This is a pre print version of the following article:

Uncoded Caching and Cross-level Coded Delivery for Non-uniform File Popularity / Ozfatura, E., Gunduz, D.. - In: IEEE TRANSACTIONS ON INFORMATION THEORY. - ISSN 0018-9448. - 68:10(2022), pp. 6842-6859. [10.1109/TIT.2022.3175900]

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

16/11/2026 08:59

(Article begins on next page)

Uncoded Caching and Cross-level Coded Delivery for Non-uniform File Popularity

Emre Ozfatura and Deniz Gündüz

arXiv:1802.01135v4 [cs.IT] 6 Dec 2021

Abstract

Proactive content caching at user devices and coded delivery is studied for a non-uniform file popularity distribution. A novel centralized uncoded caching and coded delivery scheme, called *cross-level coded delivery (CLCD)*, is proposed, which can be applied to large file libraries under non-uniform demands. In the CLCD scheme, the same sub-packetization is used for all the files in the library in order to prevent additional zero-padding in the delivery phase, and unlike the existing schemes in the literature, users requesting files from different popularity groups can still be served by the same multicast message in order to reduce the delivery rate. Simulation results indicate more than 10% reduction in the average delivery rate for typical Zipf distribution parameter values.

Index Terms

Content caching, non-uniform demands, coded delivery, multi-level caching

I. Introduction

Proactive caching is a well known strategy to reduce the network load and delivery latency in content delivery networks. Recently, it has been shown that proactive caching and coded delivery, together, can further help to reduce the network load. The "shared-link problem", in which a server with a finite file library serves the demands of cache-equipped users over a shared error-free link is analyzed in [1], where the objective is to design a caching and delivery strategy to minimize the load on the shared link, often referred to as the "delivery rate" in the literature.The proposed solution consists of two phases. In the *placement phase*, all the files in the library are divided into sub-files, and each user proactively caches a subset of these sub-files. In the *delivery phase*, the server multicasts XORed versions of the requested sub-files through the shared link, such that each user can recover its demand by exploiting both the cached sub-files and the multicast messages. In the scheme proposed in [1], the server controls the caching decisions of the users in a centralized manner. However, in [2], the authors have shown that similar gains can still be achieved when the caching is performed in a random and decentralized manner under certain assumptions. It is shown in [3] that the delivery rate can be further reduced with a coded delivery scheme that exploits the common requests. Most of the initial studies on coded caching and delivery have been built upon certain simplifying unrealistic assumptions, such as the presence of an error-free broadcast link, uniform user cache capacities, and uniform file demands. A plethora of works have since focused on trying to remove or relax these assumptions in order to bring coded caching and delivery closer to reality. To this end, the case of non-uniform cache sizes is studied in [4]–[6], extension to unequal link rates is studied in [5], [7], [8], and non-uniform file popularity is considered in [9]–[15].

In this work, we are interested in the case of non-uniform demands as this is commonly encountered in practice [16]–[19].It has been shown that by taking into account the statistics of the content popularity and by allocating the cache memory to the files in the library based on this statistics, communication load can be further reduced [9]–[15]. Most of the papers dealing with non-uniform demands consider decentralized caching in the placement phase. In [9], the authors proposed a *group-based memory sharing* approach for the placement phase, where the files in the library are grouped according to their popularities and the available cache memory at user devices is distributed among these groups. Then, in the delivery phase, the coded delivery scheme in [2] is used to deliver the missing parts of the demands. In [10], the authors introduced both a placement and a delivery strategy for non-uniform demands. Optimization of the delivery phase is analyzed as an index coding problem and the multicast messages are constructed based on the *clique cover problem*, where each missing sub-file[1] (requested but not available in the user's cache) is considered as a vertex of a conflict graph, and two sub-files are delivered together in the same multicast message if there is no edge between the corresponding vertices. The chromatic number of the conflict graph gives the number of required multicast messages, equivalently, the delivery rate normalized by the sub-file size. Due to the NP-hardness of the original graph coloring problem, the authors introduced a low-complexity suboptimal *greedy constrained coloring (GCC)* scheme. For the placement phase, similarly to [9], a group-based memory sharing approach with only two groups is considered,

Emre Ozfatura and Deniz Gündüz are with the Information Processing and Communications Lab, Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, UK.

