,4,5\}} = F_{1,\{3,4,5\}} + F_{3,\{1,4,5\}} + F_{4,\{1,3,5\}} + F_{5,\{1,3,4\}}$ |
| $V^t_{\{2,3,4,5\}} = F_{2,\{3,4,5\}} + F_{3,\{2,4,5\}} + F_{4,\{2,3,5\}} + F_{5,\{2,3,4\}}$ |
| For $\mathcal{A}^t = (5,2,3,4,1) = (F_5, F_2, F_3, F_4, F_1)$ |
| Worker 1 wants $(F_{5,\{2,3,4\}}, F_{5,\{2,3,5\}}, F_{5,\{2,4,5\}}, F_{5,\{3,4,5\}}) = \emptyset$ |
| Worker 2 wants $(F_{2,\{1,3,4\}}, F_{2,\{1,3,5\}}, F_{2,\{3,4,5\}}, F_{2,\{1,4,5\}} = \emptyset)$ |
| Worker 3 wants $(F_{3,\{1,2,4\}}, F_{3,\{1,4,5\}}, F_{3,\{2,4,5\}}, F_{3,\{1,2,5\}} = \emptyset)$ |
| Worker 4 wants $(F_{4,\{1,2,5\}}, F_{4,\{1,3,5\}}, F_{4,\{2,3,5\}}, F_{4,\{1,2,3\}} = \emptyset)$ |
| Worker 5 wants $(F_{1,\{1,2,3\}}, F_{1,\{1,2,4\}}, F_{1,\{2,3,4\}}, F_{1,\{1,3,4\}} = \emptyset)$ |
| $V^t_{\{1,2,3,4\}} = F_{5,\{2,3,4\}} + F_{2,\{1,3,4\}} + F_{3,\{1,2,4\}} + F_{4,\{1,2,3\}}$ |
| $V^t_{\{1,2,3,5\}} = F_{5,\{2,3,5\}} + F_{2,\{1,3,5\}} + F_{3,\{1,2,5\}} + F_{1,\{1,2,3\}}$ |
| $V^t_{\{1,2,4,5\}} = F_{5,\{2,4,5\}} + F_{2,\{1,4,5\}} + F_{4,\{1,2,5\}} + F_{1,\{1,2,4\}}$ |
| $V^t_{\{1,3,4,5\}} = F_{5,\{3,4,5\}} + F_{3,\{1,4,5\}} + F_{4,\{1,3,5\}} + F_{1,\{1,3,4\}}$ |
| $V^t_{\{2,3,4,5\}} = F_{2,\{3,4,5\}} + F_{3,\{2,4,5\}} + F_{4,\{2,3,5\}} + F_{1,\{2,3,4\}}$ |
| For $\mathcal{A}^t = (5,1,3,4,2) = (F_5, F_1, F_3, F_4, F_2)$ |
| Worker 1 wants $(F_{5,\{2,3,4\}}, F_{5,\{2,3,5\}}, F_{5,\{2,4,5\}}, F_{5,\{3,4,5\}}) = \emptyset$ |
| Worker 2 wants $(F_{1,\{1,3,4\}}, F_{1,\{1,3,5\}}, F_{1,\{3,4,5\}}, F_{1,\{1,4,5\}}) = \emptyset$ |
| Worker 3 wants $(F_{3,\{1,2,4\}}, F_{3,\{1,4,5\}}, F_{3,\{2,4,5\}}, F_{3,\{1,2,5\}} = \emptyset)$ |
| Worker 4 wants $(F_{4,\{1,2,5\}}, F_{4,\{1,3,5\}}, F_{4,\{2,3,5\}}, F_{4,\{1,2,3\}} = \emptyset)$ |
| Worker 5 wants $(F_{2,\{1,2,3\}}, F_{2,\{1,3,4\}}, F_{2,\{2,3,4\}}, F_{2,\{1,2,4\}} = \emptyset)$ |
| $V^t_{\{1,2,3,4\}} = F_{5,\{2,3,4\}} + F_{1,\{1,3,4\}} + F_{3,\{1,2,4\}} + F_{4,\{1,2,3\}}$ |
| $V^t_{\{1,2,3,5\}} = F_{5,\{2,3,5\}} + F_{1,\{1,3,5\}} + F_{3,\{1,2,5\}} + F_{2,\{1,2,3\}}$ |
| $V^t_{\{1,2,4,5\}} = F_{5,\{2,4,5\}} + F_{1,\{1,4,5\}} + F_{4,\{1,2,5\}} + F_{2,\{1,2,4\}}$ |
| $V^t_{\{1,3,4,5\}} = F_{5,\{3,4,5\}} + F_{3,\{1,4,5\}} + F_{4,\{1,3,5\}} + F_{2,\{1,3,4\}}$ |
| $V^t_{\{2,3,4,5\}} = F_{1,\{3,4,5\}} + F_{3,\{2,4,5\}} + F_{4,\{2,3,5\}} + F_{2,\{2,3,4\}}$ |

Finally, we consider $\mathcal{A}^t = \{5,1,3,4,2\}$. For this shuffle, $\mathcal{A}^{t-1}_1 = \mathcal{A}^t_1$ and $\mathcal{A}^{t-1}_2 = \mathcal{A}^t_2$ such that workers 1 and 2 need not to decode anything from other workers. We divide all desired sub-blocks into three sets

$$\mathcal{S}_{\{1,2\}} = \{F_{2,\{1,2,3\}}, F_{3,\{1,2,4\}}, F_{4,\{1,2,5\}}\},$$

stored by workers 1 and 2,

$$\mathcal{S}_{\{1\}} = \{F_{2,\{1,3,4\}}, F_{3,\{1,4,5\}}, F_{4,\{1,3,5\}}\},$$

stored by worker 1 and not by worker 2, and

$$\mathcal{S}_{\{2\}} = \{F_{2,\{2,3,4\}}, F_{3,\{2,4,5\}}, F_{4,\{2,3,5\}}\}$$

stored by worker 2 and not by worker 1.

The transmission for $\mathcal{S}_{\{1,2\}}$ is equivalent to a centralized data shuffling with $\mathsf{K}_{\text{eq}}=3$, $\mathsf{M}_{\text{eq}}=1$ and $\mathsf{q}_{\text{eq}}=1$. We use the following simplified scheme to let worker 1 transmit $V^t_{\{1,2,3,4\}} \oplus V^t_{\{1,2,3,5\}}$ and $V^t_{\{1,2,3,4\}} \oplus V^t_{\{1,2,4,5\}}$ such that each worker can recover $V^t_{\{1,2,3,4\}}$, $V^t_{\{1,2,3,5\}}$, and $V^t_{\{1,2,4,5\}}$ (as illustrated in Table II, $V^t_{\{1,2,3,4\}} = F_{3,\{1,2,4\}}$, $V^t_{\{1,2,3,5\}} = F_{2,\{1,2,3\}}$, and $V^t_{\{1,2,4,5\}} = F_{4,\{1,2,5\}}$). Similarly, for $\mathcal{S}_{\{1\}}$, we let worker 1 transmit $V^t_{\{1,3,4,5\}}$. For $\mathcal{S}_{\{2\}}$, we let worker 2 transmit $V^t_{\{2,3,4,5\}}$. In conclusion the total load for $\mathcal{A}^t = \{5,1,3,4,2\}$ is $\frac{2}{6}+\frac{1}{6}+\frac{1}{6} = \frac{2}{3}$. □

Now we are ready to introduce Scheme B for $m=\mathsf{K}-2$ as a generalization of Example 2. Recall that, from our earlier discussion, we can consider without loss of generality $\mathsf{q}=1$, and that $\mathcal{U}$ represents the set of workers who need not to

recover anything from others. We divide all desired sub-blocks by all workers into non-overlapping sets

$$\mathcal{S}_{\mathcal{K}}:=\{F_{d_k,\mathcal{W}}:k\in[\mathsf{K}]\setminus\mathcal{U},|\mathcal{W}|=m+1,\mathcal{W}\cap\mathcal{U}=\mathcal{K},k\notin\mathcal{W}\},\tag{66}$$

where $\mathcal{K}\subseteq\mathcal{U}$. We then encode the sub-blocks in each set in $\mathcal{S}_{\mathcal{K}}$ in (66) as follows:

- For each $\mathcal{K}\subseteq\mathcal{U}$ where $\mathcal{K}\neq\emptyset$, the transmission for $\mathcal{S}_{\mathcal{K}}$ is equivalent to a centralized data shuffling problem with $\mathsf{K}_{\text{eq}}=\mathsf{K}-|\mathcal{U}|$, $\mathsf{q}_{\text{eq}}=1$ and $\mathsf{M}_{\text{eq}}=m-|\mathcal{K}|$, where $\mathcal{U}_{\text{eq}}=\emptyset$. It can be seen that $\mathsf{K}_{\text{eq}}-\mathsf{M}_{\text{eq}}\leq 2$. Hence, we can use the optimal centralized data shuffling schemes in [11], [12].
  Alternatively, we propose the following simplified scheme. For each set $\mathcal{J}\subseteq[\mathsf{K}]$ of size $|\mathcal{J}|=m+1=\mathsf{K}-1$, where $\mathcal{J}\cap\mathcal{U}=\mathcal{K}$, we generate $V^t_{\mathcal{J}}$ as in (65). Each sub-block in $\mathcal{S}_{\mathcal{K}}$ appears in one $V^t_{\mathcal{J}}$, where $\mathcal{J}\subseteq[\mathsf{K}]$, $|\mathcal{J}|=\mathsf{K}-1$ and $\mathcal{J}\cap\mathcal{U}=\mathcal{K}$. It can be seen that for each worker $j\in[\mathsf{K}]\setminus\mathcal{U}$, among all $V^t_{\mathcal{J}}$ where $\mathcal{J}\subseteq[\mathsf{K}]$, $|\mathcal{J}|=\mathsf{K}-1$ and $\mathcal{J}\cap\mathcal{U}=\mathcal{K}$, worker $j$ knows one of them (which is $V^t_{[\mathsf{K}]\setminus\{\mathsf{u}^{t-1}_{d^t_j}\}}$). We denote all sets $\mathcal{J}\subseteq[\mathsf{K}]$ where $|\mathcal{J}|=\mathsf{K}-1$ and $\mathcal{J}\cap\mathcal{U}=\mathcal{K}$, by $\mathcal{J}_1(\mathcal{K})$, $\mathcal{J}_2(\mathcal{K})$, …, $\mathcal{J}_{\binom{\mathsf{K}-|\mathcal{U}|}{\mathsf{K}-1-|\mathcal{K}|}}(\mathcal{K})$. For $\mathcal{S}_{\mathcal{K}}$, we choose one worker in $\mathcal{K}$ to transmit $V^t_{\mathcal{J}_1(\mathcal{K})}\oplus V^t_{\mathcal{J}_2(\mathcal{K})}$, …, $V^t_{\mathcal{J}_1(\mathcal{K})}\oplus V^t_{\mathcal{J}_{\binom{\mathsf{K}-|\mathcal{U}|}{\mathsf{K}-1-|\mathcal{K}|}}(\mathcal{K})}$, such that each worker in $\mathsf{K}\setminus\mathcal{U}$ can recover all $V^t_{\mathcal{J}}$ where $\mathcal{J}\subseteq[\mathsf{K}]$, $|\mathcal{J}|=\mathsf{K}-1$ and $\mathcal{J}\cap\mathcal{U}=\mathcal{K}$.
- For $\mathcal{K}=\emptyset$, the transmission for $\mathcal{S}_{\mathcal{K}}$ is equivalent to decentralized data shuffling with $\mathsf{K}_{\text{eq}}=\mathsf{K}-|\mathcal{U}|$, $\mathsf{q}_{\text{eq}}=1$ and $\mathsf{M}_{\text{eq}}=m=\mathsf{K}-2$, where $\mathcal{U}_{\text{eq}}=\emptyset$. Hence, in this case $\mathcal{U}\leq 2$.
  If $|\mathcal{U}|=2$, we have $\mathsf{M}_{\text{eq}}=\mathsf{K}_{\text{eq}}$ and thus we do not transmit anything for $\mathcal{S}_{\emptyset}$.
  If $|\mathcal{U}|=1$, we have $\mathsf{M}_{\text{eq}}=\mathsf{K}_{\text{eq}}-1$ and thus we can use Scheme B for $m=\mathsf{K}-1$ to transmit $\mathcal{S}_{\emptyset}$.
  Finally, we consider $|\mathcal{U}|=\emptyset$. For each set $\mathcal{J}\subseteq[\mathsf{K}]$ where $|\mathcal{J}|=m+1=\mathsf{K}-1$, among all the workers in $\mathcal{J}$, there is exactly one worker in $\mathcal{J}$ where $\mathsf{u}^{t-1}_{d^t_k}\notin\mathcal{J}$ (this worker is assumed to be $k$ and we have $\mathsf{u}^{t-1}_{d^t_k}=[\mathsf{K}]\setminus\mathcal{J}$ with a slight abuse of notation). We then let worker $k$ transmit $V^t_{\mathcal{J}}$.

