Multi-Access Distributed Computing

Federico Brunero, *Member, IEEE*, and Petros Elia, *Member, IEEE*

***Abstract*—Coded distributed computing (CDC) is a new technique proposed with the purpose of decreasing the intense data exchange required for parallelizing distributed computing systems. Under the famous MapReduce paradigm, this coded approach has been shown to decrease this communication overhead by a factor that is linearly proportional to the overall computation load during the mapping phase. In this paper, we propose multi-access distributed computing (MADC) as a generalization of the original CDC model, where now *mappers* (nodes in charge of the map functions) and *reducers* (nodes in charge of the reduce functions) are distinct computing nodes that are connected through a multi-access network topology. Focusing on the MADC setting with combinatorial topology, which implies Λ mappers and K reducers such that there is a unique reducer connected to any α mappers, we propose a coded scheme and an information-theoretic converse, which jointly identify the optimal inter-reducer communication load, as a function of the computation load, to within a constant gap of 1.5. Additionally, a modified coded scheme and converse identify the optimal max-link communication load across all existing links to within a gap of 4.**

***Index Terms*—Coded distributed computing, coded multicasting, communication load, information-theoretic converse, MapReduce, communication complexity, multi-access distributed computing (MADC).**

I. Introduction

With the development of large-scale machine learning algorithms and applications relying heavily on large volumes of data, we are now experiencing an ever-growing need to distribute large computations across multiple computing nodes. Different computing frameworks, such as MapReduce [1] and Spark [2], have been proposed to address these needs, based on the aforementioned simple yet powerful concept of distributing large-scale algorithms — to be executed over a set of input data files — across multiple computing machines. Under the well-known MapReduce framework, the overall process is typically split in three distinct phases, starting with the *map phase*, the *shuffle phase* and then the *reduce phase*. During the map phase, each computing node is assigned a subset of the input data files, and proceeds to apply to each locally available file certain designated map functions. The outputs of such map functions, referred to as intermediate values (IVs), are then exchanged among the computing nodes during the shuffle phase, so that each computing node can retrieve any missing, required IVs it did not compute locally. Finally, during the reduce phase, each computing node computes one (or more) output functions depending on its assigned reduce functions, each of which takes as input the IVs computed for each input file.

F. Brunero was with the Communication Systems Department at EURECOM, 450 Route des Chappes, 06410 Sophia Antipolis, France. He is now with the Optical and Quantum Communications Laboratory at the Huawei Research Center in Munich, Germany (email: federico.brunero@huawei.com).

P. Elia is with the Communication Systems Department at EURECOM, 450 Route des Chappes, 06410 Sophia Antipolis, France (email: elia@eurecom.fr).

This work was supported by the European Research Council (ERC) through the EU Horizon 2020 Research and Innovation Program under Grant 725929 (Project DUALITY).

A. Coded Distributed Computing

Several studies have shown that the aforementioned distributed map-shuffle-reduce approach comes with bottlenecks that may severely hinder the parallelization of computationally-intensive operations. While some works [3], [4] focused on the impact of straggler nodes, other works have pointed out that the total execution time of a distributed computing application is often dominated by the shuffling process. For instance, the work in [5], having explored the behavior of several algorithms on the Amazon EC2 cluster, revealed that the communication load in the shuffle phase was in fact the dominant bottleneck in computing the above tasks in a distributed manner. Similarly, the authors in [6] observed that, for the execution of a conventional TeraSort application, more than $95\,\%$ of the overall execution time was spent for inter-node communication.

Motivated by this communication bottleneck in the shuffle phase, the authors in [6] introduced coded distributed computing (CDC) as a novel framework that can yield lower communication loads during data shuffling. This gain could be attributed to a careful and joint design of the map and the shuffle phases. Approaching the distributed computing problem from an information-theoretic perspective, the authors brought to light the interesting relationship between the computation load during the mapping phase, and the communication load of the shuffling step. In particular, the work in [6] revealed that if the computation load of the mapping phase is *carefully* increased by a factor r — which means that each input file is mapped on average by r *carefully chosen* computing nodes — then the communication load can be reduced by the same factor r by employing coding techniques during the shuffle phase.Building on the coding-based results in cache networks [7], [8], the work in [6] characterized the exact information-theoretic tradeoff between this computation and communication loads under any map-shuffle-reduce scheme with uniform mapping capabilities and uniform assignment of reduce functions.

Since its original information-theoretic inception in [6], coded distributed computing has been explored with several variations. Such variations include heterogeneity aspects where, for example, each computing node may be assigned different numbers of files to be mapped and functions to be reduced. For such settings, novel schemes, based on hypercube and hypercuboid geometric structures, were developed in [9], [10], which managed not only to compensate for the heterogeneous nature of the considered scenarios, but to also exploit these asymmetries in order to require a smaller number of input files, compared to the initial scheme in [6]. Regarding this problem of requiring a large number of input files, it is worth also mentioning the work in [11], where the authors proposed a system model for distributed computing, where the required number of input files was lowered dramatically under an assumption of a multi-rank wireless network.

Some additional works explored the scenario where the computing servers communicate with each other through switch networks [12] or in the presence of a randomized connectivity [13], whereas some other works further investigated distributed computing over wireless channels [14], as well as explored the interesting scenario where each computing node might have limited storage and computational resources [15]–[17]. A comprehensive survey on CDC is nicely presented in [18].

B. Contributions

In this work, we propose the multi-access distributed computing (MADC) model, which can be considered as an extension of the original setting introduced in [6], and which entails *mappers* (map nodes) being connected to various *reducers* (reduce nodes), and where these mappers and reducers are now distinct entities[1]. As is common, mappers are in charge of mapping subsets of the input files, whereas the reducers are in charge of collecting the IVs in order to compute the reduce functions. We will here focus on the so-called *combinatorial topology* which will define how the mappers are connected to the reducers. The choice of this topology stems from its analytical tractability that makes it interesting from an academic point of view, in addition to the fact that the same topology has also been recently studied in other settings outside of distributed computing [21]–[23]. Under such combinatorial topology, we consider Λ map nodes and $K \geq \Lambda$ reduce nodes, where each map node maps a subset of the input files, where each reduce node is connected to α map nodes, and where there is exactly one reducer for each subset of α map nodes. Each reducer can retrieve intermediate values only from the mappers it is connected to, whereas these same reducers can exchange via an error-free shared-link broadcast channel the remaining required intermediate values. A simple schematic of the model is shown in Fig. 1 for the case $\Lambda = 4$, $\alpha = 2$ and $K = 6$.

As discussed before, the communication load in the shuffle phase can represent a significant bottleneck of distributed computing. As a consequence, our objective is to minimize the volume of data exchanged by the reducers over the common-bus link during the shuffle phase, as well as the communication load between the mappers and the reducers. We start our analysis by first neglecting the communication cost between mappers and reducers, and we propose — for the aforementioned MADC model with combinatorial topology — a coded scheme that allows for efficient communication over the broadcast

[1]This choice is reasonable if we think of mappers as computing nodes that are specialized in evaluating the map functions, and of reducers as computing nodes that are specialized in evaluating the reduce functions [19], [20].

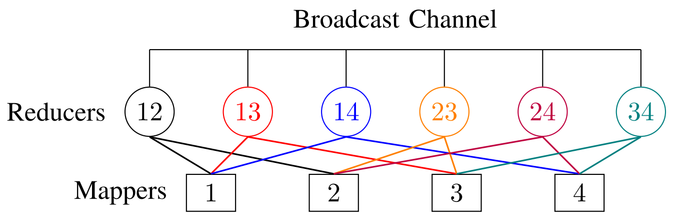

Fig. 1. Multi-access distributed computing problem with $\Lambda = 4$ mappers and $K = 6$ reducers, where each reducer is connected exactly and uniquely to a subset of $\alpha = 2$ map nodes.

communication channel. For such setting, we also provide an information-theoretic lower bound on the communication load, and we show this to be within a constant multiplicative gap of 1.5 from the achievable communication load guaranteed by the proposed coded scheme. We then proceed to also account for the download cost from mappers to reducers. For such setting, our goal is to minimize the maximal (normalized) number of bits across all links in the system. To this purpose, we introduce an additional mappers-to-reducers communication scheme and a novel converse bound which, together with the previous inter-reducer scheme, allow us to characterize the optimal max-link communication load within a constant multiplicative gap of 4. Finally, we show how the results developed to characterize both the communication load and the max-link communication load can be employed to characterize a weighted linear load metric $L_{\boldsymbol{w}} = w_1 L + w_2 J$ within a constant multiplicative factor of 2.

C. Paper Outline

The rest of the paper is organized as follows. First, the system model is presented in Section II. Next, Section III provides the main contributions of the paper. The general proofs of the achievable schemes and the converse bounds are presented from Section IV to Section VII. Finally, Section VIII concludes the paper. Some additional proofs are provided in the appendices.

D. Notation

We denote by $\mathbb{N}$ the set of non-negative integers and by $\mathbb{N}^+$ the set of positive integers. For $n \in \mathbb{N}^+$, we define $[n] := \{1, 2, \dots, n\}$. If $a, b \in \mathbb{N}^+$ such that $a < b$, then $[a : b] := \{a, a+1, \dots, b-1, b\}$. For sets we use calligraphic symbols, whereas for vectors we use bold symbols. Given a finite set $\mathcal{A}$, we denote by $|\mathcal{A}|$ its cardinality. For $n, m \in \mathbb{N}^+$, we define $[n]_m := \{\mathcal{A} : \mathcal{A} \subseteq [n], |\mathcal{A}| = m\}$. We denote by $\binom{n}{k}$ the binomial coefficient and we let $\binom{n}{k} = 0$ whenever $n < 0$, $k < 0$ or $n < k$. For $n, m \in \mathbb{N}^+$, we denote by $\mathbb{F}_{2^m}^n$ the n-dimensional vector space over the finite field with cardinality 2^m. For $n \in \mathbb{N}^+$, we denote by S_n the group of all permutations of $[n]$.

II. System Model

The general distributed computing problem [6] consists of computing Q output functions from N input files with $Q, N \in$

$\mathbb{N}^+$. Each file $w_n \in \mathbb{F}_{2^F}$ with $n \in [N]$ consists of F bits for some $F \in \mathbb{N}^+$, and the q-th function is defined as

$$\phi_q \colon \mathbb{F}_{2^F}^N \to \mathbb{F}_{2^B} \tag{1}$$

for each $q \in [Q]$, i.e., each function maps all the N input files into a stream $u_q = \phi_q(w_1, \ldots, w_N) \in \mathbb{F}_{2^B}$ of B bits. The main assumption is that each function ϕ_q is *decomposable* and so can be written as

$$\phi_q(w_1, \ldots, w_N) = h_q(g_{q,1}(w_1), \ldots, g_{q,N}(w_N)), \quad \forall q \in [Q] \tag{2}$$

where there is a map function $g_{q,n} \colon \mathbb{F}_{2^F} \to \mathbb{F}_{2^T}$ for each $n \in [N]$, which maps the input file w_n into an intermediate value (IV) $v_{q,n} = g_{q,n}(w_n) \in \mathbb{F}_{2^T}$ of T bits, and a reduce function $h_q \colon \mathbb{F}_{2^T}^N \to \mathbb{F}_{2^B}$, which maps all the IVs (one per input file) into the output value $u_q = h_q(v_{q,1}, \ldots, v_{q,N}) \in \mathbb{F}_{2^B}$ of B bits.