This work was supported by EC H2020-MSCA-ITN-2015 project SCAVENGE under grant number 675891, and by the European Research Council project BEACON under grant number 677854.

A shorter version of this paper was presented at IEEE International Conference on Communications (ICC), in 20-24 May 2018.

[1]We use the term *sub-file* to be consistent with the later part of the paper, while the papers studying the asymptotic performance of decentralized coded caching typically use the term "packet" or "bit".

where only the files in the first group are cached. They show that the proposed scheme is order-optimal when the popularity of the files follows a Zipf distribution. Results of [10] are further extended by [11] to show that group-based memory sharing with two groups[2] is order-optimal for any popularity distribution. The group-based memory sharing approach for non-uniform demands is also studied in [12], where the users do not have caches, but instead have access to multiple cache-enabled access points and the shared link is between the server and the access points. All these works analyze the asymptotic performance (as the number of sub-files goes to infinity) of the decentralized coded caching approach.

Decentralized coded caching with finite number of sub-files is recently studied in [15], [20] and [21] and to provide an efficient low-complexity solution to the clique cover problem. Hence, as a common objective, the proposed methods in these works aim to minimize the delivery rate for a given placement strategy using heuristic approaches.

For the centralized coded caching problem under a non-uniform demand distribution, a novel content placement strategy is introduced in [13] and [14] independently. In these papers, content placement is formulated as a linear optimization problem, where each file is divided into $K+1$ disjoint fragments, not necessarily of equal size. The kth fragment of each file is cached as in the placement phase of [1] with parameter $t = k-1$. In the proposed approach, the size of each fragment of each file is represented by a variable in the linear optimization problem, and the solution of the optimization problem reveals how each file should be divided into $K+1$ fragments. We remark that the group-based memory sharing scheme of [9] is a special case of these approaches, where in each file group only one fragment has non-zero size. Since this approach considers the most general placement strategy, we call it *general memory sharing*, while we will refer to the original scheme in [1] as the *naive memory sharing* scheme. Note that, in the delivery phase, when a multicast message contains sub-files with different sizes, the size of the message is determined according to the largest sub-file in the message, and the smaller are zero-padded, which results in inefficiency.

Hence, although the above scheme allows each file to be divided into different size fragments, when the file library is large, the proposed policy tends to divide the library into small number of groups, where, in each group, the size of the fragments are identical. This is because unequal fragment sizes among users requires zero-padding. In particular, we observe that, for a large file library, the proposed scheme in [14] often tends to classify the files into two groups according to their popularities, such that only the popular files are cached in an identical manner. We highlight here that without mitigating the sub-file mismatch issue, which induces zero padding, the advantage of optimization over fragment sizes is limited. Another limitation of the strategy proposed in [14] is its complexity. The number variables in the optimization problem is $N(K+1)$. Although linear optimization can be solved in polynomial time, it may still be infeasible when the library size is large (in practical scenarios the number of files are often considered to be between $10^4 - 10^6$ [17]). In the case of less skewed popularity distributions using more than two groups can be effective in reducing the delivery rate, together with a more intelligent coded delivery design that reduces or prevents zero-padding. This motivates the proposed *cross-level coded delivery (CLCD)* scheme, whose goal is to design a zero-padding-free coded delivery scheme for a given group structure. The zero-padding problem in the delivery phase of centralized coded caching was previously studied in [15], where a heuristic *"bit borrowing "* approach is introduced. The proposed algorithm greedily pairs the bits of the sub-files for a given placement strategy.

In this paper, different from the aforementioned works, we introduce a novel centralized coded caching framework, where the delivery and the placement phases are optimized jointly. In the proposed CLCD strategy, the placement phase is limited to three file groups, formed according to popularities. Then, for each possible demand realization, i.e., the number of users requesting a file from each group, we obtain the optimal delivery scheme and the corresponding delivery rate. Finally, using this information we find the optimal group sizes based on the popularity of the files in the library. We will show that the proposed CLCD scheme provides a noticable reduction in the average delivery rate, up to 10-20%, compared to the state-of-the-art.
The rest of the paper is organized as follows. In Section II, we introduce the system model and the problem definition. In Section III, we introduce the CLCD scheme and analyze it for a particular scenario, where we impose certain constraints on the placement phase. In Section IV, we extend our analysis by lifting some constraints on the placement phase, and in Section V, we present numerical comparisons of the proposed CLCD scheme with the state-of-the-art. In Section VI, we present the most general form of the CLCD scheme without any constraints on the placement phase, and finally, we conclude the paper in Section VII.