In conclusion, by comparing the loads for different cases, the worst-cases are when $\mathcal{A}^{t-1}_k\cap\mathcal{A}^t_k=\emptyset$ for each $k\in[\mathsf{K}]$ and the worst-case load achieved by Scheme B is

$$\mathsf{q}\mathsf{K}/\binom{\mathsf{K}-1}{\mathsf{K}-2}=\mathsf{q}\frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}\bigg|_{m=\mathsf{K}-1}=:\mathsf{R}_{\text{Ach.B}}|_{\mathsf{M}=(\mathsf{K}-2)\mathsf{q}},\tag{67}$$

which coincides with the converse bound.

*Storage Update Phase of time slot $t\in[T]$ for $m=\mathsf{K}-2$:* The storage update phase for $m=\mathsf{K}-2$ is the same as the one of scheme B for $m=\mathsf{K}-1$.

**Remark 3** (Scheme B realizes distributed interference alignment)**.** *In Example 2, from $\mathcal{A}^{t-1}=(5,1,2,3,4)$ to $\mathcal{A}^t=(1,2,3,4,5)$, by the first column of Table II, we see that each worker desires $\mathsf{K}-2=3$ of the sub-blocks that need to be shuffled. Since each worker cannot benefit from its own transmission, we see that the best possible scheme would have each worker recover its $\mathsf{K}-2=3$ desired sub-blocks from the $\mathsf{K}-1=4$ "useful transmissions," that is, the unwanted sub-blocks should "align" in a single transmission, e.g., for worker 1, all of its unwanted sub-blocks are aligned in $V^t_{\{2,3,4,5\}}$. From the above we see that this is exactly what happens for each worker when $\mathsf{K}-2=m$. How to realize distributed interference alignment seems to be a key question in decentralized data shuffling.* □

**Remark 4** (Extension of Scheme B to other storage sizes)**.** *We can extend Scheme B to any storage size by the following three steps:*

- *We partition the $\mathsf{N}$ data units into $\mathsf{q}$ groups, where each group contains $\mathsf{K}$ data units, and such that during the data shuffling phase each worker requests exactly one data unit and knows exactly one data unit among all the $\mathsf{K}$ data units in each group.*
- *For each group $\mathcal{H}_i$, we partition all desired sub-blocks by all workers into sets depending on which workers in $\mathcal{U}(\mathcal{H}_i)$ know them. Each set is denoted by $\mathcal{S}_{\mathcal{K}}(\mathcal{H}_i)$, which is known by workers in $\mathcal{K}\subseteq\mathcal{U}(\mathcal{H}_i)$, and is defined similarly to* (66)*.*
- *For each set $\mathcal{S}_{\mathcal{K}}(\mathcal{H}_i)$,*
  - *if $\mathcal{K}\neq\emptyset$, the transmission is equivalent to centralized data shuffling with $\mathsf{K}_{eq}=\mathsf{K}-|\mathcal{U}(\mathcal{H}_i)|$, $\mathsf{q}_{eq}=1$ and $\mathsf{M}_{eq}=\mathsf{M}-|\mathcal{K}|$. We can use the optimal centralized data shuffling scheme in [12];*
  - *if $\mathcal{K}=\emptyset$, for each set $\mathcal{J}\subseteq([\mathsf{K}]\setminus\mathcal{U}(\mathcal{H}_i))$, where $|\mathcal{J}|=m+1$, we generate the multicast messages $V^t_{\mathcal{J}}$ as defined in* (65)*.*
    *If there exists some empty sub-block in $V^t_{\mathcal{J}}$, we let the worker who demands this sub-block transmit $V^t_{\mathcal{J}}$. Otherwise, $V^t_{\mathcal{J}}$ is transmitted as Scheme B for $m=\mathsf{K}-1$.*

*Unfortunately, a general closed-form expression for the load in this general case is not available as it heavily depends on the shuffle.*

*Note: in Example 3 next we show the transmission for $\mathcal{K}=\emptyset$, can be further improved by random linear combinations.* □

**Example 3.** Consider the $(\mathsf{K},\mathsf{q},\mathsf{M})=(5,1,2)$ decentralized data shuffling problem, where $m=\mathsf{M}/\mathsf{q}=2$. From the outer bound we have $\mathsf{R}^\star_{\mathsf{u}}\geq 15/8$; if each sub-block is of size $1/\binom{\mathsf{K}-1}{m-1}=1/4$, the outer bound suggests that we need to transmit at least $15/2=7.5$ sub-blocks in total.

Let $\mathcal{A}^{t-1}=(5,1,2,3,4)$. During the storage update phase in time slot $t-1$, we partition each data unit into 4 equal-length sub-blocks, each of which has $\mathsf{B}/4$ bits, as

$$F_1=(F_{1,\{1,2\}},F_{1,\{2,3\}},F_{1,\{2,4\}},F_{1,\{2,5\}}\},\tag{68a}$$
$$F_2=(F_{2,\{1,3\}},F_{2,\{2,3\}},F_{2,\{3,4\}},F_{2,\{3,5\}}\},\tag{68b}$$
$$F_3=(F_{3,\{1,4\}},F_{3,\{2,4\}},F_{3,\{3,4\}},F_{3,\{4,5\}}\},\tag{68c}$$
$$F_4=(F_{4,\{1,5\}},F_{4,\{2,5\}},F_{4,\{3,5\}},F_{4,\{4,5\}}\},\tag{68d}$$
$$F_5=(F_{5,\{1,2\}},F_{5,\{1,3\}},F_{5,\{1,4\}},F_{5,\{1,5\}}\},\tag{68e}$$

and each worker $k$ stores $F_{i,\mathcal{W}}$ if $k\in\mathcal{W}$.

Let $\mathcal{A}^t = (1,2,3,4,5)$. During the data shuffling phase in time slot $t$, each worker must recover 3 sub-blocks of the desired data unit which it does not store, e.g., worker 1 must recover $(F_{1,\{2,3\}}, F_{1,\{2,4\}}, F_{1,\{2,5\}})$, worker 2 must recover $(F_{2,\{1,3\}}, F_{2,\{3,4\}}, F_{2,\{3,5\}})$, etc.

For each set $\mathcal{J} \subseteq [\mathsf{K}]$ where $|\mathcal{J}| = m+1 = 3$, we generate $V^t_{\mathcal{J}} = \underset{k\in\mathcal{J}}{\oplus} F_{k,\mathcal{J}\setminus\{k\}}$ as in (65). More precisely, we have

$$V^t_{\{1,2,3\}} = F_{1,\{2,3\}} \oplus F_{2,\{1,3\}}, \text{ (can be sent by worker 3)}, \tag{69a}$$

$$V^t_{\{1,2,4\}} = F_{1,\{2,4\}}, \tag{69b}$$

$$V^t_{\{1,2,5\}} = F_{1,\{2,5\}} \oplus F_{5,\{1,2\}}, \text{ (can be sent by worker 2)}, \tag{69c}$$

$$V^t_{\{1,3,4\}} = F_{3,\{1,4\}}, \tag{69d}$$

$$V^t_{\{1,3,5\}} = F_{5,\{1,3\}}, \tag{69e}$$

$$V^t_{\{1,4,5\}} = F_{4,\{1,5\}} \oplus F_{5,\{1,4\}}, \text{ (can be sent by worker 1)}, \tag{69f}$$

$$V^t_{\{2,3,4\}} = F_{2,\{3,4\}} \oplus F_{3,\{2,4\}}, \text{ (can be sent by worker 4)}, \tag{69g}$$

$$V^t_{\{2,3,5\}} = F_{2,\{3,5\}}, \tag{69h}$$

$$V^t_{\{2,4,5\}} = F_{4,\{2,5\}}, \tag{69i}$$

$$V^t_{\{3,4,5\}} = F_{3,\{4,5\}} \oplus F_{4,\{3,5\}}, \text{ (can be sent by worker 5)}. \tag{69j}$$

We deliver these multicast messages with a two-phase scheme, as follows.