In this paper, we assume to have machines that are devoted to computing map functions, and machines that are devoted to computing reduce functions. Thus, a node assigned map functions is not assigned reduce functions, and vice versa. In our setting, we consider Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers, where — in accordance with the combinatorial topology of choice — there is a unique reducer connected to each subset of α mappers. Denoting by $\mathcal{U} \in [\Lambda]_\alpha$ the reducer connected to the α mappers in the set $\mathcal{U}$, before the computation begins, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ is assigned a subset $\mathcal{W}_\mathcal{U} \subseteq [Q]$ of the output functions, where here $\mathcal{W}_\mathcal{U}$ contains the indices of the functions assigned to reducer $\mathcal{U}$. For simplicity, we assume in our setting a symmetric and uniform task assignment, which implies $|\mathcal{W}_\mathcal{U}| = Q/K = \eta_2$ for some $\eta_2 \in \mathbb{N}^+$ and for each $\mathcal{U} \in [\Lambda]_\alpha$, and $\mathcal{W}_{\mathcal{U}_1} \cap \mathcal{W}_{\mathcal{U}_2} = \emptyset$ for all $\mathcal{U}_1, \mathcal{U}_2 \in [\Lambda]_\alpha$ such that $\mathcal{U}_1 \neq \mathcal{U}_2$. Afterwards, the computation is performed across the set of mappers and reducers in a distributed manner following the map-shuffle-reduce paradigm.

TABLE I
IMPORTANT PARAMETERS FOR THE MADC SYSTEM WITH COMBINATORIAL TOPOLOGY

Number of Mappers	Λ
Multi-Access Degree	α
Number of Reducers	$K = \binom{\Lambda}{\alpha}$
Computation Load	r
Number of Input Files	N
Communication Load	L
Download Cost	J
Max-Link Load	$L_{\text{max-link}} = \max(L, J)$

During the map phase, a set of files $\mathcal{M}_\lambda \subseteq \{w_1, \ldots, w_N\}$ is assigned to the mapper λ for each $\lambda \in [\Lambda]$. Each mapper $\lambda \in [\Lambda]$ computes the intermediate values $\mathcal{V}_\lambda = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\lambda\}$ for all the Q reduce functions using the files in $\mathcal{M}_\lambda$ which it has been assigned. Since reducer $\mathcal{U} \in [\Lambda]_\alpha$ is connected to the mappers in $\mathcal{U}$, it can access the intermediate values in the set $\mathcal{V}_\mathcal{U} = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\mathcal{U}\}$, where $\mathcal{M}_\mathcal{U} = \bigcup_{\lambda \in \mathcal{U}} \mathcal{M}_\lambda$ is simply the union set of files assigned to and mapped by the map nodes in $\mathcal{U}$. When the communication cost between mappers and reducers is not neglected, we can define the *download cost* as follows.

Definition 1 (Download Cost)**.** The *download cost*, denoted by J, is defined as the maximal normalized number of bits transmitted across the links from the mappers to the reducers, and is given by

$$J := \max_{\lambda \in [\Lambda]} \max_{\mathcal{U} \in [\Lambda]_\alpha : \lambda \in \mathcal{U}} \frac{R_\lambda^\mathcal{U}}{QNT} \tag{3}$$

where $R_\lambda^\mathcal{U}$ denotes the number of bits that are transmitted by mapper $\lambda \in [\Lambda]$ to reducer $\mathcal{U} \in [\Lambda]_\alpha$ where $\lambda \in \mathcal{U}$.

Remark 1. With emphasis on distributed computing scenarios where delay constraints are present, the nature of the download cost metric above allows us to bound the overall communication delay between mappers and reducers. While indeed the download cost as defined above allows for a certain parallelizing effect to remain unaccounted for, it remains a metric that properly captures a tradeoff between the computational effort at the mappers and a worst-case delivery time of data.

Assuming that each mapper computes all possible IVs from locally available files[2], we define the *computation load* as follows.

Definition 2 (Computation Load)**.** The *computation load*, denoted by r, is defined as the total number of files mapped across the Λ map nodes and normalized by the total number of files N, and it takes the form

$$r := \frac{\sum_{\lambda \in [\Lambda]} |\mathcal{M}_\lambda|}{N}. \tag{4}$$

During the shuffle phase, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ retrieves the IVs from the mappers in $\mathcal{U}$ and creates a signal $X_\mathcal{U} \in \mathbb{F}_{2^{\ell_\mathcal{U}}}$ for some $\ell_\mathcal{U} \in \mathbb{N}^+$ and for some encoding function $\psi_\mathcal{U} \colon \mathbb{F}_{2^T}^{Q|\mathcal{M}_\mathcal{U}|} \to \mathbb{F}_{2^{\ell_\mathcal{U}}}$, where $X_\mathcal{U}$ takes the form

$$X_\mathcal{U} = \psi_\mathcal{U}(\mathcal{V}_\mathcal{U}). \tag{5}$$

Then, the signal $X_\mathcal{U}$ is multicasted to all other reducers via the broadcast link which connects the reducers. Since such link is assumed to be error-free, each reducer receives all the multicast transmissions without errors. The amount of information exchanged during this phase is referred to as the *communication load*, which is formally defined in the following.

Definition 3 (Communication Load)**.** The *communication load*, denoted by L, is defined as the total number of bits transmitted by the K reducers over the broadcast channel during the shuffle phase, and — after normalization by the number of bits of all intermediate values — this load is given by

$$L := \frac{\sum_{\mathcal{U} \in [\Lambda]_\alpha} \ell_\mathcal{U}}{QNT}. \tag{6}$$

Recalling that reducer $\mathcal{U} \in [\Lambda]_\alpha$ is assigned a subset of output functions whose indices are in $\mathcal{W}_\mathcal{U}$, each reducer $\mathcal{U} \in$

[2]This means that each mapper $\lambda \in [\Lambda]$ computes the intermediate value $v_{q,n}$ for each $q \in [Q]$ and for each $w_n \in \mathcal{M}_\lambda$.

$[\Lambda]_\alpha$ wishes to recover the IVs $\{v_{q,n} : q \in \mathcal{W}_\mathcal{U}, n \in [N]\}$ to correctly compute u_q for each $q \in \mathcal{W}_\mathcal{U}$. Thus, during the reduce phase, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ reconstructs all the needed intermediate values for each $q \in \mathcal{W}_\mathcal{U}$ using the messages communicated in the shuffle phase and the intermediate values $\mathcal{V}_\mathcal{U}$ retrieved from the mappers in $\mathcal{U}$, i.e., each reducer $\mathcal{U} \in [\Lambda]_\alpha$ computes

$$(v_{q,1}, \dots, v_{q,N}) = \chi^q_\mathcal{U}(X_\mathcal{S} : \mathcal{S} \in [\Lambda]_\alpha, \mathcal{V}_\mathcal{U}) \tag{7}$$

for each $q \in \mathcal{W}_\mathcal{U}$ and for some decoding function defined as $\chi^q_\mathcal{U} \colon \prod_{\mathcal{S}\in[\Lambda]_\alpha} \mathbb{F}_{2^{\ell_\mathcal{S}}} \times \mathbb{F}_{2^T}^{Q|\mathcal{M}_\mathcal{U}|} \to \mathbb{F}_{2^T}^N$. In the end, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ computes the output function $u_q = h_q(v_{q,1}, \dots, v_{q,N})$ for each assigned $q \in \mathcal{W}_\mathcal{U}$.

When the download cost is neglected, our goal is to characterize the optimal tradeoff between computation and communication $L^\star(r)$. This optimal tradeoff is simply defined as

$$L^\star(r) \coloneqq \inf\{L : (r, L) \text{ is achievable}\} \tag{8}$$

where the tuple (r, L) is said to be *achievable* if there exists a map-shuffle-reduce procedure such that a communication load L can be guaranteed for a given computation load r. On the other hand, when we indeed jointly consider both the inter-reducer communication cost and the mapper-to-reducer download cost, then our aim will be to characterize the optimal max-link communication load $L^\star_{\text{max-link}}(r)$, which is defined as

$$L^\star_{\text{max-link}}(r) \coloneqq \inf\{L_{\text{max-link}} : (r, L_{\text{max-link}}) \text{ is achievable}\} \tag{9}$$

where $L_{\text{max-link}} \coloneqq \max(L, J)$ represents the maximum between the communication load and the download cost for a given computation load r. In simple terms, $L_{\text{max-link}}$ represents the maximal normalized number of bits flowing across any link in the considered system model, where this metric appears in various settings such as in the recent work in [12]. Notice that we will assume, throughout the paper, uniform computational capabilities across the mappers and uniform assignment of reduce functions across the reducers, as is commonly assumed (see for example the original work in [6]).

Remark 2. When $\alpha = 1$, there are $K = \Lambda$ mapper-reducer pairs. If we consider each pair to be a single computing server and in absence of download-cost considerations, the proposed system model trivially coincides with the original setting in [6]. Hence, since the results in this paper will hold for any $\alpha \in [\Lambda]$, the proposed model can in fact be considered as a proper extension of the original coded distributed computing model. In addition, the definition of the communication load in (6) is the same as in the original CDC setting in [6], and so this explains why — as it will be clarified in Remark 4 — our coded scheme coincides with the original CDC setting when both $\alpha = 1$ and the download cost is neglected.

Remark 3. If in a data center one separates the roles of computing nodes as we suggest, we might have idle times for mappers and reducers. Nevertheless, the fact that mappers are idle in the reduce phase and vice versa should be seen as an opportunity rather than as a problem. Indeed, we remind that Fig. 1 simply represents a theoretical abstraction of the considered MADC problem with combinatorial topology, and so one should recall that mappers and reducers are most likely computing nodes that are part of a bigger computing network, where each computing node simply represents a shared network resource within a data center. As a consequence, the mappers can be exploited for other purposes within the data center when they are idle, i.e., during the reduce phase, and vice versa for what concerns the reduce nodes. This consideration should further legitimize the idea of decoupling the map phase and the reduce phase, since this makes the system more modular and flexible in terms of computational efficiency.

III. Main Results

In this section we will provide our main contributions. As we have already mentioned, we will first consider a setting where the download cost is neglected. Subsequently, we will provide some additional results for the more realistic scenario where the cost of delivering data from the mappers to the reducers is non-zero.

A. Characterizing the Communication Load

The first result that we provide is the achievable computation-communication tradeoff provided by the coded scheme that will be presented in its general form in Section IV. As a reminder, download-cost considerations are not considered here, but are postponed to the next section. The result is formally stated in the following theorem.