II. System Model

We consider a single content server with a library of N files, each of size F bits, denoted by $W_1, \dots, W_N$, serving K users, each with a cache memory of capacity MF bits. The users are connected to the server through a shared error-free link. We follow the two-phase model in [1]. Caches are filled during the *placement phase* without the knowledge of particular user demands. User requests are revealed in the delivery phase, and are satisfied simultaneously.

The request of user k is denoted by d_k, $d_k \in [N] \triangleq \{1, \dots, N\}$, and the demand vector is denoted by $\mathbf{d} \triangleq (d_1, \dots, d_K)$. The corresponding delivery rate $R(\mathbf{d})$ is defined as the total number of bits sent over the shared link, normalized by the file size. We assume that the user requests are independent of each other, and each user requests file W_n with probability p_n, where

[2]In certain cases there might be an additional group consist of only a single file.

Algorithm 1: Naive coded delivery for non-uniform content placement

Input : $\mathcal{K}^h$, $\mathcal{K}^l$

1 **for** $a \in \{h, l\}$ **do**
2 **for** $\mathcal{S} \subseteq: |\mathcal{S}| = t_a$ **do**
3 **if** $\mathcal{S} \cap \mathcal{K}^a \neq \emptyset$ **then**
4 Multicast $\bigoplus_{s \in \mathcal{K}^a} W_{d_s, \mathcal{S}\setminus\{s\}}$
5 **end**
6 **end**
7 **end**
8 Serve users in $\mathcal{K}^r$ with unicast messages.

$p_1 \geq p_2 \geq \cdots \geq p_N$, and $\sum_{n=1}^{N} p_n = 1$. Let $P(\mathbf{d})$ denote the probability of observing demand vector $\mathbf{d}$, where $P(\mathbf{d}) = \prod_{k=1}^{K} p_{d_k}$. We want to minimize the average delivery rate $\bar{R}$, defined as

$$\bar{R} \triangleq \sum_{\mathbf{d}} P(\mathbf{d}) R(\mathbf{d}). \tag{1}$$

Next, we explain the uncoded caching and centralized coded delivery scheme, introduced in [1]. We say that a file W_n is *cached at level* t, if it is divided into $\binom{K}{t}$ non-overlapping sub-files of equal size, and each sub-file is cached by a distinct subset of t users. Then, each sub-file can be identified by an index term $\mathcal{I}$, where $\mathcal{I} \subseteq [K]$ and $|\mathcal{I}| = t$, such that sub-file $W_{n,\mathcal{I}}$ is cached by users $k \in \mathcal{I}$. Following a placement phase in which all the files are cached at level t, as proposed in [1], in the delivery phase, for each subset $\mathcal{S} \subseteq [K]$, $|\mathcal{S}| = t + 1$, all the requests of the users in $\mathcal{S}$ can be served simultaneously by multicasting

$$\bigoplus_{s \in \mathcal{S}} W_{d_s, \mathcal{S}\setminus\{s\}}. \tag{2}$$

Thus, with a single multicast message the server can deliver $t + 1$ sub-files, and achieve a *multicasting gain* of $t + 1$.

III. Cross-level Coded Delivery (CLCD) scheme

A. An introductory overview

In the proposed CLCD scheme, the file library is divided into three groups according to their popularity: the most popular N_h files are called the *high-level* files, the less popular N_l files are called the *low-level* files, and the remaining least popular $N_r = N - (N_h + N_l)$ files are called the *zero-level* files. All the files in the same group are cached at the same level. Hence, the scheme can be represented by five parameters t_h, t_l, N_h, N_l, N_r, where t_h and t_l represent the caching levels of the low-level and high-level files, respectively, whereas the zero-level files are not cached at all. Hence, we use $CL(t_h, t_l, N_h, N_l, N_r)$ to denote the overall caching strategy.