- Phase 1. It can be seen that the multicast messages like $V^t_{\{1,2,3\}}$ in (69) (which is known by worker 3 only) can be sent by one specific worker. Similarly, we can let workers 2, 1, 4 and 5 broadcast $V^t_{\{1,2,5\}}$, $V^t_{\{1,4,5\}}$, $V^t_{\{2,3,4\}}$ and $V^t_{\{3,4,5\}}$, respectively.
- Phase 2. After the above Phase 1, the remaining messages are known by two workers. For example, $V^t_{\{1,2,4\}} = F_{1,\{2,4\}}$ is known by workers 2 and 4; we can let worker 2 transmit $V^t_{\{1,2,4\}}$. If we do so as Scheme B, since each multicast message in (69) has $\mathsf{B}/4$ bits and there are 10 multicast messages in (69), the total load is $10/4$.
  In this example Phase 2 of Scheme B can be improved as follows. The problem with the above strategy (i.e., assign each multicast message to a worker) is that we have not leveraged the fact that, after Phase 1, there are still five sub-blocks to be delivered (one demanded per worker, namely $F_{1,\{2,4\}}, F_{2,\{3,5\}}, F_{3,\{1,4\}}, F_{4,\{2,5\}}, F_{5,\{1,3\}}$), each of which is known by two workers. Therefore, we can form random linear combinations so that each worker can recover all three of the unstored sub-blocks. In other words, if a worker receivers from each of the remaining $\mathsf{K}-1=4$ workers $3/4\times$"size of a sub-block" linear equations then it solved for the three missing sub-blocks, that is, each worker broadcasts $\frac{3\mathsf{B}}{16}$ random linear combinations of all bits of the two sub-blocks he stores among $F_{1,\{2,4\}}, F_{2,\{3,5\}}, F_{3,\{1,4\}}, F_{4,\{2,5\}}, F_{5,\{1,3\}}$. It can be checked that for worker 1, the received $3\mathsf{B}/4$ random linear combinations from other workers are linearly independent known $F_{1,\{2,4\}}$ and $F_{5,\{1,3\}}$ as $\mathsf{B}\to\infty$, such that it can recover all these five sub-blocks. By the symmetry, each other worker can also recover these five sub-blocks.

In conclusion, the total load with this two-phase is $5(1+3/4)\times 1/4 = \frac{35}{16} < 10/4$, which is achieved by Scheme B. By comparison, the load of Scheme A is $\frac{27}{8}$ and the converse bound under the constraint of uncoded storage in Theorem 1 is $\frac{15}{8}$.

As a final remark, note that the five sub-blocks in Phase 2 is symmetric, i.e., the number of sub-blocks stored by each worker is the same and the one known by each worker is also the same. In general case, this symmetry may not hold (thus the conditional linearly independent of the received random linear combinations by each worker may not hold), and the generalization of this scheme is part of ongoing work. □

**Remark 5** (Limitation of Scheme B for small storage size)**.** *The data shuffling phase with uncoded storage can be represented by a directed graph, where each sub-block demanded by some worker is represented by a node in the graph. A directed edge exists from node $a$ to node $b$ in the graph if the worker demanding the data represented by node $b$ has the data represented by node $a$ in its storage. By generating a directed graph for the data shuffling phase as described in Section II, we can see that each multicast message in* (65) *is the sum of the sub-blocks contained in a clique, where a clique is a set of nodes where each two of them are connected in two directions. The sub-blocks in each multicast message in* (65) *also form a distributed clique, where a distributed clique is a clique whose nodes are all known by some worker.*

*Consider the case where $\mathsf{K} = \mathsf{N}$ is much larger than $\mathsf{M} = m = 2$ (i.e., small storage size regime). Consider a "cyclic shuffle" of the form $\mathcal{A}^{t-1}_1 = \{\mathsf{K}\}$ and $\mathcal{A}^{t-1}_k = \{k-1\}$ for $k \in [2:\mathsf{K}]$, to $\mathcal{A}^t_k = \{k\}$ for $k \in [\mathsf{K}]$. Each data unit is split into $\binom{\mathsf{K}-1}{m-1} = \mathsf{K}-1$ sub-blocks and each worker needs to recover $\binom{\mathsf{K}-2}{m-1} = \mathsf{K}-2$ sub-blocks.*

*If during the data shuffling phase in time slot $t$ we generate the multicast messages as above, only 2 of the $\mathsf{K}-2$ demanded sub-blocks are in a distributed clique of size $m=2$. More precisely, let us focus on worker 1 who needs to recover $F_1$. Notice that each sub-block of $F_1$ is stored by worker 2, and each of its demanded sub-blocks $F_{1,\{2,j\}}$ where $j \in [\mathsf{K}]\setminus\{1,2\}$, is in the multicast message*

$$V^t_{\{1,2,j\}} = F_{1,\{2,j\}} \oplus F_{2,\{1,j\}} \oplus F_{j,\{1,2\}}. \tag{70}$$

*When $j=3$, it can be seen that $F_{3,\{1,2\}}$ is empty because all sub-blocks of $F_3$ are stored by worker 4, and thus $V^t_{\{1,2,3\}} = F_{1,\{2,3\}} \oplus F_{2,\{1,3\}}$ could be transmitted by worker 3. However, when $j=4$, both sub-blocks $F_{2,\{1,4\}}$ and $F_{4,\{1,2\}}$ are empty, and thus $V^t_{\{1,2,4\}} = F_{1,\{2,4\}}$. Similarly, among all the $\mathsf{K}-2$ demanded sub-blocks by worker 1, only $F_{1,\{2,3\}}$ and $F_{1,\{2,\mathsf{K}\}}$ are in cliques including two sub-blocks, while the remaining $\mathsf{K}-4$ ones are in cliques including only one sub-block.*

*If the delivery strategy is to assign each multicast message, or distributed clique, to a worker, we see that most of the distributed cliques have a multicast coding gain of 1 (where the multicast coding gain is the gain on the transmitted load compared to uncoded/direct transmission, e.g., if we transmit*

*one sub-block in a distributed clique of length* 2*, the multicast coding gain to transmit this sub-block is* 2*). Hence, Scheme B is generally inefficient for* $m \in [2 : \mathsf{K}-3]$*. In this paper we mainly use Scheme B for* $m \in \{\mathsf{K}-2, \mathsf{K}-1, \mathsf{K}\}$ *for which it is optimal under the constraint of uncoded storage.* □

### C. Scheme C in Theorem 4

To overcome the limitation of Scheme B described in Remark 5, in the following we propose Scheme C for $\mathsf{M}/\mathsf{q} = m = 2$ based on an *unconventional distributed clique-covering* strategy for the two-sender distributed index coding problems proposed in [29].

The storage update phase of Scheme C is the same as Scheme B, which is structural invariant, and thus we only describe the transition from time slot $t-1$ to time slot $t$. The main idea is not to use the conventional distributed clique-covering method which transmits distributed cliques (e.g, the multicast messages in Scheme B), because most of the distributed cliques only contain one sub-block, and most sub-blocks are not in any distributed clique including more than one sub-blocks, as explained in Remark 5. Instead, we propose a novel decentralized data shuffling scheme to increase the efficiency (coded multicasting gain) of the transmissions. The main idea is that we search for cliques with length $m = 2$; if this clique is not a distributed clique, we add one node to the clique and transmit the three nodes by two binary sums, such that each node can be recovered by the corresponding user; if this clique is a distributed clique we proceed as in Scheme B.

Before introducing the details of Scheme C, we first recall the unconventional clique-covering method for the two-sender distributed index coding problems proposed in [29, Theorem 8, Case 33].

**Proposition 1** (Unconventional Distributed Clique-covering in [29])**.** *In a distributed index coding problem with two senders (assumed to be* $k_1$ *and* $k_2$*), as illustrated Fig. 3, there are three messages* $a$ *(demanded by receiver* $u(a)$*),* $b$ *(demanded by receiver* $u(b)$*), and* $c$ *(demanded by receiver* $u(c)$*), where* $u(a)$ *stores* $b$*,* $u(b)$ *stores* $a$*, and* $u(c)$ *stores* $a$*. Sender* $k_1$ *stores* $a$ *and* $c$*, and sender* $k_2$ *stores* $b$ *and* $c$*. We can let worker* $k_1$ *transmit* $a \oplus c$ *and worker* $k_2$ *transmit* $b \oplus c$*, such that workers* $u(a)$*,* $u(b)$ *and* $u(c)$ *can recover node* $a$*,* $b$ *and* $c$*, respectively.*

*Indeed, receiver* $u(a)$ *knows* $b$ *such that it can recover* $c$*, and then recover* $a$*. Similarly receiver* $u(b)$ *can recover* $b$*. User* $u(c)$ *knows* $a$*, such that it can recover* $c$ *from* $a \oplus c$*.*

In the decentralized data shuffling problem, each data unit is divided into sub-blocks depending on which subset of workers stored them before the data shuffling phase; each sub-block desired by a worker is an independent message in the corresponding distributed index coding problem; thus the data shuffling phase is a $\mathsf{K}$-sender distributed index coding problem that contains a number of messages that in general is doubly exponential in the number of workers in the original decentralized data shuffling problem. Hence, it is non-trivial to use Proposition 1 in the decentralized data shuffling problem.

We then illustrate the main idea of Scheme C by means of an example.

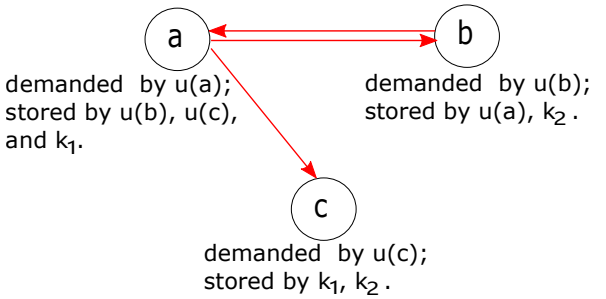



Fig. 3: Directed graph of the two-sender ($k_1$ and $k_2$) distributed index problem in Proposition 1. A direct edge from node $a$ to node $b$, means that receiver $u(b)$ demanding message $b$ stores message $a$ demanded by receiver $u(a)$.

**Example 4.** We consider the same example as Remark 4, where $\mathsf{K} = 5$, $\mathsf{q} = 1$ and $\mathsf{M} = 2$. Let $\mathcal{A}_1^{t-1} = \{5\}$ and $\mathcal{A}_k^{t-1} = \{k-1\}$ for $k \in [2:5]$, and $\mathcal{A}_k^t = \{k\}$ for $k \in [5]$. The data units are split as in (68).

In the data shuffling phase of time slot $t$, the distributed clique-covering method in Scheme B has many sub-blocks which are not in any distributed clique including more than one sub-block (e.g., $F_{1,\{2,4\}}$), as explained in Remark 5. Moreover, the sub-blocks $F_{1,\{2,3\}}$ and $F_{3,\{1,4\}}$ are in a clique in the graph, but none of the workers can transmit $F_{1,\{2,3\}} \oplus F_{3,\{1,4\}}$, and thus it is not a distributed clique. However, if we add $F_{1,\{2,4\}}$ to the group, and transmit the two sums $F_{1,\{2,3\}} \oplus F_{1,\{2,4\}}$ (by worker 2) and $F_{3,\{1,4\}} \oplus F_{1,\{2,4\}}$ (by worker 4), we see that worker 1 (who knows $F_{3,\{1,4\}}$) can recover $F_{1,\{2,4\}}$ from the second sum, and then recover $F_{1,\{2,3\}}$ from the first sum. Similarly, worker 3 (who knows $F_{1,\{2,3\}}$) can recover $F_{1,\{2,4\}}$ from the first sum, and then recover $F_{3,\{1,4\}}$ from the second sum. It can be seen that $F_{1,\{2,3\},1}$ is message $a$, $F_{3,\{1,4\},1}$ is message $b$, and $F_{1,\{2,4\},1}$ is message $c$ in Proposition 1, while $u(a)$ and $u(c)$ are both worker 1, $u(b)$ is worker 3. In addition, workers 2 and 4 serve as senders $k_1$ and $k_2$.