Theorem 1 (Achievable Bound)**.** *Consider the MADC setting with combinatorial topology. Then, the optimal communication load $L^\star(r)$ is upper bounded by $L_{\text{UB}}(r)$ which is a piecewise linear curve with corner points*

$$(r, L_{\text{UB}}(r)) = \left(r, \frac{\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\binom{r+\alpha}{r} - 1\right)} \right), \quad \forall r \in [\Lambda - \alpha + 1]. \tag{10}$$

Proof. The proof of the scheme is reported in Section IV. □

Remark 4. As we already mentioned in Remark 2, if we set $\alpha = 1$ and we consider each mapper-reducer pair as a unique entity, we obtain the same system model in [6]. Interestingly, we can see that, if we specialize the result in Theorem 1 to the case $\alpha = 1$, we obtain the same computation-communication tradeoff as in [6, Theorem 1], as our topology can be considered to be a proper extension of the original CDC model.

We proceed to construct an information-theoretic converse on the communication load of the MADC setting. As it will be pointed out in the general proof in Section V, the construction of the converse takes inspiration from [10, Lemma 2] as well as from ideas in [22]. Essentially, the bound here manages to merge the approach in [10, Lemma 2], where a converse bound is built using key properties of the entropy function, with the index coding techniques in [22], where the nodes of a side information graph are iteratively selected in a proper way to systematically identify large acyclic subgraphs that are used to develop a tight converse. The result is formally stated in the following.

Theorem 2 (Converse Bound)**.** *Consider the MADC setting with combinatorial topology. Then, the optimal communication load $L^\star(r)$ is lower bounded by $L_{\mathrm{LB}}(r)$ which is a piecewise linear curve with corner points*

$$(r, L_{\mathrm{LB}}(r)) = \left(r, \frac{\binom{\Lambda}{r+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \right), \quad \forall r \in [\Lambda - \alpha + 1]. \tag{11}$$

Proof. The proof is described in Section V. □

Finally, from the results in Theorem 1 and Theorem 2, we can provide an order optimality guarantee for the MADC model. Indeed, comparing the achievable performance and the converse bound, we conclude that the two are within a constant multiplicative gap. We see this in the following theorem[3].

Theorem 3 (Order Optimality)**.** *For the MADC system with combinatorial topology, Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers for a fixed value of $\alpha \in [2 : \Lambda]$, the achievable performance in Theorem 1 is within a factor of at most 1.5 from the optimal.*

Proof. The proof is described in Appendix A. □

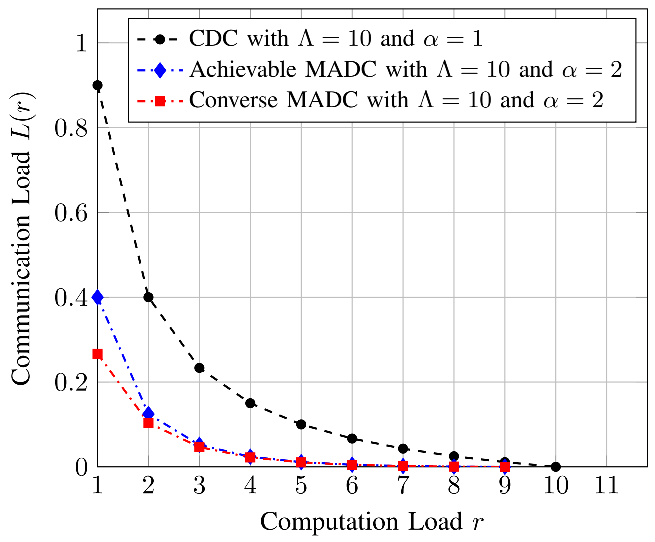

Fig. 2. Comparison between original CDC, where there are $\Lambda = 10$ pairs of mappers and reducers, and MADC with combinatorial topology, $\Lambda = 10$ mappers and $K = 45$ reducers, where each of them is uniquely associated to $\alpha = 2$ mappers.

In Fig. 2 we can see a comparison between the original CDC framework and the proposed MADC model. More specifically, for the first setting we consider $\Lambda = 10$ pairs of mappers and reducers, where each pair $\lambda \in [10]$ can be considered as a unique computing server having its own subset $\mathcal{M}_\lambda$ of assigned files. For the second setting we consider $\Lambda = 10$ mappers and $K = \binom{10}{2} = 45$ reducers, where there is a reducer connected to any $\alpha = 2$ mappers. The performance of our proposed MADC model is better than the original CDC framework and this can be attributed to the fact that each reduce has access to $\alpha = 2$ mappers, which consequently implies (intuitively) much less data to be transferred over the common-bus link.

[3]The order optimality result in Theorem 3 excludes the value $\alpha = 1$. Indeed, it can be verified that for such case the achievable performance in Theorem 1 and the converse in Theorem 2 are within a factor of at most 2. However, we already know that the coded scheme in [6] is exactly optimal when $\alpha = 1$. Hence, such value is neglected when comparing the aforementioned results.

Remark 5. As our combinatorial architecture cannot be directly compared with the original CDC model, the purpose of Fig. 2 remains mainly to offer an indication of the communicational efficiencies of the proposed model. Unfortunately, any direct comparison with other topologies with the same access degree α remains unattainable due to the lack of such topologies in the literature.

B. Characterizing the Max-Link Load

We now consider a distributed computing scenario where the download cost may be prominent and so must be accounted for. In such case, we switch to the more realistic max-link load metric, which is an alternative metric that captures both communication and download costs. The following describes the achievable max-link communication load.

Theorem 4 (Achievable Bound)**.** *Consider the MADC setting with combinatorial topology. Then, the optimal max-link communication load $L^\star_{\text{max-link}}(r)$ is upper bounded by $L_{\text{max-link,UB}}(r)$ which is given by*

$$L_{\text{max-link,UB}}(r) = \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} \frac{\tilde{a}^j_\star}{N}, \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}^j_\star}{N} \right) \tag{12}$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \dots, \tilde{a}^\Lambda_\star)$ is the optimal solution to the linear program

$$\min_{\tilde{\boldsymbol{a}}_{\mathcal{M}}} \quad \frac{1}{2}\sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{13a}$$

$$\text{subject to} \quad \tilde{a}^j_{\mathcal{M}} \geq 0, \quad \forall j \in [\Lambda] \tag{13b}$$

$$\sum_{j\in[\Lambda]} \frac{\tilde{a}^j_{\mathcal{M}}}{N} = 1 \tag{13c}$$

$$\sum_{j\in[\Lambda]} j\frac{\tilde{a}^j_{\mathcal{M}}}{N} \leq r \tag{13d}$$

and where $\tilde{\boldsymbol{a}}_{\mathcal{M}} = (\tilde{a}^1_{\mathcal{M}}, \dots, \tilde{a}^\Lambda_{\mathcal{M}})$ is the control variable.

Proof. The proof of the scheme is reported in Section VI. □

We proceed by proposing an information-theoretic converse on the max-link communication load. The result is presented in the following theorem.

Theorem 5 (Converse Bound)**.** *Consider the MADC setting with combinatorial topology. Then, the optimal max-link communication load $L^\star_{\text{max-link}}(r)$ is lower bounded by $L_{\text{max-link,LB}}(r)$ which is given by*

$$L_{\text{max-link,LB}}(r) = \frac{1}{2}\sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_\star}{N} \tag{14}$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \dots, \tilde{a}^\Lambda_\star)$ is the optimal solution to the linear program in (13).

Proof. The proof is described in Section VII. □

Finally, we can compare the results in Theorem 4 and Theorem 5 to establish the gap to optimality of the achievable performance in Theorem 4. Notice that now we do not exclude the value $\alpha = 1$ for such comparison, since for such case there is no previously known optimality result to the best of our knowledge.

Theorem 6 (Order Optimality)**.** *For the MADC system with combinatorial topology, Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers for a fixed value of $\alpha \in [\Lambda]$, the achievable performance in Theorem 4 is within a factor of at most 4 from the optimal.*

Proof. The proof is described in Appendix B. □

C. Characterizing a General Weighted Load

We devote this last part to show how the results developed to characterize the communication load $L^\star$ and the max-link load $L^\star_{\text{max-link}}$ can be employed to directly characterize also a general weighted load $L_{\boldsymbol{w}} = w_1 L + w_2 J$. The following theorems provide an achievable bound, a converse bound and an order optimality result, respectively.

Theorem 7 (Achievable Bound)**.** *Consider the MADC setting with combinatorial topology. Then, the optimal weighted load $L^\star_{\boldsymbol{w}}(r)$ is upper bounded by $L_{\boldsymbol{w},\text{UB}}(r)$ which is given by*

$$L_{\boldsymbol{w},\text{UB}}(r) = \sum_{j\in[\Lambda]} \left(w_1 \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} + w_2 \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_\star}{N} \qquad (15)$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \dots, \tilde{a}^\Lambda_\star)$ is the optimal solution to the linear program

$$\min_{\tilde{\boldsymbol{a}}_{\mathcal{M}}} \quad \sum_{j\in[\Lambda]} \left(w_1 \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + w_2 \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_{\mathcal{M}}}{N} \qquad (16a)$$

$$\text{subject to} \quad \tilde{a}^j_{\mathcal{M}} \geq 0, \quad \forall j \in [\Lambda] \qquad (16b)$$

$$\sum_{j\in[\Lambda]} \frac{\tilde{a}^j_{\mathcal{M}}}{N} = 1 \qquad (16c)$$

$$\sum_{j\in[\Lambda]} j\frac{\tilde{a}^j_{\mathcal{M}}}{N} \qquad (16d)$$

and where $\tilde{\boldsymbol{a}}_{\mathcal{M}} = (\tilde{a}^1_{\mathcal{M}}, \dots, \tilde{a}^\Lambda_{\mathcal{M}})$ is the control variable.

Proof. The proof follows from a weighted linear combination of the achievable schemes for both the communication load and the download cost described in Section VI. □

Theorem 8 (Converse Bound)**.** *Consider the MADC setting with combinatorial topology. Then, the optimal max-link communication load $L^\star_{\text{max-link}}(r)$ is lower bounded by $L_{\text{max-link,LB}}(r)$ which is given by*

$$L_{\text{max-link,LB}}(r) = \sum_{j\in[\Lambda]} \left(w_1 \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + w_2 \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_\star}{N} \qquad (17)$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \dots, \tilde{a}^\Lambda_\star)$ is the optimal solution to the linear program in (16).

Proof. The proof follows from a weighted linear combination of the converse bounds for both the communication load and the download cost already derived in Section VII. □

Theorem 9 (Order Optimality)**.** *For the MADC system with combinatorial topology, Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers for a fixed value of $\alpha \in [\Lambda]$, the achievable performance in Theorem 7 is within a factor of at most 2 from the optimal.*

Proof. The proof works along the same lines as the proof in Appendix B. □

IV. Proof of Achievable Bound in Theorem 1

We assume that there are Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers, and we assume the aforementioned combinatorial topology where each reducer is exactly and uniquely connected to α mappers. We then consider some arbitrary computation load $r \in [\Lambda - \alpha + 1]$ and we consider $Q = \eta_2 K$ output functions with $\eta_2 \in \mathbb{N}^+$, allowing us to separate the Q functions into K disjoint groups $\mathcal{W}_{\mathcal{U}}$ for each $\mathcal{U} \in [\Lambda]_\alpha$, so that each reducer is assigned η_2 functions, corresponding to $|\mathcal{W}_{\mathcal{U}}| = \eta_2$ for each $\mathcal{U} \in [\Lambda]_\alpha$.