For the given parameters t_h, t_l, N_h, N_l, N_r, the required normalized cache-memory size $M(t_h, t_l, N_h, N_l, N_r)$ is given by

$$M(t_h, t_l, N_h, N_l, N_r) \triangleq \frac{t_h N_h + t_l N_l}{K}, \tag{3}$$

since caching a file at level t means dividing it into $\binom{K}{t}$ sub-files and letting each user cache $\binom{K-1}{t-1}$ sub-files; that is, each user caches t/K portion of each file.

The key contribution of the CLCD scheme is in the delivery phase. When the user demands are revealed, they are clustered into three groups, denoted by $\mathcal{K}^h$, $\mathcal{K}^l$ and $\mathcal{K}^r$, which correspond to the set of users that request high-level, low-level, and zero-level files, respectively. A naive approach would be to serve the users in each set $\mathcal{K}^a$, $a \in \{h, l, r\}$, separately as in Algorithm 1. The main limitation of this approach is that, although multicasting gain of $t_a + 1$ is targeted at line 4 of Algorithm 1, the achieved multicasting gain is limited to $|\mathcal{S} \cap \mathcal{K}^a|$. The main objective of the CLCD strategy is to design a proper way of forming multicast messages such that users from two different groups can be targeted as well in order to reduce the communication load. To this end, CLCD addresses two main challenges; first, how to form the set of target users for multicating, second, how to align sub-file sizes. The latter challenge stems from the mismatch between the sizes of high-level and low-level sub-files. This mismatch can be resolved either at the placement or at the delivery phase. That is, the size of each multicast message can be fixed, or can be adjusted during the delivery phase according to the targeted users. This will be further clarified later in the paper. To simplify the presentation and gradually introduce the key ideas behind the CLCD, in the following subsections we will start introducing special cases of the general CLCD scheme, and gradually extend the analysis to the more general version.

User	1	2	3	4	5	6	7
index of cached sub-files	$\{1,2\}$	$\{1,2\}$	$\{1,3\}$	$\{1,4\}$	$\{1,5\}$	$\{1,6\}$	$\{1,7\}$
	$\{1,3\}$	$\{2,3\}$	$\{2,3\}$	$\{2,4\}$	$\{2,5\}$	$\{2,6\}$	$\{2,7\}$
	$\{1,4\}$	$\{2,4\}$	$\{3,4\}$	$\{3,4\}$	$\{3,5\}$	$\{3,6\}$	$\{3,7\}$
	$\{1,5\}$	$\{2,5\}$	$\{3,5\}$	$\{4,5\}$	$\{4,5\}$	$\{4,6\}$	$\{4,7\}$
	$\{1,6\}$	$\{2,6\}$	$\{3,6\}$	$\{4,6\}$	$\{5,6\}$	$\{5,6\}$	$\{5,7\}$
	$\{1,7\}$	$\{2,7\}$	$\{3,7\}$	$\{4,7\}$	$\{5,7\}$	$\{6,7\}$	$\{6,7\}$

TABLE I: Sub-file placement: for each user the sub-files in red are cached for all the files, whereas the sub-files in blue are cached for only high-level files.

B. $\mathrm{CL}(\mathrm{t_h}=2,\mathrm{t_l}=1,\mathrm{N_h},\mathrm{N_l},\mathrm{N_r}=0)$

In this section, we focus on a particular case of CLCD scheme, $CL(2,1,N_h,N_l,0)$, that is high-level files are cached at level $t_h=2$ and low-level files are cached at level $t_l=1$, $N_h=N-N_l$, that is, all the files are cached at either at level 1 or level 2. We will later extend our analysis to the more general CLCD schemes.

1) Placement phase: We first want to remark that the placement strategy employed here is not specific to $t_h=2$ and can be utilized for more general schemes as well, as long as $t_l=1$. The placement of the high-level files is identical to the one in [1] such that each file is divided into $\binom{K}{t_h}$ sub-files and the high-level files are cached at level t_h; that is, each user caches $\binom{K-1}{t_h-1}$ sub-files of each high-level file.