Similarly, the sub-blocks $F_{1,\{2,4\}}$ and $F_{4,\{1,5\}}$ are in a clique in the graph, but none of the workers can transmit $F_{1,\{2,4\}} \oplus F_{4,\{1,5\}}$. However, if we add $F_{1,\{2,5\}}$ to the group, and transmit $F_{1,\{2,4\}} \oplus F_{1,\{2,5\}}$ (by worker 2) and $F_{1,\{2,5\}} \oplus F_{4,\{1,5\}}$ (by worker 5), then worker 1 (who knows $F_{4,\{1,5\}}$) can recover $F_{1,\{2,5\}}$ from the second sum, and then recover $F_{1,\{2,4\}}$ from the first sum; also, worker 4 (who knows $F_{1,\{2,4\}}$) can recover $F_{1,\{2,5\}}$ from the first sum, and then recover $F_{4,\{1,5\}}$ from the second sum.

In general, we have the following data shuffling scheme in time slot $t$. Recall that $\mathsf{u}_i^{t-1}$ denotes the worker who should recover $F_i$ at the end of time slot $t-1$. We partition each sub-block into 3 equal-length sub-pieces as $F_{i,\mathcal{W}} = (F_{i,\mathcal{W},1}, F_{i,\mathcal{W},2}, F_{i,\mathcal{W},3})$ (recall $|\mathcal{W}| = m = 2$ and $\mathsf{u}_i^{t-1} \in \mathcal{W}$). In general, we consider a pair $(a, b)$, where $a \in [5]$ and $b \in [5] \setminus \{a, \mathsf{u}_a^{t-1}\}$, i.e., $a$ is a worker with the request $F_a$ in time slot $t$, while $b$ is another worker which is not the one who requests $F_a$ in time slot $t-1$. We have two cases:

1) If $a \neq \mathsf{u}_b^{t-1}$, we find the group of sub-blocks $F_{a,\{\mathsf{u}_a^{t-1},b\}}$, $F_{a,\{\mathsf{u}_a^{t-1},\mathsf{u}_b^{t-1}\}}$, and $F_{b,\{a,\mathsf{u}_b^{t-1}\}}$. We pick

one of untransmitted sub-pieces for each of these three sub-blocks. Assume the picked sub-pieces are $n_1$, $n_2$, and $n_3$, respectively. We let worker $\mathsf{u}_a^{t-1}$ transmit $F_{a,\{\mathsf{u}_a^{t-1},b\},n_1} \oplus F_{a,\{\mathsf{u}_a^{t-1},\mathsf{u}_b^{t-1}\},n_2}$, and let worker $\mathsf{u}_b^{t-1}$ transmit $F_{a,\{\mathsf{u}_a^{t-1},\mathsf{u}_b^{t-1}\},n_2} \oplus F_{b,\{a,\mathsf{u}_b^{t-1}\},n_3}$. It can be seen that worker $a$ can recover $F_{a,\{\mathsf{u}_a^{t-1},b\},n_1}$ and $F_{a,\{\mathsf{u}_a^{t-1},\mathsf{u}_b^{t-1}\},n_2}$ while worker $b$ can recover $F_{b,\{a,\mathsf{u}_b^{t-1}\},n_3}$.

2) If $a = \mathsf{u}_b^{t-1}$, we assume $c = \mathsf{u}_a^{t-1}$ and $d = \mathsf{u}_c^{t-1}$ (e.g, if $a = 1$ and $b = 5$, we have $c = \mathsf{u}_1^{t-1} = 2$ and $d = \mathsf{u}_c^{t-1} = 3$), i.e., worker $a$ requests $F_a$ in time slot $t$, worker $c$ requests $F_a$ in time slot $t-1$ and requests $F_c$ in time slot $t$, worker $d$ requests $F_c$ in time slot $t-1$ and requests $F_d$ in time slot $t$. We find the group of sub-blocks $F_{a,\{c,b\}}$, $F_{a,\{c,d\}}$, and $F_{c,\{a,d\}}$. Notice that $F_{a,\{c,d\}}$ and $F_{c,\{a,d\}}$ form a distributed clique. We pick one of untransmitted sub-pieces for each of these three sub-blocks (assumed to be $n_1$, $n_2$, and $n_3$, respectively). We let worker $c$ transmit $F_{a,\{c,b\},n_1}$, and let worker $d$ transmit $F_{a,\{c,d\},n_2} \oplus F_{c,\{a,d\},n_3}$. It can be seen that worker $a$ can recover $F_{a,\{c,b\},n_1}$ and $F_{a,\{c,d\},n_2}$ while worker $c$ can recover $F_{c,\{a,d\},n_3}$.

By this construction, see also Table III, each sub-block appears in three groups such that each of its sub-pieces is transmitted. We use two binary sums to encode each group containing 3 sub-pieces, such that the coding gain of this scheme is $2/3$. The achieved worst-case load is $5/2$, while the achieved loads by Schemes A and B are $\frac{27}{8}$ and $\frac{10}{4}$, respectively.

The introduced scheme in this example has the same load as Scheme B. However, in general when $\mathsf{M} = 2\mathsf{q}$, the coding gain of the new scheme is $2/3$, and this is independent of $\mathsf{K}$. For the other schemes, the coding gain of Scheme B is close to 1 if $\mathsf{K}$ is large, and the same holds for Scheme A (as it will be shown in Section VI-D). Therefore, Scheme C is preferable to the other schemes when $\mathsf{M} = 2\mathsf{q}$ and $\mathsf{K}$ is large. □

We are now ready to generalize the scheme in Example 4 to the case $\mathsf{M}/\mathsf{q} = m = 2$.

*Structural Invariant Data Partitioning and Storage:* This is the same as in Scheme B (described in Section VI-B), i.e., each sub-block of $F_i, i \in [\mathsf{N}]$, is stored by worker $\mathsf{u}_i^t$ and by another worker in $[\mathsf{K}] \setminus \{\mathsf{u}_i^t\}$. The length of each sub-block is $\frac{\mathsf{B}}{\binom{\mathsf{K}-1}{m-1}} = \frac{\mathsf{B}}{\mathsf{K}-1}$.

*Data Shuffling Phase of time slot $t \in [T]$:* As in Scheme B, we partition the $\mathsf{N}$ data units into $\mathsf{q}$ equal-length groups such that, during the data shuffling phase of time slot $t$, among all the $\mathsf{K}$ data units in each group, each worker requests exactly one data unit and knows exactly one data unit. To simplify the description, as in Scheme B, we focus on one group and remove the $\mathcal{H}_i$ in the notation. In addition, we can restrict attention to $\mathcal{U} = \emptyset$, because if $\mathcal{U} \neq \emptyset$ we can divide all desired sub-blocks by all workers into sets as we did in Scheme B. For each set which is known by some worker in $\mathcal{U}$, the transmission for this set is equivalent to centralized data shuffling. Thus we only need to consider the set which is not known by any worker in $\mathcal{U}$, and the transmission for this set is equivalent to decentralized data shuffling with $\mathsf{K}_{\text{eq}} = \mathsf{K} - |\mathcal{U}|$, $\mathsf{q}_{\text{eq}} = 1$ and $\mathsf{M}_{\text{eq}} = m$. Hence, for the simplicity, we only consider $\mathcal{U} = \emptyset$ for Scheme C.

We define a set of pair of workers as

$$\mathcal{Y} := \Big\{(a,b) : a \in [\mathsf{K}], b \in \big([\mathsf{K}] \setminus \{\mathsf{u}_{d_a^t}^{t-1}, a\}\big)\Big\}. \tag{71}$$

For each vector $(a,b) \in \mathcal{Y}$, we divide $F_{d_a^t,\{b,\mathsf{u}_{d_a^t}^{t-1}\}}$ into 3 non-overlapping and equal-length sub-pieces, $F_{d_a^t,\{b,\mathsf{u}_{d_a^t}^{t-1}\},1}$, $F_{d_a^t,\{b,\mathsf{u}_{d_a^t}^{t-1}\},2}$ and $F_{d_a^t,\{b,\mathsf{u}_{d_a^t}^{t-1}\},3}$. For each vector $(a,b) \in \mathcal{Y}$, we consider two cases:

- Case $\mathsf{u}_{d_b^t}^{t-1} \neq a$: For each one of $F_{d_a^t,\{\mathsf{u}_{d_a^t}^{t-1},b\}}$, $F_{d_b^t,\{\mathsf{u}_{d_b^t}^{t-1},a\}}$, and $F_{d_a^t,\{\mathsf{u}_{d_a^t}^{t-1},\mathsf{u}_{d_b^t}^{t-1}\}}$, we select one of its non-transmitted sub-pieces and assume they are $F_{d_a^t,\{\mathsf{u}_{d_a^t}^{t-1},b\},n_1}$, $F_{d_b^t,\{\mathsf{u}_{d_b^t}^{t-1},a\},n_2}$, and $F_{d_a^t,\{\mathsf{u}_{d_a^t}^{t-1},\mathsf{u}_{d_b^t}^{t-1}\},n_3}$. By Proposition 1,

  worker $\mathsf{u}_{d_a^t}^{t-1}$ transmits $F_{d_a^t,\{\mathsf{u}_{d_a^t}^{t-1},b\},n_1} \oplus F_{d_a^t,\{\mathsf{u}_{d_a^t}^{t-1},\mathsf{u}_{d_b^t}^{t-1}\},n_3}$;

  worker $\mathsf{u}_{d_b^t}^{t-1}$ transmits $F_{d_b^t,\{\mathsf{u}_{d_b^t}^{t-1},a\},n_2} \oplus F_{d_a^t,\{\mathsf{u}_{d_a^t}^{t-1},\mathsf{u}_{d_b^t}^{t-1}\},n_3}$,

  such that each of the above linear combinations can be decoded by its requesting worker. For example in Table III, for pair $(1,3)$, we let worker 2 transmit $F_{1,\{2,3\},1} \oplus F_{1,\{2,4\},1}$, and worker 4 transmit $F_{1,\{2,4\},1} \oplus F_{3,\{1,4\},1}$.
- Case $\mathsf{u}_{d_b^t}^{t-1} = a$: Let $\mathsf{u}_{d_a^t}^{t-1} = c$ and $\mathsf{u}_{d_c^t}^{t-1} = d$. For each one of $F_{d_a^t,\{c,b\}}$, $F_{d_a^t,\{c,d\}}$, and $F_{d_c^t,\{a,d\}}$, we select one of its non-transmitted sub-pieces and assume they are $F_{d_a^t,\{c,b\},n_1}$, $F_{d_a^t,\{c,d\},n_2}$, and $F_{d_c^t,\{a,d\},n_3}$. By Proposition 1,

  worker $c$ transmits $F_{d_a^t,\{c,b\},n_1}$;

  worker $d$ transmit $F_{d_a^t,\{c,d\},n_2} \oplus F_{d_c^t,\{a,d\},n_3}$,

  such that each of the above sub-pieces can be decoded by its requesting worker. For example in Table III, for vector $(1,5)$, we let worker 2 transmit $F_{1,\{2,5\},2}$ and worker 3 transmit $F_{1,\{2,3\},2} \oplus F_{2,\{1,3\},1}$. Notice that in the case if $d = b$, we have $F_{d_a^t,\{c,b\}} = F_{d_a^t,\{c,d\}}$, and thus we transmit two different sub-pieces of $F_{d_a^t,\{c,b\}}$ for the vector $(a,b)$.