A. Map Phase

This phase follows the same mapping strategy proposed by the authors in [6]. Hence, the input database is split in $\binom{\Lambda}{r}$ disjoint batches, each containing $\eta_1 = N/\binom{\Lambda}{r}$ files, where we assume that N is large enough such that $\eta_1 \in \mathbb{N}^+$. Consequently, we have a batch of files for each $\mathcal{T}_1 \subseteq [\Lambda]$ such that $|\mathcal{T}_1| = r$, which implies

$$\{w_1, \dots, w_N\} = \bigcup_{\mathcal{T}_1 \subseteq [\Lambda] : |\mathcal{T}_1| = r} \mathcal{B}_{\mathcal{T}_1} \qquad (18)$$

where we denote by $\mathcal{B}_{\mathcal{T}_1}$ the batch of η_1 files associated with the label $\mathcal{T}_1$. Then, mapper $\lambda \in [\Lambda]$ is assigned all batches $\mathcal{B}_{\mathcal{T}_1}$ having $\lambda \in \mathcal{T}_1$, which means that

$$\mathcal{M}_\lambda = \{\mathcal{B}_{\mathcal{T}_1} : \mathcal{T}_1 \subseteq [\Lambda], |\mathcal{T}_1| = r, \lambda \in \mathcal{T}_1\}. \qquad (19)$$

We can see that the computation load constraint is satisfied, since we have

$$\frac{\sum_{\lambda\in[\Lambda]} |\mathcal{M}_\lambda|}{N} = \frac{\Lambda \eta_1 \binom{\Lambda-1}{r-1}}{\eta_1 \binom{\Lambda}{r}} = r \qquad (20)$$

Then, each mapper computes Q intermediate values for each assigned input file, so for each $\lambda \in [\Lambda]$ we have $\mathcal{V}_\lambda = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\lambda\}$. Since then each reducer has access to α mappers, reducer $\mathcal{U} \in [\Lambda]_\alpha$ can retrieve[4] the intermediate values in $\mathcal{V}_{\mathcal{U}} = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_{\mathcal{U}}\}$ recalling that $\mathcal{M}_{\mathcal{U}} = \cup_{\lambda\in\mathcal{U}} \mathcal{M}_\lambda$. Since $|\mathcal{V}_{\mathcal{U}}| = Q\eta_1 \left(\binom{\Lambda}{r} - \binom{\Lambda-\alpha}{r}\right)$ for each $\mathcal{U} \in [\Lambda]_\alpha$, we can conclude that each computing node has access to all the intermediate values when $r \geq \Lambda - \alpha + 1$. Hence, we focus on the non-trivial regime $r \in [\Lambda - \alpha + 1]$ for any given Λ and α.

[4]Since we are presenting here the proof of the achievable bound in Theorem 1, we remind that in such case the download cost is neglected.

B. Shuffle Phase

Consider reducer $\mathcal{U} \in [\Lambda]_\alpha$. Let $\mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{U})$ with $|\mathcal{S}| = r$. First, for each $\mathcal{R} \subseteq (\mathcal{S} \cup \mathcal{U})$ such that $|\mathcal{R}| = \alpha$ and $\mathcal{R} \neq \mathcal{U}$, and for $\mathcal{T}_1 = (\mathcal{S} \cup \mathcal{U}) \setminus \mathcal{R}$, reducer $\mathcal{U}$ concatenates the intermediate values $\{v_{q,n} : q \in \mathcal{W}_\mathcal{R}, w_n \in \mathcal{B}_{\mathcal{T}_1}\}$ into the symbol $U_{\mathcal{W}_\mathcal{R},\mathcal{T}_1} = (v_{q,n} : q \in \mathcal{W}_\mathcal{R}, w_n \in \mathcal{B}_{\mathcal{T}_1}) \in \mathbb{F}_{2^{\eta_2 \eta_1 T}}$. Subsequently, such symbol is evenly split in $\left(\binom{r+\alpha}{r} - 1\right)$ segments as

$$U_{\mathcal{W}_\mathcal{R},\mathcal{T}_1} = (U_{\mathcal{W}_\mathcal{R},\mathcal{T}_1,\mathcal{T}_2} : \mathcal{T}_2 \subseteq (\mathcal{R} \cup \mathcal{T}_1), |\mathcal{T}_2| = \alpha, \mathcal{T}_2 \neq \mathcal{R}) . \tag{21}$$

Then, reducer $\mathcal{U}$ constructs the coded message

$$\bigoplus_{\mathcal{R} \subseteq (\mathcal{S} \cup \mathcal{U}) : |\mathcal{R}| = \alpha, \mathcal{R} \neq \mathcal{U}} U_{\mathcal{W}_\mathcal{R},(\mathcal{S} \cup \mathcal{U}) \setminus \mathcal{R}, \mathcal{U}} \tag{22}$$

for each $\mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{U})$ with $|\mathcal{S}| = r$, and finally concatenates all of them into the following message

$$X_\mathcal{U} = \left(\bigoplus_{\substack{\mathcal{R} \subseteq (\mathcal{S} \cup \mathcal{U}): \\ |\mathcal{R}| = \alpha, \mathcal{R} \neq \mathcal{U}}} U_{\mathcal{W}_\mathcal{R},(\mathcal{S} \cup \mathcal{U}) \setminus \mathcal{R}, \mathcal{U}} : \mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{U}), |\mathcal{S}| = r \right) \tag{23}$$

which is multicasted to all other reducers via the error-free broadcast channel.

C. Reduce Phase

Consider reducer $\mathcal{U} \in [\Lambda]_\alpha$. Since such reducer is connected to α mappers, it misses a total of $\eta_2 \eta_1 \binom{\Lambda - \alpha}{r}$ intermediate values, i.e, it misses η_2 intermediate values for each of the η_1 files in each batch that is not assigned to the mappers in $\mathcal{U}$. More precisely, reducer $\mathcal{U}$ misses the symbol $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1}$ for each $\mathcal{T}_1 \subseteq ([\Lambda] \setminus \mathcal{U})$ with $|\mathcal{T}_1| = r$. We know that during the shuffle phase such symbol is evenly split in $\left(\binom{r+\alpha}{r} - 1\right)$ segments as

$$U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1} = (U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2} : \mathcal{T}_2 \subseteq (\mathcal{U} \cup \mathcal{T}_1), |\mathcal{T}_2| = \alpha, \mathcal{T}_2 \neq \mathcal{U}) . \tag{24}$$

For each $\mathcal{T}_2 \subseteq (\mathcal{U} \cup \mathcal{T}_1)$ with $|\mathcal{T}_2| = \alpha$ and $\mathcal{T}_2 \neq \mathcal{U}$, we can verify that reducer $\mathcal{U}$ can decode $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2}$ from $X_{\mathcal{T}_2}$. Indeed, there exists an $\mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{T}_2)$ with $|\mathcal{S}| = r$ such that $\mathcal{S} = (\mathcal{U} \cup \mathcal{T}_1) \setminus \mathcal{T}_2$. For such $\mathcal{S}$, the corresponding coded message in $X_{\mathcal{T}_2}$ is

$$\bigoplus_{\mathcal{R} \subseteq (\mathcal{S} \cup \mathcal{T}_2) : |\mathcal{R}| = \alpha, \mathcal{R} \neq \mathcal{T}_2} U_{\mathcal{W}_\mathcal{R},(\mathcal{S} \cup \mathcal{T}_2) \setminus \mathcal{R}, \mathcal{T}_2} =$$

$$= \bigoplus_{\mathcal{R} \subseteq (\mathcal{U} \cup \mathcal{T}_1) : |\mathcal{R}| = \alpha, \mathcal{R} \neq \mathcal{T}_2} U_{\mathcal{W}_\mathcal{R},(\mathcal{U} \cup \mathcal{T}_1) \setminus \mathcal{R}, \mathcal{T}_2} \tag{25}$$

$$= U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2} \oplus \underbrace{\bigoplus_{\substack{\mathcal{R} \subseteq (\mathcal{U} \cup \mathcal{T}_1): \\ |\mathcal{R}| = \alpha, \mathcal{R} \neq \mathcal{T}_2, \mathcal{R} \neq \mathcal{U}}} U_{\mathcal{W}_\mathcal{R},(\mathcal{U} \cup \mathcal{T}_1) \setminus \mathcal{R}, \mathcal{T}_2}}_{\text{interference}} . \tag{26}$$

Notice that reducer $\mathcal{U}$ can cancel the interference term by using the intermediate values retrieved from mappers in $\mathcal{U}$, so it can correctly decode $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2}$. By following the same rationale for each $\mathcal{T}_2 \subseteq (\mathcal{U} \cup \mathcal{T}_1)$ with $|\mathcal{T}_2| = \alpha$ and $\mathcal{T}_2 \neq \mathcal{U}$, we can conclude that reducer $\mathcal{U}$ can correctly recover $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1}$ and can do so for each $\mathcal{T}_1 \subseteq ([\Lambda] \setminus \mathcal{U})$, completely recovering all the $\eta_2 \eta_1 \binom{\Lambda - \alpha}{r}$ missing intermediate values. The same holds for any other $\mathcal{U} \in [\Lambda]_\alpha$, and so we can conclude that each reducer is able to recover from the multicast messages of other reducers all the missing intermediate values.

D. Communication Load

The communication load guaranteed by the coded scheme described above is given by

$$L_{\text{UB}}(r) = \frac{\sum_{\mathcal{U} \in [\Lambda]_\alpha} |X_\mathcal{U}|}{QNT} \tag{27}$$

$$= \frac{\binom{\Lambda}{\alpha} \eta_2 \eta_1 \binom{\Lambda - \alpha}{r} T / \left(\binom{r+\alpha}{r} - 1\right)}{Q \eta_1 \binom{\Lambda}{r} T} \tag{28}$$

$$= \frac{\binom{\Lambda - \alpha}{r}}{\binom{\Lambda}{r} \left(\binom{r+\alpha}{r} - 1\right)} \tag{29}$$

for each $r \in [\Lambda - \alpha + 1]$. Notice that the lower convex envelope of the achievable points $\{(r, L_{\text{LB}}(r)) : r \in [\Lambda - \alpha + 1]\}$ is achievable by adopting the memory-sharing strategy presented in [6]. The proof is concluded. $\square$

V. Proof of Converse Bound in Theorem 2

A. Preliminaries

We begin the proof by introducing some useful notation. For $q \in [Q]$ and $n \in [N]$, we let $V_{q,n}$ be an i.i.d. random variable and we let $v_{q,n}$ be the realization of $V_{q,n}$. Then, we define

$$D_\mathcal{U} := \{V_{q,n} : q \in \mathcal{W}_\mathcal{U}, n \in [N]\} \tag{30}$$

$$C_\mathcal{U} := \{V_{q,n} : q \in [Q], w_n \in \mathcal{M}_\mathcal{U}\} \tag{31}$$

$$Y_\mathcal{U} := (D_\mathcal{U}, C_\mathcal{U}). \tag{32}$$

Recalling that we denote by $X_\mathcal{U}$ the multicast message transmitted by reducer $\mathcal{U} \in [\Lambda]_\alpha$, the equation

$$H(X_\mathcal{U} \mid C_\mathcal{U}) = 0 \tag{33}$$

holds, since $X_\mathcal{U}$ is a function of the intermediate values retrieved by reducer $\mathcal{U}$. Moreover, for any map-shuffle-reduce scheme, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ has to be able to correctly recover all the intermediate values $D_\mathcal{U}$ given the transmissions of all reducers $X_{[\Lambda]_\alpha} := (X_\mathcal{U} : \mathcal{U} \in [\Lambda]_\alpha)$ and given the IVs $C_\mathcal{U}$ computed by the mappers in $\mathcal{U}$. Thus, the equation

$$H(D_\mathcal{U} \mid X_{[\Lambda]_\alpha}, C_\mathcal{U}) = 0 \tag{34}$$

holds for each $\mathcal{U} \in [\Lambda]_\alpha$.