The key difference appears in the placement of the low-level files; instead of dividing files into $\binom{K}{1}$ sub-files, we use the same sub-packetization scheme used for the high-level files above and group them into K disjoint and equal-size subsets so that each user caches the sub-files of a different subset. Equivalently, each sub-file is cached by only a single user, and each user caches only $\binom{K-1}{t_h-1}/t_h$ sub-files of each low-level file exclusively. We assume here that $\binom{K}{t_h}$ is divisible by K, which holds for any t_h when K is a prime number. We emphasize that this assumption is not required for CLCD in general; however, under this assumption all the files are divided into equal number of sub-files, thus the alignment problem can be resolved at the placement phase as we explain next.

Although the presented placement scheme is general, in the case of $t_h=2$, we can provide a closed form expression for the achievable delivery rate as a function of the requested files from each caching level, which is denoted by $k^a \triangleq |\mathcal{K}^a|, a\in\{l,h,r\}$, and the corresponding delivery strategy can be obtained in polynomial time. This is particularly important, since it implies a low complexity coded delivery design with a certain achievable rate guarantee in the low cache-memory regime. Next, we provide an example to better illustrate the proposed placement strategy.

Example 1. Consider a network of $K=7$ users and a library of $N=7$ files with decreasing popularity. To simplify the notation, we will denote the files as $W_1=A$, $W_2=B$, $W_3=C$, $W_4=D$, $W_5=E$, $W_6=F$, $W_7=G$. We set the normalized cache size as $M=12/7$. For the given cache size, we have $N_h=5$; that is, the most popular files, $\{A,B,C,D,E\}$, are treated as high-level files, whereas, G and F as low level files. Each file is divided into $\binom{K}{2}=21$ sub-files, and each user stores $K-1=6$ sub-files of each high-level file, and $\frac{K-1}{2}=3$ sub-files of each low-level file. User cache contents after the placement phase is illustrated in Table I; for each user the sub-files in red are cached for all the files, whereas the sub-files in blue are cached for only the high-level files.

Remark 1. *In Example 1, we use a* systematic placement *strategy for the low-level files; such that, for each user, the index set of the cached sub-files of a low-level file is a subset of the index set of the cached sub-files of a high level file. Although, this particular placement strategy may not affect the delivery rate performance, it can be beneficial in practice if the library and/or the file popularity dynamically varies over time and the cached content should be updated accordingly [22]. If a high-level file becomes low-level, then the user simply removes the sub-files colored with blue, and similarly, if a low-level file becomes high-level, then the user only downloads the sub-files colored with blue from the server. This systematic placement strategy can be summarized in the following way. Let $I_i=\{\mathcal{S}:|\mathcal{S}|=2,\mathcal{S}\subseteq\mathcal{K},i\in\mathcal{S}\}$ be the set of index sets of sub-files that contain i. Then, when a file W_k is cached at level 2, user i caches all the sub-files $W_{k,\mathcal{S}}$, $\mathcal{S}\in I_i$. Further, let each I_i, for $i\in[K]$, be ordered as illustrated in Table I, where the ith column includes the elements of I_i within an order. Once we have ordered sets for each user, then, starting from the first user, each user selects the first $\frac{K-1}{2}$ index set, that are not chosen, in order to cache corresponding low-level files as illustrated in Table I for $K=7$.*

2) Delivery phase: Delivery phase of the proposed scheme consists of four steps. We will continue to use Example 1 to explain the key ideas of the delivery scheme

Example 1 (continued). Assume that user k requests file W_k, $k\in[K]$, so that $\mathcal{K}^h=\{1,2,3,4,5\}$ and $\mathcal{K}^l=\{6,7\}$. Before explaining the delivery steps in detail, we want to highlight the main idea behind the CLCD scheme, which has been inspired by the *bit borrowing* approach introduced in [15]. For the given example, if the conventional coded delivery scheme is used for delivering the missing sub-files of the high-level and low-level users separately, then the server should send all the messages given in Table II. Now, consider the following three messages in Table II: $A_{26}\oplus B_{16}$, which targets high-level users 1 and 2, and F_{12} and F_{23}, which are sent as unicast messages to low-level user 6. In the cross-level design, the server decomposes the