Next we prove that after considering all the pairs in $\mathcal{Y}$, each sub-piece of sub-block $F_{d_{a_1}^t,\{\mathsf{u}_{d_{a_1}^t}^{t-1},b_1\}}$ has been transmitted for $(a_1,b_1) \in \mathcal{Y}$. For each $(a_1,b_1) \in \mathcal{Y}$, if there exists one worker $c \notin \{a_1,b_1\}$ such that $c = \mathsf{u}_{d_{a_1}^t}^{t-1}$, $b_1 = \mathsf{u}_{d_c^t}^{t-1}$, and $a_1 = \mathsf{u}_{d_{b_1}^t}^{t-1}$ (in Example 4, this pair $(a_1,b_1)$ does not exist), we transmit two sub-pieces of $F_{d_{a_1}^t,\{\mathsf{u}_{d_{a_1}^t}^{t-1},b_1\}}$ in the transmission for the pair $(a_1,b_1)$ and one sub-piece in the transmission for the pair $(b_1,c)$. Otherwise, we transmit the three sub-pieces of $F_{d_{a_1}^t,\{\mathsf{u}_{d_{a_1}^t}^{t-1},b_1\}}$ in the following three transmissions for three different pairs:

1) The transmission for the pair $(a_1,b_1)$.
2) The transmission for the pair $(a_1,b_2)$ where $\mathsf{u}_{d_{b_2}^t}^{t-1} = b_1$ if the requested data unit by worker $\mathsf{u}_{d_{a_1}^t}^{t-1}$ in time slot $t$ was not stored by worker $b_1$ at the end of time slot

TABLE III: Transmission of Scheme C for Example 4

| Considered vectors | Groups of sub-pieces | First sum | Second sum |
|---|---|---|---|
| $(1,3)$ | $F_{1,\{2,3\},1}, F_{1,\{2,4\},1}, F_{3,\{1,4\},1}$ | $F_{1,\{2,3\},1} \oplus F_{1,\{2,4\},1}$ | $F_{1,\{2,4\},1} \oplus F_{3,\{1,4\},1}$ |
| $(1,4)$ | $F_{1,\{2,4\},2}, F_{1,\{2,5\},1}, F_{4,\{1,5\},1}$ | $F_{1,\{2,4\},2} \oplus F_{1,\{2,5\},1}$ | $F_{1,\{2,5\},1} \oplus F_{4,\{1,5\},1}$ |
| $(1,5)$ | $F_{1,\{2,5\},2}, F_{1,\{2,3\},2}, F_{2,\{1,3\},1}$ | $F_{1,\{2,5\},2}$ | $F_{1,\{2,3\},2} \oplus F_{2,\{1,3\},1}$ |
| $(2,4)$ | $F_{2,\{3,4\},1}, F_{2,\{3,5\},1}, F_{4,\{2,5\},1}$ | $F_{2,\{3,4\},1} \oplus F_{2,\{3,5\},1}$ | $F_{2,\{3,5\},1} \oplus F_{4,\{2,5\},1}$ |
| $(2,5)$ | $F_{2,\{3,5\},2}, F_{2,\{1,3\},2}, F_{5,\{1,2\},1}$ | $F_{2,\{3,5\},2} \oplus F_{2,\{1,3\},2}$ | $F_{2,\{1,3\},2} \oplus F_{5,\{1,2\},1}$ |
| $(2,1)$ | $F_{2,\{1,3\},3}, F_{2,\{3,4\},2}, F_{3,\{2,4\},1}$ | $F_{2,\{1,3\},3}$ | $F_{2,\{3,4\},2} \oplus F_{3,\{2,4\},1}$ |
| $(3,5)$ | $F_{3,\{4,5\},1}, F_{3,\{1,4\},2}, F_{5,\{1,3\},1}$ | $F_{3,\{4,5\},1} \oplus F_{3,\{1,4\},2}$ | $F_{3,\{1,4\},2} \oplus F_{5,\{1,3\},1}$ |
| $(3,1)$ | $F_{3,\{1,4\},3}, F_{3,\{2,4\},2}, F_{1,\{2,3\},3}$ | $F_{3,\{1,4\},3} \oplus F_{3,\{2,4\},2}$ | $F_{3,\{2,4\},2} \oplus F_{1,\{2,3\},3}$ |
| $(3,2)$ | $F_{3,\{2,4\},3}, F_{3,\{4,5\},2}, F_{4,\{3,5\},1}$ | $F_{3,\{2,4\},3}$ | $F_{3,\{4,5\},2} \oplus F_{4,\{3,5\},1}$ |
| $(4,1)$ | $F_{4,\{1,5\},2}, F_{4,\{2,5\},2}, F_{1,\{2,4\},3}$ | $F_{4,\{1,5\},2} \oplus F_{4,\{2,5\},2}$ | $F_{4,\{2,5\},2} \oplus F_{1,\{2,4\},3}$ |
| $(4,2)$ | $F_{4,\{2,5\},3}, F_{4,\{3,5\},2}, F_{2,\{3,4\},3}$ | $F_{4,\{2,5\},3} \oplus F_{4,\{3,5\},2}$ | $F_{4,\{3,5\},2} \oplus F_{2,\{3,4\},3}$ |
| $(4,3)$ | $F_{4,\{3,5\},3}, F_{4,\{1,5\},3}, F_{5,\{1,4\},1}$ | $F_{4,\{3,5\},3}$ | $F_{4,\{1,5\},3} \oplus F_{5,\{1,4\},1}$ |
| $(5,2)$ | $F_{5,\{1,2\},2}, F_{5,\{1,3\},2}, F_{2,\{3,5\},3}$ | $F_{5,\{1,2\},2} \oplus F_{5,\{1,3\},2}$ | $F_{5,\{1,3\},2} \oplus F_{2,\{3,5\},3}$ |
| $(5,3)$ | $F_{5,\{1,3\},3}, F_{5,\{1,4\},2}, F_{3,\{4,5\},3}$ | $F_{5,\{1,3\},3} \oplus F_{5,\{1,4\},2}$ | $F_{5,\{1,4\},2} \oplus F_{3,\{4,5\},3}$ |
| $(5,4)$ | $F_{5,\{1,4\},3}, F_{5,\{1,2\},3}, F_{1,\{2,5\},3}$ | $F_{5,\{1,4\},3}$ | $F_{5,\{1,2\},3} \oplus F_{1,\{2,5\},3}$ |

$t-1$, (e.g., in Table III, let $(a_1,b_1)=(1,4)$, one sub-piece of $F_{1,\{2,4\}}$ appears in the transmission for vector $(a_1,b_2)=(1,3)$).

Otherwise, the transmission for the pair $(a_1,b_3)$ where $\mathsf{u}_{d^t_{b_3}}^{t-1}=a_1$ (e.g., in Table III, let $(a_1,b_1)=(1,3)$, one sub-piece of $F_{1,\{2,3\}}$ appears in the transmission for vector $(a_1,b_3)=(1,5)$).

3) The transmission for the pair $(b_1,a_1)$ if $\mathsf{u}_{d^t_{b_1}}^{t-1}\neq a_1$ (e.g., in Table III, let $(a_1,b_1)=(1,3)$, one sub-piece of $F_{1,\{2,3\}}$ appears in the transmission for vector $(b_1,a_1)=(3,1)$).

Otherwise, the transmission for the pair $(b_1,b_4)$ where $\mathsf{u}_{d^t_{b_4}}^{t-1}=b_1$ (e.g., in Table III, let $(a_1,b_1)=(1,5)$, one sub-piece of $F_{1,\{2,5\}}$ appears in the transmission for vector $(b_1,b_4)=(5,4)$).

This shows that $F_{d^t_{a_1},\{\mathsf{u}_{d^t_{a_1}}^{t-1},b_1\}}$ is transmitted. In Scheme C, we transmit each three sub-pieces in two sums, and thus the multicast coding gain is $2/3$.

Finally, by comparing the loads for different cases, the worst-cases are when $\mathcal{A}_k^{t-1}\cap\mathcal{A}_k^t=\emptyset$ for each $k\in[\mathsf{K}]$ and the worst-case load achieved by Scheme C is

$$\mathsf{q}\frac{2\mathsf{K}(\mathsf{K}-2)}{3(\mathsf{K}-1)} =: \mathsf{R}_{\text{Ach.C}}|_{\mathsf{M}=2\mathsf{q}}, \tag{72}$$

which is optimal within a factor $4/3$ compared to the converse bound $\frac{\mathsf{K}(\mathsf{K}-2)}{2(\mathsf{K}-1)}$ under the constraint of uncoded storage for $\mathsf{M}/\mathsf{q}=2$.

*Storage Update Phase of time slot $t\in[T]$:* The storage update phase of Scheme C is the same as the one of Scheme B.

### *D. Optimality Results of the Combined Achievable Scheme*

Since the proposed converse bound is a piecewise linear curve with corner points $\big(m\mathsf{q},\mathsf{q}\frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}\big)$ for $m\in[\mathsf{K}]$ and these corner points are successively convex, it follows immediately that the combined scheme in Corollary 1 is optimal under the constraint of uncoded storage when $\mathsf{M}/\mathsf{q}=1$ or $\mathsf{M}/\mathsf{q}\in[\mathsf{K}-2,\mathsf{K}]$, thus proving Theorem 5 (and also Corollary 2).

In order to prove Theorem 6 (i.e., an order optimality result for the cases not covered by Theorem 5 or Corollary 2) we proceed as follows. Recall that the proposed converse bound is a piecewise linear curve with successively convex corner points, and that the straight line in the storage-load tradeoff between two achievable points is also achievable by memory-sharing. Hence, in the following, we focus on each corner point of the converse bound $\big(m\mathsf{q},\mathsf{q}\frac{\mathsf{K}-m}{m}\frac{\mathsf{K}}{\mathsf{K}-1}\big)$ where $m\in[\mathsf{K}]$, and characterize the multiplicative gap between the combined scheme and the converse when $\mathsf{M}=m\mathsf{q}$. Thus the multiplicative gap between the achievability and the converse curves is upper bounded by the maximum of the obtained $\mathsf{K}$ gaps.