B. Developing the Lower Bound

For a given file assignment $\mathcal{M} := (\mathcal{M}_1, \ldots, \mathcal{M}_\Lambda)$, we let $L_\mathcal{M}$ be the corresponding communication load under this assignment $\mathcal{M}$. Then, we provide a lower bound on $L_\mathcal{M}$ for any given file assignment in the following lemma.

Lemma 1. *Consider a specific file assignment $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_\Lambda)$. Let $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ be a permutation of the set $[\Lambda]$ and define*

$$\mathcal{D}_i := (D_{\mathcal{U}^i} : \mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}, |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i) \tag{35}$$

$$\mathcal{C}_i := (C_{\mathcal{U}^i} : \mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}, |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i) \tag{36}$$

$$\mathcal{Y}_{i-1} := (Y_{\mathcal{U}^j} : j \in [\alpha : i-1], \mathcal{U}^j \subseteq \{c_1, \ldots, c_j\}, |\mathcal{U}^j| = \alpha, c_j \in \mathcal{U}^j) \tag{37}$$

for each $i \in [\alpha : \Lambda]$. Then, the communication load is lower bounded by

$$L_{\mathcal{M}} \geq \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}). \tag{38}$$

Proof. The proof is described in Appendix C. □

Now, we proceed with the proof. Denote by $\tilde{a}^{\mathcal{T}}$ the number of files which are mapped exclusively by the mappers in $\mathcal{T}$ for some $\mathcal{T} \subseteq [\Lambda]$. As each reducer $\mathcal{U} \in [\Lambda]_\alpha$ does not have access to the intermediate values of all those files that are not mapped by the mappers in $\mathcal{U}$, the term $\tilde{a}^{\mathcal{T}}$ represents the number of files whose intermediate values are required by each reducer $\mathcal{U} \in [\Lambda]_\alpha$ that does not have access to the mappers in $\mathcal{T}$, i.e., each reducer $\mathcal{U} \in [\Lambda]_\alpha$ such that $\mathcal{U} \cap \mathcal{T} = 0$ or, equivalently, each reducer $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ such that $|\mathcal{U}| = \alpha$. Taking advantage of the independence of the intermediate values and recalling that each reducer computes η_2 disjoint output functions, from Lemma 1 and for a given permutation $\boldsymbol{c} = (c_1, \dots, c_\Lambda)$ of the set $[\Lambda]$, we can further write

$$L_{\mathcal{M}} \geq \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{39}$$

$$= \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i \subseteq \{c_1,\dots,c_i\}: \\ |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i}} H(D_{\mathcal{U}^i} \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{40}$$

$$= \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i \subseteq \{c_1,\dots,c_i\}: \\ |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i}} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1,\dots,c_i\}} \tilde{a}^{\mathcal{T}} \eta_2 T \tag{41}$$

$$= \frac{1}{KN} \sum_{i \in [\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i \subseteq \{c_1,\dots,c_i\}: \\ |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i}} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1,\dots,c_i\}} \tilde{a}^{\mathcal{T}}. \tag{42}$$

If we build a bound as the one in Lemma 1 for each permutation of the set $[\Lambda]$ and we sum up all these bounds together, we obtain the expression

$$L_{\mathcal{M}} \geq \frac{1}{KN\Lambda!} \sum_{\boldsymbol{c} \in S_\Lambda} \sum_{i \in [\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i \subseteq \{c_1,\dots,c_i\}: \\ |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i}} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1,\dots,c_i\}} \tilde{a}^{\mathcal{T}} \tag{43}$$

where we recall that S_Λ represents the group of all permutations of $[\Lambda]$. Our goal now is to simplify this expression and we start doing so by counting how many times each term $\tilde{a}^{\mathcal{T}}$ appears in the RHS of (43) for any fixed $\mathcal{T} \subseteq [\Lambda]$ with $|\mathcal{T}| = j$ and $j \in [\Lambda]$.

First, we focus on some reducer $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$. We can see that $\tilde{a}^{\mathcal{T}}$ appears in the RHS of (43) for all those permutations in S_Λ for which $\mathcal{U} = \mathcal{U}^i$ for some $i \in [\alpha : \Lambda]$ such that $\mathcal{U}^i \subseteq \{c_1, \dots, c_i\}$ with $|\mathcal{U}^i| = \alpha$ and $c_i \in \mathcal{U}^i$, and such that $\mathcal{T} \subseteq ([\Lambda] \setminus \{c_1, \dots, c_i\})$ with $|\mathcal{T}| = j$. Denoting by $\mathcal{P}_{\mathcal{U},\mathcal{T}}$ the set of such permutations, we can see that

$$|\mathcal{P}_{\mathcal{U},\mathcal{T}}| = \alpha! j! (\Lambda - \alpha - j)! \binom{\Lambda}{\alpha + j} \tag{44}$$

where $|\mathcal{P}_{\mathcal{U},\mathcal{T}}|$ represents the number of permutation vectors in S_Λ for which the elements in $\mathcal{U}$ appear before the elements in $\mathcal{T}$ in the permutation vector $\boldsymbol{c} \in S_\Lambda$. The same reasoning applies to any reducer $\mathcal{U} \in [\Lambda]_\alpha$ for which $\mathcal{U} \cap \mathcal{T} = \emptyset$. As a consequence, the term $\tilde{a}^{\mathcal{T}}$ appears in the RHS of (43) a total of

$$\sum_{\mathcal{U} \in [\Lambda]_\alpha : \mathcal{U} \cap \mathcal{T} = \emptyset} |\mathcal{P}_{\mathcal{U},\mathcal{T}}| = \binom{\Lambda - j}{\alpha} \alpha! j! (\Lambda - \alpha - j)! \binom{\Lambda}{\alpha + j} \tag{45}$$

times. The same rationale holds for any $\tilde{a}^{\mathcal{T}}$ where $\mathcal{T} \subseteq [\Lambda]$ and $|\mathcal{T}| = j$ with $j \in [\Lambda]$. Consequently, we can rewrite the expression in (43) as

$$L_{\mathcal{M}} \geq \frac{1}{KN\Lambda!} \sum_{\boldsymbol{c} \in S_\Lambda} \sum_{i \in [\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i \subseteq \{c_1,\dots,c_i\}: \\ |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i}} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1,\dots,c_i\}} \tilde{a}^{\mathcal{T}} \tag{46}$$

$$= \frac{1}{KN\Lambda!} \sum_{j \in [\Lambda]} \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \Bigg(\binom{\Lambda - j}{\alpha} \alpha! j! \times (\Lambda - \alpha - j)! \binom{\Lambda}{\alpha + j} \tilde{a}^{\mathcal{T}} \Bigg) \tag{47}$$

$$= \frac{1}{KN} \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{j}} \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \tilde{a}^{\mathcal{T}} \tag{48}$$

$$= \frac{1}{K} \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{j}} \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{49}$$

where $\tilde{a}^j_{\mathcal{M}} \coloneqq \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \tilde{a}^{\mathcal{T}}$ is defined as the total number of files which are mapped by exactly j map nodes under this particular file assignment $\mathcal{M}$.

For any given file assignment $\mathcal{M}$ and for any given computation load $r \in [K]$, the fact that $|\mathcal{M}_1| + \dots + |\mathcal{M}_\Lambda| \leq rN$ also implies that $\tilde{a}^j_{\mathcal{M}} \geq 0$ for each $j \in [\Lambda]$, as well as implies that $\sum_{j \in [\Lambda]} \tilde{a}^j_{\mathcal{M}} = N$ and that $\sum_{j \in [\Lambda]} j \tilde{a}^j_{\mathcal{M}} \leq rN$. Thus, we can further lower bound the above using Jensen's inequality and the fact that $\binom{\Lambda}{\alpha+j} / \binom{\Lambda}{j}$ is convex and decreasing[5] in j. Hence, we can write

$$L_{\mathcal{M}} \geq \frac{1}{K} \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{j}} \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{50}$$

$$\geq \frac{1}{K} \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{r}} \tag{51}$$

$$= \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{\alpha} \binom{\Lambda}{r}} \tag{52}$$

where (51) holds due to $\sum_{j \in [\Lambda]} \tilde{a}^j_{\mathcal{M}} = N$ and the computation constraint $\sum_{j \in [\Lambda]} j \tilde{a}^j_{\mathcal{M}} \leq rN$.