We do not consider the corner points where $m\in\{1,\mathsf{K}-2,\mathsf{K}-1,\mathsf{K}\}$ because the optimality of the combined scheme has been proved. We have:

- $\mathsf{M}=2\mathsf{q}$: It was proved in Section VI-C that the multiplicative gap between the Scheme C and the converse bound is $4/3$.
- Interpolate the achievable bound for Scheme A in (21) between $\mathsf{M}_1=\big(1+g\frac{\mathsf{K}-1}{\mathsf{K}}\big)\mathsf{q}$ and $\mathsf{M}_2=\big(1+(g+1)\frac{\mathsf{K}-1}{\mathsf{K}}\big)\mathsf{q}$ to match the converse bound in (20) at $\mathsf{M}_3=(g+1)\mathsf{q}$ where $g\in[2:\mathsf{K}-4]$: For each $g\in[2:\mathsf{K}-4]$, we have

$$\mathsf{M}_1=\left(1+g\frac{\mathsf{K}-1}{\mathsf{K}}\right)\mathsf{q},\ \mathsf{R}_{\text{Ach.A}}(\mathsf{M}_1)=\mathsf{q}\frac{\mathsf{K}-g}{g}, \tag{73}$$

$$\mathsf{M}_2=\left(1+(g+1)\frac{\mathsf{K}-1}{\mathsf{K}}\right)\mathsf{q},\ \mathsf{R}_{\text{Ach.A}}(\mathsf{M}_2)=\mathsf{q}\frac{\mathsf{K}-g-1}{g+1}. \tag{74}$$

By memory-sharing between $(\mathsf{M}_1,\mathsf{R}_{\text{Ach.A}}(\mathsf{M}_1))$ and $(\mathsf{M}_2,\mathsf{R}_{\text{Ach.A}}(\mathsf{M}_2))$ with coefficient $\alpha=(\mathsf{K}-1-g)/(\mathsf{K}-1)$, we get

$$\begin{aligned}\mathsf{M}_3&=\alpha\mathsf{M}_1+(1-\alpha)\mathsf{M}_2|_{\alpha=1-g/(\mathsf{K}-1)}\\&=(1+g)\mathsf{q},\end{aligned} \tag{75}$$

as in the converse bound for $m=g+1\in[3:\mathsf{K}-3]$, and

$$\begin{aligned}\mathsf{R}_{\text{Ach.A}}(\mathsf{M}_3)&=\alpha\mathsf{R}_{\text{Ach.A}}(\mathsf{M}_1)+(1-\alpha)\mathsf{R}_{\text{Ach.A}}(\mathsf{M}_2)\\&=\frac{\mathsf{q}\mathsf{K}-g}{g}\frac{\mathsf{K}-1-g}{\mathsf{K}-1}+\frac{\mathsf{q}g}{\mathsf{K}-1}\frac{\mathsf{K}-g-1}{g+1}\\&=\frac{\mathsf{q}(\mathsf{K}-g-1)(\mathsf{K}g+\mathsf{K}-g)}{(\mathsf{K}-1)g(g+1)}.\end{aligned} \tag{76}$$

From (the proof of) Theorem 1, we know that

$$\mathsf{R}_{\text{Out}}(\mathsf{M}_3) \geq \mathsf{N}\frac{1-g/(\mathsf{K}-1)}{g+1} = \mathsf{q}\frac{\mathsf{K}}{\mathsf{K}-1}\frac{\mathsf{K}-g-1}{g+1}. \tag{77}$$

Hence, from (76) and (77), we have

$$\begin{aligned}\frac{\mathsf{R}_{\text{Ach.A}}(\mathsf{M}_3)}{\mathsf{R}_{\text{Out}}(\mathsf{M}_3)} &\leq \frac{\mathsf{K}g+\mathsf{K}-g}{g\mathsf{K}} = 1-\frac{1}{\mathsf{K}}+\frac{1}{g}\\ &\leq 1-0+\frac{1}{2} = \frac{3}{2} \text{ (since } g\geq 2).\end{aligned} \tag{78}$$

We then focus on $m\mathsf{q} \leq \mathsf{M} \leq (m+1)\mathsf{q}$, where $m \in [\mathsf{K}-1]$. The converse bound in Theorem 1 for $m\mathsf{q} \leq \mathsf{M} \leq (m+1)\mathsf{q}$ is a straight line between $(\mathsf{M}', \mathsf{R}') = (m\mathsf{q}, \frac{(\mathsf{K}-m)\mathsf{K}}{m(\mathsf{K}-1)})$ and $(\mathsf{M}'', \mathsf{R}'') = \left((m+1)\mathsf{q}, \frac{(\mathsf{K}-m-1)\mathsf{K}}{(m+1)(\mathsf{K}-1)}\right)$.

- $m = 1$. The combined scheme can achieve $(\mathsf{M}', \mathsf{R}')$ and $(\mathsf{M}'', 4\mathsf{R}''/3)$. Hence, by memory-sharing, the multiplicative gap between the combined scheme and the converse bound is less than $4/3$.
- $m = 2$. The combined scheme can achieve $(\mathsf{M}', 4\mathsf{R}'/3)$ and $\left(\mathsf{M}'', \left(1-\frac{1}{\mathsf{K}}+\frac{1}{2}\right)\mathsf{R}''\right)$. Hence, by memory-sharing, the multiplicative gap between the combined scheme and the converse bound is less than $1-\frac{1}{\mathsf{K}}+\frac{1}{2}$.
- $m \in [3:\mathsf{K}-4]$. The combined scheme can achieve $\left(\mathsf{M}', \left(1-\frac{1}{\mathsf{K}}+\frac{1}{m-1}\right)\mathsf{R}'\right)$ and $\left(\mathsf{M}'', \left(1-\frac{1}{\mathsf{K}}+\frac{1}{m}\right)\mathsf{R}''\right)$. Hence, by memory-sharing, the multiplicative gap between the combined scheme and the converse bound is less than $1-\frac{1}{\mathsf{K}}+\frac{1}{m-1}$.
- $m = \mathsf{K}-3$. The combined scheme can achieve $\left(\mathsf{M}', \left(1-\frac{1}{\mathsf{K}}+\frac{1}{m-1}\right)\mathsf{R}'\right)$ and $(\mathsf{M}'', \mathsf{R}'')$. Hence, by memory-sharing, the multiplicative gap between the combined scheme and the converse bound is less than $1-\frac{1}{\mathsf{K}}+\frac{1}{m-1}$.
- $m \in \{\mathsf{K}-2, \mathsf{K}-1\}$. The combined scheme can achieve $(\mathsf{M}', \mathsf{R}')$ and $(\mathsf{M}'', \mathsf{R}'')$. Hence, the combined scheme coincides with the converse bound.

This concludes the proof of Theorem 6.

*On communication cost of peer-to-peer operations:* By comparing the decentralized data shuffling converse bound and the optimal centralized data shuffling load (denoted by $\mathsf{R}_{\text{Opt.Cen}}(\mathsf{M})$), we have $\mathsf{R}_{\text{Out}}(\mathsf{M})/\mathsf{R}_{\text{Opt.Cen}}(\mathsf{M}) = \mathsf{K}/(\mathsf{K}-1)$ for any $\mathsf{q} \leq \mathsf{M} \leq \mathsf{K}\mathsf{q}$. In addition, the maximum multiplicative gap between the achieved load by the combined scheme and $\mathsf{R}_{\text{Out}}(\mathsf{M})$, is $\max\left\{1-\frac{1}{\mathsf{K}}+\frac{1}{m-1}, \frac{4}{3}\right\}$, where $m \geq 3$. Hence, the maximum multiplicative gap between the achieved load by the combined scheme and $\mathsf{R}_{\text{Opt.Cen}}(\mathsf{M})$ is

$$\begin{aligned}&\frac{\mathsf{K}}{\mathsf{K}-1}\max\left\{1-\frac{1}{\mathsf{K}}+\frac{1}{m-1}, \frac{4}{3}\right\}\\ &= \max\left\{1+\frac{\mathsf{K}}{(m-1)(\mathsf{K}-1)}, \frac{4\mathsf{K}}{3(\mathsf{K}-1)}\right\},\end{aligned} \tag{79}$$

which is no more than $5/3$ if $\mathsf{K} \geq 5$. In addition, when $\mathsf{K} \leq 4$, by Corollary 2, the combined scheme is optimal such that the communication cost of peer-to-peer operations is $\mathsf{K}/(\mathsf{K}-1) \leq 2$. In general, the communication cost of peer-to-peer operations is no more than a factor of 2 as stated in Corollary 3.

In addition, with a similar proof as above to analyse each storage size regime $m\mathsf{q} \leq \mathsf{M} \leq (m+1)\mathsf{q}$ where $m \in [\mathsf{K}-1]$, we prove Corollary 3.

## VII. Conclusions

In this paper, we introduced the decentralized data shuffling problem and studied its fundamental limits. We proposed a converse bound under the constraint of uncoded storage and three achievable schemes. In general, under the constraint of uncoded storage, our schemes are order optimal to within a factor of $3/2$, and exactly optimal for small and large storage sizes, or the systems with no more than four workers.

## Appendix

We define

$$\mathcal{V}_{\mathcal{S}} := \{k \in \mathcal{S} : d_k \in \mathcal{S}\},\ \forall \mathcal{S} \subseteq [\mathsf{K}], \tag{80}$$

where $\mathcal{V}_{\mathcal{S}}$ represents the subset of workers in $\mathcal{S}$ whose demanded data units in time slot $t$, indexed by $\mathcal{A}_k^t = \mathcal{A}_{d_k}^{t-1}$ in (29), were stored by some workers in $\mathcal{S}$ at the end of time slot $t-1$[5].

In addition, we also define $Y_{\mathcal{S}}^t$ as the sub-blocks that any worker in $\mathcal{S}$ either needs to store at the end of time slot $t$ or has stored at the end of time slot $t-1$, that is,

$$\begin{aligned}Y_{\mathcal{S}}^t &:= \left\{F_i : i \in \cup_{k\in\mathcal{S}}\mathcal{A}_k^t\right\} \cup \left\{Z_k^{t-1} : k \in \mathcal{S}\right\}\\ &= \left\{F_i : i \in \cup_{k\in\mathcal{S}}(\mathcal{A}_k^t \cup \mathcal{A}_k^{t-1})\right\}\\ &\cup \left\{F_{i,\mathcal{W}} : i \notin \cup_{k\in\mathcal{S}}(\mathcal{A}_k^t \cup \mathcal{A}_k^{t-1}), \mathcal{W} \cap \mathcal{S} \neq \emptyset\right\}.\end{aligned} \tag{81}$$

With the above definitions and recalling $X_{\mathcal{S}}^t$ defined in (30) represents the messages sent by the workers in $\mathcal{S}$ during time slot $t$, we have the following lemma:

**Lemma 1** (Induction Lemma)**.** *For each non-empty set $\mathcal{S} \subseteq [\mathsf{K}]$, we have*

$$\begin{aligned}&H\left(X_{\mathcal{S}}^t | Y_{[\mathsf{K}]\backslash\mathcal{S}}^t\right)\\ &\geq \sum_{m=1}^{|\mathcal{S}|}\sum_{k\in\mathcal{V}_{\mathcal{S}}}\sum_{i\in\mathcal{A}_k^t}\sum_{\substack{\mathcal{W}\subseteq\mathcal{S}\backslash\{k\}:\\ \mathsf{u}_i^{t-1}\in\mathcal{W}, |\mathcal{W}|=m}}\frac{|F_{i,\mathcal{W}}|}{m}.\end{aligned} \tag{82}$$

Lemma 1 is the key novel contribution of our proof. The bound in (82) can be intuitively explained as follows: $H(X_{\mathcal{S}}^t | Y_{[\mathsf{K}]\backslash\mathcal{S}}^t)$ is lower bounded by the size of the requested sub-blocks by the workers in $\mathcal{V}_{\mathcal{S}}$ (instead of in $\mathcal{S}$ as in the distributed computing problem [14]) because each requested data unit by the workers in $\mathcal{S}\backslash\mathcal{V}_{\mathcal{S}}$ was requested in the previous time slot by some workers in $[\mathsf{K}] \setminus \mathcal{S}$ because of the storage constraint in (8) and the definition of $\mathcal{V}_{\mathcal{S}}$ in (80).