Given that (52) is independent of the file assignment $\mathcal{M}$ and lower bounds $L_{\mathcal{M}}$ for any $\mathcal{M}$ such that $|\mathcal{M}_1| + \dots + |\mathcal{M}_\Lambda| \leq rN$, we can further write

$$L^\star(r) \geq \inf_{\mathcal{M} : |\mathcal{M}_1| + \dots + |\mathcal{M}_\Lambda| \leq rN} L_{\mathcal{M}} \tag{53}$$

[5]This was already proved in the proof of [22, Lemma 3] by writing down each combinatorial coefficient in $\binom{\Lambda}{\alpha+j} / \binom{\Lambda}{j}$ as a finite product and using then the general Leibniz rule to show that its second derivative is non-negative.

$$\geq \inf_{\mathcal{M}:|\mathcal{M}_1|+\cdots+|\mathcal{M}_\Lambda|\leq rN} \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \tag{54}$$

$$= \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \tag{55}$$

$$= L_{\text{LB}}(r). \tag{56}$$

Notice that the bound $L_{\text{LB}}(r)$ can be extended to include also the non-integer values of r as described in [6]. This concludes the proof. □

VI. Proof of Achievable Bound in Theorem 4

As we mentioned in the statement of Theorem 4, the coded scheme depends on the solution of the linear program in (13). Hence, the first step is to evaluate the optimal solution[6] $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \dots, \tilde{a}^\Lambda_\star)$. Next, we partition the input database in Λ parts, where we denote by $\mathcal{L}_j$ the j-th part, which contains $|\mathcal{L}_j| = \tilde{a}^j_\star$ files for each $j \in [\Lambda]$. Then, each part $j \in [\Lambda]$ of the database is split in $\binom{\Lambda}{j}$ batches containing η_j files each for some $\eta_j \in \mathbb{N}$, so that $\tilde{a}^j_\star = \eta_j \binom{\Lambda}{j}$ for each $j \in [\Lambda]$. This implies

$$\{w_1, \dots, w_N\} = \bigcup_{j\in[\Lambda]} \mathcal{L}_j \tag{57}$$

$$= \bigcup_{j\in[\Lambda]} \bigcup_{\mathcal{T}_1\subseteq[\Lambda]:|\mathcal{T}_1|=j} \mathcal{B}_{j,\mathcal{T}_1} \tag{58}$$

where we denote by $\mathcal{B}_{j,\mathcal{T}_1}$ the batch containing η_j files associated with the label $\mathcal{T}_1$. Then, mapper $\lambda \in [\Lambda]$ is assigned all batches $\mathcal{B}_{j,\mathcal{T}_1}$ having $\lambda \in \mathcal{T}_1$ for each $j \in [\Lambda]$, which implies

$$\mathcal{M}_\lambda = \{\mathcal{B}_{j,\mathcal{T}_1} : j \in [\Lambda], \mathcal{T}_1 \subseteq [\Lambda], |\mathcal{T}_1| = j, \lambda \in \mathcal{T}_1\}. \tag{59}$$

The computation load constraint is satisfied, since we have

$$\frac{\sum_{\lambda\in[\Lambda]} |\mathcal{M}_\lambda|}{N} = \frac{\Lambda \sum_{j\in[\Lambda]} \eta_j \binom{\Lambda-1}{j-1}}{N} \tag{60}$$

$$= \frac{\sum_{j\in[\Lambda]} j\eta_j \binom{\Lambda}{j}}{N} \tag{61}$$

$$= \frac{\sum_{j\in[\Lambda]} j\tilde{a}^j_\star}{N} \leq r \tag{62}$$

where the last inequality holds under the constraint in (13d).

Our goal is to provide an achievable scheme for the max-link communication load. Recalling that we denote by L and J the communication load and the download cost, respectively, we will have

$$L^\star_{\text{max-link}}(r) \leq L_{\text{max-link,UB}}(r) = \max(L, D). \tag{63}$$

A. Communication Load

For what concerns the communication load, we can take advantage of the achievable scheme described in Section IV. Simply, the scheme in Section IV is applied Λ times, one time per partition $\mathcal{L}_j$ which is considered as an independent input database. If we denote by L_j the communication load when we focus on the part $\mathcal{L}_j$, we have that L_j is given by

$$L_j = \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j} - 1\right)} \frac{\tilde{a}^j_\star}{N} \tag{64}$$

for each $j \in [\Lambda]$. Hence, the overall communication load L is given by

$$L = \sum_{j\in[\Lambda]} L_j = \sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j} - 1\right)} \frac{\tilde{a}^j_\star}{N}. \tag{65}$$

B. Download Cost

We remind that the download cost is defined as

$$J = \max_{\lambda\in[\Lambda]} \max_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} \frac{R^{\mathcal{U}}_\lambda}{QNT} \tag{66}$$

where $R^{\mathcal{U}}_\lambda$ represents the number of bits which are sent from mapper λ to reducer $\mathcal{U}$. This quantity is minimized if the number of bits transmitted over each link connecting a mapper to a reducer is the same. This can be accomplished as follows.

Consider a reducer $\mathcal{U} \in [\Lambda]_\alpha$ and a mapper $\lambda \in \mathcal{U}$. According to the file assignment above, mapper λ computes the IVs in the set $\mathcal{V}_\lambda = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\lambda\}$. The set $\mathcal{V}_\lambda$ can equivalently be written as follows

$$\mathcal{V}_\lambda = \{\mathcal{V}_{\lambda,\mathcal{S}} : i \in [\alpha], \mathcal{S} \subseteq (\mathcal{U} \setminus \{\lambda\}), |\mathcal{S}| = i - 1\} \tag{67}$$

where $\mathcal{V}_{\lambda,\mathcal{S}}$ is defined as

$$\mathcal{V}_{\lambda,\mathcal{S}} := \left\{v_{q,n} : q \in [Q], w_n \in \mathcal{M}^{\lambda\cup\mathcal{S}}\right\} \tag{68}$$

and where $\mathcal{M}^{\lambda\cup\mathcal{S}} := \bigcap_{s\in(\lambda\cup\mathcal{S})} \mathcal{M}_s$. This simply says that the set $\mathcal{V}_{\lambda,\mathcal{S}}$ contains the IVs which are mapped by mapper λ and the $(i-1)$ mappers in $\mathcal{S}$. Hence, if we evenly split $\mathcal{V}_{\lambda,\mathcal{S}}$ in i segments as follows

$$\mathcal{V}_{\lambda,\mathcal{S}} = (\mathcal{V}_{\lambda,\mathcal{S},s} : s \in (\lambda \cup \mathcal{S})) \tag{69}$$

we simply let mapper λ send $\mathcal{V}_{\lambda,\mathcal{S},\lambda}$. This implies

$$R^{\mathcal{U}}_\lambda = \sum_{i\in[\alpha]} \sum_{\mathcal{S}\subseteq(\mathcal{U}\setminus\{\lambda\}):|\mathcal{S}|=i-1} |\mathcal{V}_{\lambda,\mathcal{S},\lambda}| \tag{70}$$

$$= \sum_{i\in[\alpha]} \sum_{\mathcal{S}\subseteq(\mathcal{U}\setminus\{\lambda\}):|\mathcal{S}|=i-1} \frac{|\mathcal{V}_{\lambda,\mathcal{S}}|}{i} \tag{71}$$

$$= \sum_{i\in[\alpha]} \sum_{\mathcal{S}\subseteq(\mathcal{U}\setminus\{\lambda\}):|\mathcal{S}|=i-1} \sum_{j\in[\Lambda]} \frac{\eta_j \binom{\Lambda-\alpha}{j-i} QT}{i} \tag{72}$$

$$= \sum_{j\in[\Lambda]} \sum_{i\in[\alpha]} \frac{\eta_j \binom{\Lambda-\alpha}{j-i} QT}{i} \binom{\alpha-1}{i-1} \tag{73}$$

for each $\lambda \in [\Lambda]$ and $\mathcal{U} \in [\Lambda]_\alpha$ with $\lambda \in \mathcal{U}$. Hence, we can further write

$$J = \max_{\lambda\in[\Lambda]} \max_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} \frac{R^{\mathcal{U}}_\lambda}{QNT} \tag{74}$$

$$= \frac{1}{QNT} \sum_{j\in[\Lambda]} \sum_{i\in[\alpha]} \frac{\eta_j \binom{\Lambda-\alpha}{j-i} QT}{i} \binom{\alpha-1}{i-1} \tag{75}$$

[6]The linear program in (13) is not infeasible nor unbounded. Hence, it admits an optimal solution. Nevertheless, we cannot easily find an analytical solution to the optimization problem as the coefficients in the sum over j are not convex.

$$= \sum_{j\in[\Lambda]} \sum_{i\in[\alpha]} \frac{\binom{\Lambda-\alpha}{j-i}\binom{\alpha-1}{i-1}}{i\binom{\Lambda}{j}} \frac{\tilde{a}_\star^j}{N} \tag{76}$$

recalling that $\tilde{a}_\star^j = \eta_j\binom{\Lambda}{j}$ for each $j \in [\Lambda]$. Further, the following lemma holds.

Lemma 2. *For any non-negative integers Λ, α and j, we have*

$$\sum_{i\in[\alpha]} \frac{\binom{\Lambda-\alpha}{j-i}\binom{\alpha-1}{i-1}}{i\binom{\Lambda}{j}} = \frac{\binom{\Lambda}{j} - \binom{\Lambda-\alpha}{j}}{\alpha\binom{\Lambda}{j}}. \tag{77}$$

Proof. The proof is described in Appendix D. □

As a consequence, the download cost J is equivalently given by

$$J = \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{j} - \binom{\Lambda-\alpha}{j}}{\alpha\binom{\Lambda}{j}} \frac{\tilde{a}_\star^j}{N} \tag{78}$$

$$= \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_\star^j}{N}. \tag{79}$$

C. Max-Link Communication Load

Since we have now the expressions for both L and J, we can write explicitly the achievable max-link communication load as follows

$$L_{\text{max-link,UB}}(r) = \max(L, D) \tag{80}$$

$$= \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} \frac{\tilde{a}_\star^j}{N}, \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_\star^j}{N} \right). \tag{81}$$

The expression above coincides with the achievable expression in Theorem 4. The proof is concluded. □

VII. Proof of Converse Bound in Theorem 5

We quickly recall that $L_{\mathcal{M}}$ denotes the communication load under the file assignment $\mathcal{M} = (\mathcal{M}_1, \dots, \mathcal{M}_\Lambda)$ and that we know

$$L_{\mathcal{M}} \geq \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^j}{N} \tag{82}$$

from Section V. Then, after denoting by $J_{\mathcal{M}}$ and by $L_{\text{max-link},\mathcal{M}}(r) = \max(L_{\mathcal{M}}, J_{\mathcal{M}})$ the download cost and the max-link communication load, respectively, under file assignment $\mathcal{M}$, we can write

$$L_{\text{max-link},\mathcal{M}}(r) = \max(L_{\mathcal{M}}, J_{\mathcal{M}}) \tag{83}$$

$$\geq \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^j}{N}, J_{\mathcal{M}} \right). \tag{84}$$