This lemma is proved by induction, inspired by [14].

*Proof:* Case $|\mathcal{S}| = 1$: If $\mathcal{S} = \{k\}$ where $k \in [\mathsf{K}]$, we have that $\mathcal{V}_{\{k\}} = \emptyset$ and thus the RHS of (82) is 0; thus (82) holds for $|\mathcal{S}| = 1$ because entropy is non-negative.

[5]For example, if $\mathsf{K} = 4$ and $(d_1, \ldots, d_4) = (2, 3, 4, 1)$, we have $\mathcal{V}_{\{2,3\}} = \{2\}$ because $d_2 = 3$ and thus the requested data unit by worker 2 in time slot $t$ was stored by worker 3 at the end of time slot $t-1$; similarly, we have $\mathcal{V}_{\{2,4\}} = \emptyset$ and $\mathcal{V}_{\{1,2,4\}} = \{1, 4\}$.

Case $|\mathcal{S}| \leq s$: Assume that (82) holds for all non-empty $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| \leq s$ for some integer $s \in [\mathsf{K}-1]$.

Case $|\mathcal{S}| = s+1$: Having assumed that the lemma holds for all $\mathcal{S} \subseteq [\mathsf{K}]$ where $|\mathcal{S}| \leq s$, we aim to show that for any set $\mathcal{J} \subseteq [\mathsf{K}]$ where $|\mathcal{J}| = s+1$, we have

$$H(X^t_{\mathcal{J}}|Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) \geq \sum_{m=1}^{|\mathcal{J}|} \sum_{k\in\mathcal{V}_{\mathcal{J}}} \sum_{i\in\mathcal{A}^t_k} \sum_{\substack{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k\}):\\ |\mathcal{W}|=m,\mathsf{u}_i^{t-1}\in\mathcal{W}}} \frac{|F_{i,\mathcal{W}}|}{m}. \tag{83}$$

From the independence bound on entropy we have

$$H(X^t_{\mathcal{J}}|Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) = \frac{1}{|\mathcal{J}|}\sum_{k\in\mathcal{J}}\Big(H(X^t_{\mathcal{J}\setminus\{k\}}|X^t_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) + H(X^t_k|Y^t_{[\mathsf{K}]\setminus\mathcal{J}})\Big) \tag{84a}$$

$$\geq \frac{1}{|\mathcal{J}|}\Big(\sum_{k\in\mathcal{J}} H(X^t_{\mathcal{J}\setminus\{k\}}|X^t_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) + H(X^t_{\mathcal{J}}|Y^t_{[\mathsf{K}]\setminus\mathcal{J}})\Big), \tag{84b}$$

and thus

$$(|\mathcal{J}|-1)H(X^t_{\mathcal{J}}|Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) \geq \sum_{k\in\mathcal{J}} H(X^t_{\mathcal{J}\setminus\{k\}}|X^t_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) \tag{84c}$$

$$\geq \sum_{k\in\mathcal{J}} H(X^t_{\mathcal{J}\setminus\{k\}}|X^t_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}, Z^{t-1}_k) \tag{84d}$$

$$= \sum_{k\in\mathcal{J}} H(X^t_{\mathcal{J}\setminus\{k\}}, \{F_i : i\in\mathcal{A}^t_k\}|X^t_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}, Z^{t-1}_k) \tag{84e}$$

$$= \sum_{k\in\mathcal{J}} H(\{F_i : i\in\mathcal{A}^t_k\}|Z^{t-1}_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) + \sum_{k\in\mathcal{J}} H(X^t_{\mathcal{J}\setminus\{k\}}|\{F_i : i\in\mathcal{A}^t_k\}, Z^{t-1}_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) \tag{84f}$$

$$= \sum_{k\in\mathcal{J}} H(\{F_i : i\in\mathcal{A}^t_k\}|Z^{t-1}_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) + \sum_{k\in\mathcal{J}} H\big(X^t_{\mathcal{J}\setminus\{k\}}|Y^t_{([\mathsf{K}]\setminus\mathcal{J})\cup\{k\})}\big), \tag{84g}$$

where (84d) follows because we added $Z^{t-1}_k$ in the conditioning, and conditioning cannot increase entropy, where (84e) follows because $\{F_i : i\in\mathcal{A}^t_k\}$ is a function of $(Z^{t-1}_k, X^t)$ by the decoding constraint in (5) (note that the knowledge of $(Y^t_{[\mathsf{K}]\setminus\mathcal{J}}, Z^{t-1}_k)$ implies the knowledge of $Z^{t-1}_{\{k\}\cup[\mathsf{K}]\setminus\mathcal{J}}$ and thus of $X^t_{\{k\}\cup[\mathsf{K}]\setminus\mathcal{J}}$ by the encoding constraint in (4)), where (84f) follow because $X^t_k$ is a function of $Z^{t-1}_k$ (see the encoding constraint in (4)), and (84g) from the definition in (81).

Next we bound the first term of (84g) by using the independence of the sub-blocks, and the second term of (84g) by the induction assumption. More precisely,

- First term of (84g). For each $k\in\mathcal{J}$, if $k\notin\mathcal{V}_{\mathcal{J}}$ , we have $\{F_i : i\in\mathcal{A}^t_k\} \subseteq Y^t_{[\mathsf{K}]\setminus\mathcal{J}}$. So for each $k\in\mathcal{J}$, by independence of sub-blocks, we have (85) and thus we rewrite the first term of (84g) as

$$\sum_{k\in\mathcal{J}} H(\{F_i : i\in\mathcal{A}^t_k\}|Z^{t-1}_k, Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) = \sum_{k\in\mathcal{V}_{\mathcal{J}}}\sum_{m\in[|\mathcal{J}|]}\sum_{i\in\mathcal{A}^t_k}\sum_{\substack{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k\}):\\ |\mathcal{W}|=m,\mathsf{u}_i^{t-1}\in\mathcal{W}}} |F_{i,\mathcal{W}}|. \tag{86}$$

- Second term of (84g). By the induction assumption,

$$\sum_{k\in\mathcal{J}} H\big(X^t_{\mathcal{J}\setminus\{k\}}|Y^t_{([\mathsf{K}]\setminus\mathcal{J})\cup\{k\}}\big) = \sum_{k\in\mathcal{J}} H\big(X^t_{\mathcal{J}\setminus\{k\}}|Y^t_{[\mathsf{K}]\setminus(\mathcal{J}\setminus\{k\})}\big) \geq \sum_{k\in\mathcal{J}}\sum_{u\in\mathcal{V}_{\mathcal{J}\setminus\{k\}}}\sum_{m=1}^{|\mathcal{J}|-1}\sum_{i\in\mathcal{A}^t_u}\sum_{\substack{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k,u\}):\\ |\mathcal{W}|=m,\mathsf{u}_i^{t-1}\in\mathcal{W}}} \frac{|F_{i,\mathcal{W}}|}{m}. \tag{87}$$

In order to combine (86) with (87), both terms need to have the summations in the same form. Let us focus on one worker $u'\in\mathcal{V}_{\mathcal{J}}$ and one sub-block $F_{i',\mathcal{W}'}$, where $i'\in\mathcal{A}^t_{u'}$, $\mathcal{W}'\subseteq\mathcal{J}\setminus\{u'\}$, $|\mathcal{W}'|=m$, and $\mathsf{u}^t_{i'}\in\mathcal{W}'$. On the RHS of (87), for each $k\in\mathcal{J}\setminus(\mathcal{W}'\cup\{u'\})$, it can be seen that $F_{i',\mathcal{W}'}$ appears once in the sum

$$\sum_{m\in[|\mathcal{J}|-1]}\sum_{u\in\mathcal{V}_{\mathcal{J}\setminus\{k\}}}\sum_{i\in\mathcal{A}^t_u}\sum_{\substack{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k,u\}):\\ |\mathcal{W}|=m,\mathsf{u}_i^{t-1}\in\mathcal{W}}} \frac{|F_{i,\mathcal{W}}|}{m}, \tag{88}$$

hence, the coefficient of $F_{i',\mathcal{W}'}$ on the RHS of (87) is $(|\mathcal{J}|-m-1)/m$. Thus, from (87), we have

$$\sum_{k\in\mathcal{J}} H\big(X^t_{\mathcal{J}\setminus\{k\}}|Y^t_{[\mathsf{K}]\setminus\mathcal{J})\cup\{k\}}\big) \tag{89a}$$

$$\geq \sum_{u'\in\mathcal{V}_{\mathcal{J}}}\sum_{m\in[|\mathcal{J}|-1]}\sum_{i'\in\mathcal{A}^t_{u'}}\sum_{\substack{\mathcal{W}'\subseteq(\mathcal{J}\setminus\{u'\}):\\ |\mathcal{W}'|=m,\mathsf{u}_{i'}^{t-1}\in\mathcal{W}'}} \frac{|F_{i',\mathcal{W}'}|(|\mathcal{J}|-m-1)}{m} \tag{89b}$$

$$= \sum_{u'\in\mathcal{V}_{\mathcal{J}}}\sum_{m\in[|\mathcal{J}|]}\sum_{i'\in\mathcal{A}^t_{u'}}\sum_{\substack{\mathcal{W}'\subseteq(\mathcal{J}\setminus\{u'\}):\\ |\mathcal{W}'|=m,\mathsf{u}_{i'}^{t-1}\in\mathcal{W}'}} \frac{|F_{i',\mathcal{W}'}|(|\mathcal{J}|-m-1)}{m}. \tag{89c}$$