To develop a lower bound on $J_{\mathcal{M}}$, we start from the definition of the download cost, so that we have

$$J_{\mathcal{M}} = \max_{\lambda\in[\Lambda]} \max_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} \frac{R_\lambda^{\mathcal{U}}}{QNT} \tag{85a}$$

$$\geq \max_{\lambda\in[\Lambda]} \frac{1}{\binom{\Lambda-1}{\alpha-1}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} R_\lambda^{\mathcal{U}} \tag{85b}$$

$$\geq \frac{1}{\Lambda\binom{\Lambda-1}{\alpha-1}QNT} \sum_{\lambda\in[\Lambda]} \sum_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} R_\lambda^{\mathcal{U}} \tag{85c}$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha} \sum_{\lambda\in\mathcal{U}} R_\lambda^{\mathcal{U}} \tag{85d}$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha} R^{\mathcal{U}} \tag{85e}$$

where $R^{\mathcal{U}}$ is defined as

$$R^{\mathcal{U}} := \sum_{\lambda\in\mathcal{U}} R_\lambda^{\mathcal{U}} \tag{86}$$

to represent the overall number of bits received by reducer $\mathcal{U} \in [\Lambda]_\alpha$. Now, since each reducer $\mathcal{U}$ is expected to receive all the IVs mapped by the mappers in $\mathcal{U}$, we have

$$R^{\mathcal{U}} \geq H(C_{\mathcal{U}}) \tag{87}$$

where we recall that $C_{\mathcal{U}} = \{V_{q,n} : q \in [Q], w_n \in \mathcal{M}_{\mathcal{U}}\}$ from Section V. Hence, we can write

$$J_{\mathcal{M}} \geq \frac{1}{\alpha\binom{\Lambda}{\alpha}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha} R^{\mathcal{U}} \tag{88a}$$

$$\geq \frac{1}{\alpha\binom{\Lambda}{\alpha}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha} H(C_{\mathcal{U}}) \tag{88b}$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha} \sum_{\mathcal{T}\subseteq[\Lambda]:\mathcal{T}\cap\mathcal{U}\neq\emptyset} \tilde{a}^{\mathcal{T}} QT \tag{88c}$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha}N} \sum_{j\in[\Lambda]} \sum_{\mathcal{T}\subseteq[\Lambda]:|\mathcal{T}|=j} \left(\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}\right) \tilde{a}^{\mathcal{T}} \tag{88d}$$

$$= \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_{\mathcal{M}}^j}{N} \tag{88e}$$

recalling that $\tilde{a}_{\mathcal{M}}^j = \sum_{\mathcal{T}\subseteq[\Lambda]:|\mathcal{T}|=j} \tilde{a}^{\mathcal{T}}$. To conclude, for a given file assignment $\mathcal{M}$, the max-link communication load is lower bounded as

$$L_{\text{max-link},\mathcal{M}}(r) \geq \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^j}{N}, J_{\mathcal{M}} \right) \tag{89}$$

$$\geq \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^j}{N}, \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_{\mathcal{M}}^j}{N} \right) \tag{90}$$

$$\geq \frac{1}{2} \sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}_{\mathcal{M}}^j}{N}. \tag{91}$$

Since each file assignment $\mathcal{M}$ such that $|\mathcal{M}_1|+\dots+|\mathcal{M}_\Lambda| \leq rN$ also implies that $\tilde{a}_{\mathcal{M}}^j \geq 0$ for each $j \in [\Lambda]$, as well as implies that $\sum_{j\in[\Lambda]} \tilde{a}_{\mathcal{M}}^j = N$ and that $\sum_{j\in[\Lambda]} j\tilde{a}_{\mathcal{M}}^j \leq rN$, the max-link load $L^\star_{\text{max-link}}(r)$ is lower bounded by the solution to the following linear program

$$\min_{\tilde{\boldsymbol{a}}_{\mathcal{M}}} \quad \frac{1}{2}\sum_{j\in[\Lambda]}\left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}}+\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}\right)\frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{92a}$$

$$\text{subject to} \quad \tilde{a}^j_{\mathcal{M}} \geq 0, \quad \forall j \in [\Lambda] \tag{92b}$$

$$\sum_{j\in[\Lambda]}\frac{\tilde{a}^j_{\mathcal{M}}}{N}=1 \tag{92c}$$

$$\sum_{j\in[\Lambda]} j\frac{\tilde{a}^j_{\mathcal{M}}}{N}\leq r \tag{92d}$$

where $\tilde{\boldsymbol{a}}_{\mathcal{M}}=(\tilde{a}^1_{\mathcal{M}},\dots,\tilde{a}^\Lambda_{\mathcal{M}})$ is the control variable. The proof is concluded. □

VIII. Conclusions

In this work, we introduced multi-access distributed computing, focusing on the MADC model with combinatorial topology, which implies Λ mappers and $K=\binom{\Lambda}{\alpha}$ reducers, so that there is a reducer for any set of α mappers. Neglecting at first the download cost from mappers to reducers and so focusing only on the inter-reducer communication load, we proposed a novel scheme which, together with an information-theoretic converse, characterizes the optimal communication load within a constant multiplicative gap of 1.5. Subsequently, we jointly considered the setting which keeps into account the download cost and for such scenario we characterized the optimal max-link communication load within a multiplicative factor of 4. We point out that the here introduced achievable shuffling scheme generalizes the original coded scheme in [6] (corresponding to the case $\alpha=1$).

Interesting future directions could include the study of the here proposed MADC setting when mappers and reducers have heterogeneous computational resources. A careful study of other multi-access network topologies is also another challenging research direction. Reflecting a design freedom, the search for the best possible topology, for a given computation load, remains a very pertinent open problem in distributed computing.

Appendix A
Proof of Order Optimality in Theorem 3

To prove the order optimality result in Theorem 3, we need to upper bound the ratio $L_{\text{UB}}(r)/L^\star(r)$ for each $r\in[\Lambda-\alpha+1]$. We start by noting that the following

$$\frac{L_{\text{UB}}(r)}{L^\star(r)} \leq \frac{L_{\text{UB}}(r)}{L_{\text{LB}}(r)} \tag{93}$$

$$= \frac{\binom{\Lambda-\alpha}{r}}{\cancel{\binom{\Lambda}{r}}\left(\binom{r+\alpha}{r}-1\right)}\frac{\cancel{\binom{\Lambda}{r}}\binom{\Lambda}{\alpha}}{\binom{\Lambda}{r+\alpha}} \tag{94}$$

$$= \frac{\binom{\Lambda-\alpha}{r}}{\left(\binom{r+\alpha}{r}-1\right)}\frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{r+\alpha}} \tag{95}$$

$$= \frac{\binom{r+\alpha}{r}}{\binom{r+\alpha}{r}-1} =: b_r \tag{96}$$

holds. Further, we notice that b_r is decreasing in r, since

$$b_{r+1} = \frac{\binom{r+1+\alpha}{r+1}}{\binom{r+1+\alpha}{r+1}-1} \tag{97}$$

$$= \frac{1}{1-\frac{1}{\binom{r+1+\alpha}{r+1}}} \tag{98}$$

$$= \frac{1}{1-\frac{r+1}{r+1+\alpha}\frac{1}{\binom{r+\alpha}{r}}} \tag{99}$$

$$< \frac{1}{1-\frac{1}{\binom{r+\alpha}{r}}} \tag{100}$$

$$= b_r \tag{101}$$

for each $r\in\mathbb{N}^+$. Thus, considering that $r\in[\Lambda-\alpha+1]$, we can further write

$$\frac{L_{\text{UB}}(r)}{L^\star(r)} \leq \frac{\binom{r+\alpha}{r}}{\binom{r+\alpha}{r}-1} \tag{102}$$

$$\leq \frac{\alpha+1}{\alpha} \tag{103}$$

where the last term is upper bounded when α is set to its minimum value. Now, after neglecting the value $\alpha=1$ — in which case the corresponding achievable performance in Theorem 1 was already proved to be exactly optimal in [6] — we focus on the case where $\alpha\in[2:\Lambda]$, which implies that

$$\frac{L_{\text{UB}}(r)}{L^\star(r)} \leq \frac{\alpha+1}{\alpha} \tag{104}$$

$$\leq \frac{3}{2}. \tag{105}$$

The proof is concluded. □

Appendix B
Proof of Order Optimality in Theorem 6

From Theorem 4 we know that $L^\star_{\text{max-link}}(r)$ is upper bounded as

$$L^\star_{\text{max-link}}(r) \leq L_{\text{max-link,UB}}(r) \tag{106}$$

$$= \max\left(\sum_{j\in[\Lambda]}\frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)}\frac{\tilde{a}^j_\star}{N}, \sum_{j\in[\Lambda]}\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}\frac{\tilde{a}^j_\star}{N}\right) \tag{107}$$

$$\leq \sum_{j\in[\Lambda]}\left(\frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}\right)\frac{\tilde{a}^j_\star}{N} \tag{108}$$

$$= \sum_{j\in[\Lambda]} c_j\frac{\tilde{a}^j_\star}{N} \tag{109}$$

where the coefficient c_j is defined as

$$c_j := \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}. \tag{110}$$

At the same time, we know from Theorem 5 that $L^\star_{\text{max-link}}(r)$ is lower bounded as

$$\begin{aligned} L^\star_{\text{max-link}}(r) &\geq L_{\text{max-link,LB}}(r) && (111)\\ &= \frac{1}{2}\sum_{j\in[\Lambda]}\left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}}+\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}\right)\frac{\tilde{a}^j_\star}{N} && (112)\\ &= \frac{1}{2}\sum_{j\in[\Lambda]} d_j\frac{\tilde{a}^j_\star}{N} && (113)\end{aligned}$$

where the coefficient d_j is defined as

$$d_j \coloneqq \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}}+\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}. \quad (114)$$

Hence, we can evaluate the gap to optimality from the ratio $L_{\text{max-link,UB}}(r)/L_{\text{max-link,LB}}(r)$. In particular, we have

$$\begin{aligned} \frac{L_{\text{max-link,UB}}(r)}{L_{\text{max-link,LB}}(r)} &\leq 2\frac{\sum_{j\in[\Lambda]} c_j\tilde{a}^j_\star/N}{\sum_{j\in[\Lambda]} d_j\tilde{a}^j_\star/N} && (115)\\ &= 2\frac{\sum_{j\in[\Lambda]:\tilde{a}^j_\star>0} c_j\tilde{a}^j_\star/N}{\sum_{j\in[\Lambda]:\tilde{a}^j_\star>0} d_j\tilde{a}^j_\star/N} && (116)\\ &\leq 2\max_{j\in[\Lambda]:\tilde{a}^j_\star>0}\frac{c_j\tilde{a}^j_\star/N}{d_j\tilde{a}^j_\star/N} && (117)\\ &= 2\max_{j\in[\Lambda]:\tilde{a}^j_\star>0}\frac{c_j}{d_j} && (118)\\ &\leq 2\max_{j\in[\Lambda]}\frac{c_j}{d_j} && (119)\\ &= 2\max\left(\max_{j\in[\Lambda-\alpha]}\frac{c_j}{d_j},\max_{j\in[\Lambda-\alpha+1:\Lambda]}\frac{c_j}{d_j}\right). && (120)\end{aligned}$$

Now, we can see that $c_j = d_j$ when $j > \Lambda-\alpha$. Else, when $j\in[\Lambda-\alpha]$, we have

$$\begin{aligned} \frac{c_j}{d_j} &= \frac{\frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)}+\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}}{\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}}+\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}} && (121)\\ &\leq \max\left(\frac{\frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)}}{\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}}},\frac{\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}}{\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}}\right) && (122)\\ &= \max\left(\frac{\binom{\Lambda-\alpha}{j}}{\left(\binom{j+\alpha}{j}-1\right)}\frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{\alpha+j}},1\right). && (123)\end{aligned}$$

Since we know from Appendix A that

$$\frac{\binom{\Lambda-\alpha}{j}}{\left(\binom{j+\alpha}{j}-1\right)}\frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{\alpha+j}} \leq \frac{\alpha+1}{\alpha} \quad (124)$$

we can further write

$$\begin{aligned} \max_{j\in[\Lambda-\alpha]}\frac{c_j}{d_j} &\leq \max_{j\in[\Lambda-\alpha]}\max\left(\frac{\binom{\Lambda-\alpha}{j}}{\left(\binom{j+\alpha}{j}-1\right)}\frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{\alpha+j}},1\right) && (125)\\ &\leq \max\left(\frac{\alpha+1}{\alpha},1\right). && (126)\end{aligned}$$

Hence, we can conclude that

$$\begin{aligned} \frac{L_{\text{max-link,UB}}(r)}{L_{\text{max-link,LB}}(r)} &\leq 2\max\left(\max_{j\in[\Lambda-\alpha]}\frac{c_j}{d_j},\max_{j\in[\Lambda-\alpha+1:\Lambda]}\frac{c_j}{d_j}\right) && (127)\\ &\leq 2\max\left(\max\left(\frac{\alpha+1}{\alpha},1\right),1\right) && (128)\\ &\leq 2\max\left(\max\left(2,1\right),1\right) && (129)\\ &= 4. && (130)\end{aligned}$$

The proof is concluded. □

APPENDIX C
PROOF OF LEMMA 1

Consider a permutation $\boldsymbol{c} = (c_1,\dots,c_\Lambda)$ of the set $[\Lambda]$. We know that $H(D_{\mathcal{U}} \mid X_{[\Lambda]_\alpha}, C_{\mathcal{U}}) = 0$ holds for any valid shuffle scheme and for each $\mathcal{U}\in[\Lambda]_\alpha$. Given this, for $\mathcal{U}^\alpha = \{c_1,\dots,c_\alpha\}$ we can write

$$\begin{aligned} H(X_{[\Lambda]_\alpha}) &\geq H(X_{[\Lambda]_\alpha} \mid C_{\mathcal{U}^\alpha}) && (131)\\ &= H(X_{[\Lambda]_\alpha}, D_{\mathcal{U}^\alpha} \mid C_{\mathcal{U}^\alpha}) - H(D_{\mathcal{U}^\alpha} \mid X_{[\Lambda]_\alpha}, C_{\mathcal{U}^\alpha}) && (132)\\ &= H(X_{[\Lambda]_\alpha}, D_{\mathcal{U}^\alpha} \mid C_{\mathcal{U}^\alpha}) && (133)\\ &= H(D_{\mathcal{U}^\alpha} \mid C_{\mathcal{U}^\alpha}) + H(X_{[\Lambda]_\alpha} \mid C_{\mathcal{U}^\alpha}, D_{\mathcal{U}^\alpha}) && (134)\\ &= H(\mathcal{D}_{\mathcal{U}^\alpha} \mid \mathcal{C}_{\mathcal{U}^\alpha}) + H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_\alpha) && (135)\end{aligned}$$

where (131) follows from the fact that conditioning does not increase entropy, and where (133) holds because of the decodability condition $H(D_{\mathcal{U}} \mid X_{[\Lambda]_\alpha}, C_{\mathcal{U}}) = 0$ for each $\mathcal{U}\in[\Lambda]_\alpha$. Similarly, for each $i\in[\alpha+1:\Lambda]$ we can write

$$\begin{aligned} H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_{i-1}) &\geq H(X_{[\Lambda]_\alpha} \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) && (136)\\ &= H(X_{[\Lambda]_\alpha}, \mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1})\\ &\quad - H(\mathcal{D}_i \mid X_{[\Lambda]_\alpha}, \mathcal{C}_i, \mathcal{Y}_{i-1}) && (137)\\ &= H(X_{[\Lambda]_\alpha}, \mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) && (138)\\ &= H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1})\\ &\quad + H(X_{[\Lambda]_\alpha} \mid \mathcal{D}_i, \mathcal{C}_i, \mathcal{Y}_{i-1}) && (139)\\ &= H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) + H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_i) && (140)\end{aligned}$$

where again (136) is true as conditioning does not increase entropy, and where (138) follows because

$$\begin{aligned} H(\mathcal{D}_i \mid X_{[\Lambda]_\alpha}, \mathcal{C}_i, \mathcal{Y}_{i-1}) &\leq H(\mathcal{D}_i \mid X_{[\Lambda]_\alpha}, \mathcal{C}_i) && (141)\\ &\leq \sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\dots,c_i\}:\\ |\mathcal{U}^i|=\alpha, c_i\in\mathcal{U}^i}} H(D_{\mathcal{U}^i} \mid X_{[\Lambda]_\alpha}, C_{\mathcal{U}^i}) && (142)\\ &= 0 && (143)\end{aligned}$$

due to the independence of intermediate values and the decodability condition. Considering that $H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_\Lambda) = 0$, we can use iteratively the above to obtain

$$H(X_{[\Lambda]_\alpha}) \geq \sum_{i\in[\alpha:\Lambda]} H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}). \quad (144)$$

Further, we notice that $L_{\mathcal{M}} \geq H(X_{[\Lambda]_\alpha})/QNT$. This concludes the proof. □

Appendix D
Proof of Lemma 2

First, we rewrite the equality in Lemma 2 as

$$\sum_{i\in[\alpha]} \frac{\binom{\Lambda-\alpha}{j-i}\binom{\alpha-1}{i-1}}{i\binom{\Lambda}{j}} = \frac{\binom{\Lambda}{j}-\binom{\Lambda-\alpha}{j}}{\alpha\binom{\Lambda}{j}} \tag{145}$$

$$\sum_{i\in[\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j} - \binom{\Lambda-\alpha}{j} \tag{146}$$

$$\sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j}. \tag{147}$$

Thus, proving the equality in Lemma 2 is equivalent to showing that the following equality

$$\sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j} \tag{148}$$

holds. From Vandermonde's identity, we know that

$$\sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j} \tag{149}$$

and so it suffices to show that

$$\sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i}. \tag{150}$$

Consider first the case $j \leq \alpha$. This means that we can write

$$\begin{aligned}\sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} &= \sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} \\ &\quad + \underbrace{\sum_{i\in[j+1:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i}}_{=0} \qquad (151)\\ &= \sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} \qquad (152)\end{aligned}$$

where $\sum_{i\in[j+1:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = 0$ since we have $\binom{\Lambda-\alpha}{j-i} = 0$ for $i \in [j+1:\alpha]$. Similarly, if we consider $j \geq \alpha$, we have

$$\begin{aligned}\sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} &= \sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} \\ &\quad + \underbrace{\sum_{i\in[\alpha+1:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i}}_{=0} \qquad (153)\\ &= \sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} \qquad (154)\end{aligned}$$

where $\sum_{i\in[\alpha+1:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = 0$ since we have $\binom{\alpha}{i} = 0$ for $i \in [\alpha+1:j]$. Hence, for any value of $j \in [0:\Lambda]$, we can conclude that

$$\sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j}. \tag{155}$$

The proof is concluded. □

References

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[2] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010.

[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, pp. 1514–1529, Mar. 2018.

[4] U. Kumar and J. Kumar, "A comprehensive review of straggler handling algorithms for MapReduce framework," *Int. J. Grid Distrib. Comput.*, vol. 7, no. 4, pp. 139–148, Aug. 2014.

[5] Z. Zhang, L. Cherkasova, and B. T. Loo, "Performance modeling of MapReduce jobs in heterogeneous cloud environments," in *2013 IEEE 6th Int. Conf. Cloud Comput.*, 2013, pp. 839–846.

[6] 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.

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

[8] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[9] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial coded design for heterogeneous distributed computing," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5672–5685, Sep. 2021.

[10] ——, "A combinatorial design for cascaded coded distributed computing on general networks," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5686–5700, Sep. 2021.

[11] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *2018 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1291–1295.

[12] K. Wan, M. Ji, and G. Caire, "Topological coded distributed computing," in *GLOBECOM 2020 - 2020 IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[13] S. R. Srinivasavaradhan, L. Song, and C. Fragouli, "Distributed computing trade-offs with random connectivity," in *2018 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1281–1285.

[14] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2643–2654, Oct. 2017.

[15] Q. Yan, S. Yang, and M. Wigger, "A storage-computation-communication tradeoff for distributed computing," in *2018 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–5.

[16] ——, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," in *2018 IEEE Inf. Theory Workshop (ITW)*, Nov. 2018, pp. 1–5.

[17] ——, "Storage-computation-communication tradeoff in distributed computing: Fundamental limits and complexity," *IEEE Trans. Inf. Theory*, vol. 68, no. 8, pp. 5496–5512, Aug. 2022.

[18] J. S. Ng, W. Y. B. Lim, N. C. Luong, Z. Xiong, A. Asheralieva, D. Niyato, C. Leung, and C. Miao, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1800–1837, 2021.

[19] Y. Shan, B. Wang, J. Yan, Y. Wang, N. Xu, and H. Yang, "FPMR: MapReduce framework on FPGA," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays - FPGA '10*, Feb. 2010, pp. 93–102.

[20] Y.-M. Choi and H. K.-H. So, "Map-reduce processing of k-means algorithm with FPGA-accelerated computer cluster," in *2014 IEEE 25th Int. Conf. Application-Specific Systems, Architectures and Processors*, Jun. 2014.

[21] P. N. Muralidhar, D. Katyal, and B. S. Rajan, "Maddah-Ali-Niesen scheme for multi-access coded caching," in *2021 IEEE Inf. Theory Workshop (ITW)*, Oct. 2021, pp. 1–6.

[22] F. Brunero and P. Elia, "Fundamental limits of combinatorial multi-access caching," *IEEE Trans. Inf. Theory*, vol. 69, no. 2, pp. 1037–1056, Feb. 2023.

[23] K. Vaidya and B. S. Rajan, "Cache-aided multi-access multi-user private information retrieval," in *2022 20th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, Sep. 2022, pp. 243–253.

Federico Brunero (M '23) was born in Torino, Italy, in 1995. He received the B.Sc. degree in Telecommunications Engineering from Politecnico di Torino in 2017, the M.Sc. degree in Communications and Computer Networks Engineering from Politecnico di Torino in 2019, the M.Sc. degree in Electrical and Computer Engineering from University of Illinois Chicago (UIC) in 2019, the M.Sc. degree in Data Science and Engineering from Institute Mines-Télécom and EURECOM in 2021, and the Ph.D. degree in Computer Science, Telecommunications and Electronics from Sorbonne Université in 2022. Since 2023, he has been working as Senior Research Engineer with the Optical and Quantum Communications Laboratory at the Huawei Research Center in Munich, Germany. His research interests include digital signal processing, information theory, coding theory and machine learning.

Petros Elia (M '09) received the B.Sc. degree from the Illinois Institute of Technology and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Southern California (USC), Los Angeles, in 2001 and 2006, respectively. He is currently a Professor with the Department of Communication Systems, EURECOM, Sophia Antipolis, France. His latest research deals with distributed computing and with the intersection of caching and communications in multiuser settings. He has also worked in the area of complexity-constrained communications, MIMO, queueing theory and cross-layer design, coding theory, information-theoretic limits in cooperative communications, and surveillance networks. He is a Fulbright Scholar, a co-recipient of the NEWCOM++ Distinguished Achievement Award (2008–2011) for a sequence of publications on the topic of complexity in wireless communications, and a recipient of the ERC Consolidator Grant (2017–2022) on cache-aided wireless communications.