We next take (86) and (89c) into (84g) to obtain,

$$H(X^t_{\mathcal{J}}|Y^t_{[\mathsf{K}]\setminus\mathcal{J}}) \geq \frac{1}{|\mathcal{J}|-1}\sum_{k\in\mathcal{V}_{\mathcal{J}}}\sum_{m=1}^{|\mathcal{J}|}\sum_{i\in\mathcal{A}^t_k}\sum_{\substack{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k\}):\\ |\mathcal{W}|=m,\mathsf{u}_i^{t-1}\in\mathcal{W}}} |F_{i,\mathcal{W}}| + \frac{1}{|\mathcal{J}|-1}\sum_{k\in\mathcal{V}_{\mathcal{J}}}\sum_{m=1}^{|\mathcal{J}|}\sum_{i\in\mathcal{A}^t_k}\sum_{\substack{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k\}):\\ |\mathcal{W}|=m,\mathsf{u}_i^{t-1}\in\mathcal{W}}} \frac{|F_{i,\mathcal{W}}|(|\mathcal{J}|-m-1)}{m} \tag{90a}$$

$$= \sum_{k\in\mathcal{V}_{\mathcal{J}}}\sum_{m\in[|\mathcal{J}|]}\sum_{i\in\mathcal{A}^t_k}\sum_{\substack{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k\}):\\ |\mathcal{W}|=m,\mathsf{u}_i^{t-1}\in\mathcal{W}}} \frac{|F_{i,\mathcal{W}}|}{m}, \tag{90b}$$

which proves Lemma 1. ■

$$H(\{F_i : i \in \mathcal{A}_k^t\}|Z_k^{t-1}, Y_{[\mathsf{K}]\setminus\mathcal{J}}^t) = \begin{cases} \sum_{m=1}^{|\mathcal{J}|} \sum_{i\in\mathcal{A}_k^t} \quad \sum_{\mathcal{W}\subseteq(\mathcal{J}\setminus\{k\}):|\mathcal{W}|=m, \mathsf{u}_i^{t-1}\in\mathcal{W}} |F_{i,\mathcal{W}}|, & k \in \mathcal{V}_{\mathcal{J}} \\ 0 & \text{otherwise} \end{cases} \tag{85}$$

## REFERENCES


[1] J. Chung, K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Ubershuffle: Communication-efficient data shuffling for sgd via coding theory," in *NIPS 2017, ML Systems Workshop*.

[2] M. Gurbuzbalaban, A. Ozdaglar, and P. Parrilo, "Why random reshuffling beats stochastic gradient descent," *available at arXiv:1510.08560*, Oct. 2015.

[3] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, Mar. 2018.

[4] F. A. Tobagi and V. B. Hunt, "Performance analysis of carrier sense multiple access with collision detection," *Computer Networks*, vol. 4, pp. 245–259, 1980.

[5] ANSI/IEEE Standard 802.5, "Token ring access method and physical layer specifications," *IEEE Press*, 1985.

[6] M. A. Attia and R. Tandon, "Information theoretic limits of data shuffling for distributed learning," *in IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016.

[7] ——, "On the worst-case communication overhead for distributed data shuffling," *in 54th Annual Allerton Conf. on Commun., Control, and Computing (Allerton)*, Sep. 2016.

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

[9] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," *in IEEE Infor. Theory Workshop*, Sep. 2016.

[10] Q. Yu, M. A. Maddah-Ali, and S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Infor. Theory*, vol. 64, pp. 1281 – 1296, Feb. 2018.

[11] M. A. Attia and R. Tandon, "Near optimal coded data shuffling for distributed learning," vol. 65, no. 11, pp. 7325–7349, Nov. 2019.

[12] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling for distributed learning systems," *arXiv:1807.04255, a preliminary version is in IEEE Int. Symp. Inf. Theory 2018*, Jul. 2018.

[13] M. Ji, G. Caire, and A. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 849–869, 2016.

[14] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.

[15] Y. H. Ezzeldin, M. Karmoose, and C. Fragouli, "Communication vs distributed computation: An alternative trade-off curve," *in IEEE Inf. Theory Workshop (ITW)*, Nov. 2017.

[16] N. Woolsey, R. Chen, and M. Ji, "A new combinatorial design of coded distributed computing," *in IEEE Int. Symp. Inf. Theory*, Jun. 2018.

[17] K. Konstantinos and A. Ramamoorthy, "Leveraging coding techniques for speeding up distributed computing," *available at arXiv:1802.03049*, Feb. 2018.

[18] S. R. Srinivasavaradhan, L. Song, and C. Fragouli, "Distributed computing trade-offs with random connectivity," *in IEEE Int. Symp. Inf. Theory*, June. 2018.

[19] S. Prakash, A. Reisizadeh, R. Pedarsani, and S. Avestimehr, "Coded computing for distributed graph analytics," *in IEEE Int. Symp. Inf. Theory*, June. 2018.

[20] B. Guler, A. S. Avestimehr, and A. Ortega, "A topology-aware coding framework for distributed graph processing," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.

[21] L. Songze, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2643–2654, Oct. 2017.

[22] Q. Yan, S. Yang, and M. Wigger, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," *arXiv:1806.07565*, Jun. 2018.

[23] N. Woolsey, R. Chen, and M. Ji, "Cascaded coded distributed computing on heterogeneous networks," *arXiv:1901.07670*, Jan. 2019.

[24] ——, "Coded distributed computing with heterogeneous function assignments," *arXiv:1902.10738*, Feb. 2019.

[25] P. Sadeghi, F. Arbabjolfaei, and Y. Kim, "Distributed index coding," *in IEEE Inf. Theory Workshop 2016*, Sep. 2016.

[26] Y. Liu, P. Sadeghi, F. Arbabjolfaei, and Y. Kim, "Capacity theorems for distributed index coding," *arXiv:1801.09063*, Jan. 2018.

[27] Y. Birk and T. Kol, "Informed source coding on demand (iscod) over broadcast channels," *in Proc. IEEE Conf. Comput. Commun.*, pp. 1257–1264, 1998.

[28] A. Porter and M. Wootters, "Embedded index coding," *available at arXiv:1904.02179*, Apr. 2019.

[29] C. Thapa, L. Ong, S. J. Johnson, and M. Li, "Structural characteristics of two-sender index coding," *arXiv:1711.08150v2*, Jun. 2019.

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



**Kai Wan** (S '15 – M '18) received the M.Sc. and Ph.D. degrees in Communications from Université Paris Sud-CentraleSupélec, France, in 2014 and 2018. He is currently a post-doctoral researcher with the Communications and Information Theory Chair (CommIT) at Technische Universität Berlin, Berlin, Germany. His research interests include coded caching, index coding, distributed storage, wireless communications, and distributed computing.

**Daniela Tuninetti** (M '98 – SM '13) is currently a Professor within the Department of Electrical and Computer Engineering at the University of Illinois at Chicago (UIC), which she joined in 2005. Dr. Tuninetti got her Ph.D. in Electrical Engineering in 2002 from ENST/Télécom ParisTech (Paris, France, with work done at the Eurecom Institute in Sophia Antipolis, France), and she was a postdoctoral research associate at the School of Communication and Computer Science at the Swiss Federal Institute of Technology in Lausanne (EPFL, Lausanne, Switzerland) from 2002 to 2004. Dr. Tuninetti is a recipient of a best paper award at the European Wireless Conference in 2002, of an NSF CAREER award in 2007, and named University of Illinois Scholar in 2015. Dr. Tuninetti was the editor-in-chief of the IEEE Information Theory Society Newsletter from 2006 to 2008, an editor for IEEE COMMUNICATION LETTERS from 2006 to 2009, for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2011 to 2014; and for IEEE TRANSACTIONS ON INFORMATION THEORY from 2014 to 2017. She is currently a distinguished lecturer for the Information Theory society. Dr. Tuninetti's research interests are in the ultimate performance limits of wireless interference networks (with special emphasis on cognition and user cooperation), coexistence between radar and communication systems, multi-relay networks, content-type coding, cache-aided systems and distributed private coded computing.

**Mingyue Ji** (S '09 – M '15) received the B.E. in Communication Engineering from Beijing University of Posts and Telecommunications (China), in 2006, the M.Sc. degrees in Electrical Engineering from Royal Institute of Technology (Sweden) and from University of California, Santa Cruz, in 2008 and 2010, respectively, and the PhD from the Ming Hsieh Department of Electrical Engineering at University of Southern California in 2015. He subsequently was a Staff II System Design Scientist with Broadcom Corporation (Broadcom Limited) in 2015-2016. He is now an Assistant Professor of Electrical and Computer Engineering Department and an Adjunct Assistant Professor of School of Computing at the University of Utah. He received the IEEE Communications Society Leonard G. Abraham Prize for the best IEEE JSAC paper in 2019, the best paper award in IEEE ICC 2015 conference, the best student paper award in IEEE European Wireless 2010 Conference and USC Annenberg Fellowship from 2010 to 2014. He is interested the broad area of information theory, coding theory, concentration of measure and statistics with the applications of caching networks, wireless communications, distributed computing and storage, security and privacy and (statistical) signal processing.

**Giuseppe Caire** (S '92 – M '94 – SM '03 – F '05) was born in Torino in 1965. He received the B.Sc. in Electrical Engineering from Politecnico di Torino in 1990, the M.Sc. in Electrical Engineering from Princeton University in 1992, and the Ph.D. from Politecnico di Torino in 1994. He has been a post-doctoral research fellow with the European Space Agency (ESTEC, Noordwijk, The Netherlands) in 1994-1995, Assistant Professor in Telecommunications at the Politecnico di Torino, Associate Professor at the University of Parma, Italy, Professor with the Department of Mobile Communications at the Eurecom Institute, Sophia-Antipolis, France, a Professor of Electrical Engineering with the Viterbi School of Engineering, University of Southern California, Los Angeles, and he is currently an Alexander von Humboldt Professor with the Faculty of Electrical Engineering and Computer Science at the Technical University of Berlin, Germany.

He received the Jack Neubauer Best System Paper Award from the IEEE Vehicular Technology Society in 2003, the IEEE Communications Society & Information Theory Society Joint Paper Award in 2004 and in 2011, the Leonard G. Abraham Prize for best IEEE JSAC paper in 2019, the Okawa Research Award in 2006, the Alexander von Humboldt Professorship in 2014, the Vodafone Innovation Prize in 2015, and an ERC Advanced Grant in 2018. Giuseppe Caire is a Fellow of IEEE since 2005. He has served in the Board of Governors of the IEEE Information Theory Society from 2004 to 2007, and as officer from 2008 to 2013. He was President of the IEEE Information Theory Society in 2011. His main research interests are in the field of communications theory, information theory, channel and source coding with particular focus on wireless communications.

**Pablo Piantanida** (SM '16) received both B.Sc. in Electrical Engineering and the M.Sc (with honors) from the University of Buenos Aires (Argentina) in 2003, and the Ph.D. from Université Paris-Sud (Orsay, France) in 2007. Since October 2007 he has joined the Laboratoire des Signaux et Systèmes (L2S), at CentraleSupélec together with CNRS (UMR 8506) and Université Paris-Sud, as an Associate Professor of Network Information Theory. He is currently associated with Montreal Institute for Learning Algorithms (Mila) at Université de Montréal, Quebec, Canada. He is an IEEE Senior Member and serves as Associate Editor for IEEE Transactions on Information Forensics and Security. He served as General Co-Chair of the 2019 IEEE International Symposium on Information Theory (ISIT). His research interests lie broadly in information theory and its interactions with other fields, including multi-terminal information theory, Shannon theory, machine learning and representation learning, statistical inference, cooperative communications, communication mechanisms for security and privacy.