High-level users	Multicast messages	Low-level users	Multicast messages
1 2 3	$A_{23} \oplus B_{13} \oplus C_{12}$	6 7	$G_{16} \oplus F_{17}$
1 2 4	$A_{24} \oplus B_{14} \oplus D_{12}$	6 7	$G_{26} \oplus F_{27}$
1 2 5	$A_{25} \oplus B_{15} \oplus E_{12}$	6 7	$G_{67} \oplus F_{37}$
1 3 4	$A_{34} \oplus C_{14} \oplus D_{13}$	6	F_{12}
1 3 5	$A_{35} \oplus C_{13} \oplus E_{13}$	6	F_{13}
1 4 5	$A_{45} \oplus D_{15} \oplus E_{14}$	6	F_{14}
2 3 4	$B_{34} \oplus C_{24} \oplus D_{23}$	6	F_{23}
2 3 5	$B_{35} \oplus C_{25} \oplus E_{23}$	6	F_{24}
2 4 5	$B_{45} \oplus D_{25} \oplus E_{24}$	6	F_{25}
3 4 5	$C_{45} \oplus D_{35} \oplus E_{34}$	6	F_{34}
1 2	$A_{26} \oplus B_{16}$	6	F_{35}
1 2	$A_{27} \oplus B_{17}$	6	F_{36}
1 3	$A_{36} \oplus C_{16}$	6	F_{45}
1 3	$A_{37} \oplus C_{17}$	6	F_{46}
1 4	$A_{46} \oplus D_{16}$	6	F_{47}
1 4	$A_{47} \oplus D_{17}$	6	F_{15}
1 5	$A_{56} \oplus E_{16}$	6	F_{56}
1 5	$A_{57} \oplus E_{17}$	6	F_{57}
2 3	$B_{36} \oplus C_{26}$	7	G_{12}
2 3	$B_{37} \oplus C_{27}$	7	G_{13}
2 4	$B_{46} \oplus D_{26}$	7	G_{14}
2 4	$B_{47} \oplus D_{27}$	7	G_{23}
2 5	$B_{56} \oplus E_{26}$	7	G_{24}
2 5	$B_{57} \oplus E_{27}$	7	G_{25}
3 4	$C_{46} \oplus D_{36}$	7	G_{34}
3 4	$C_{47} \oplus D_{37}$	7	G_{35}
3 5	$C_{56} \oplus E_{36}$	7	G_{36}
3 5	$C_{57} \oplus E_{37}$	7	G_{45}
4 5	$D_{56} \oplus E_{46}$	7	G_{46}
4 5	$D_{57} \oplus E_{47}$	7	G_{47}
1	A_{67}	7	G_{15}
2	B_{67}	7	G_{56}
3	C_{67}	7	G_{57}
4	D_{67}	-	-
5	E_{67}	-	-

TABLE II: Multicast messages constructed according to the conventional coded delivery scheme for Example 1.

message $A_{26} \oplus B_{16}$ into sub-files A_{26} and B_{16}, and instead, pairs them with sub-files F_{12} and F_{23} to construct cross-level messages $A_{26} \oplus F_{12}$, which targets high-level user 1 and low-level user 6, and $B_{16} \oplus F_{23}$, which targets high-level user 2 and low-level user 6. Hence, the server multicasts two messages instead of three. We call this process as *multicast message decomposition*, since the cross-level messages are constructed by decomposing a message constructed according to the conventional coded delivery scheme.

The key challenge in CLCD is deciding which messages to decompose, and how to pair the decomposed high-level sub-files with the low-level sub-files. Now, if we go back to Example 1, all the messages in blue in Table II will be used for cross-level delivery and the total number of broadcast messages will be reduced to 51 from 68 (24% reduction). In the delivery phase, the server first multicasts the messages in red in Table II, then the messages in blue are used to construct cross-level messages, and finally the remaining messages in green are sent as in the conventional coded delivery scheme. Now, we will further explain the four steps of the delivery phase:

Step 1) Intra-high-level delivery: The first step of the delivery phase is identical to that in (2) for $t = 2$. The only difference is that, now we consider only the users in $\mathcal{K}^h$, instead of $[K]$. This corresponds to the red messages in the second column of Table II.

Step 2) Intra-low-level delivery: The second step also follows (2) with $t = 1$, targeting low-level users in $\mathcal{K}^l$. These are the red messages in the last column of Table II.

Step 3) Cross-level delivery: This step is the main novelty of the CLCD scheme. First, note that each high-level user has $(K-1)/2$ sub-files in its cache that are requested by a low-level user. For instance, in Example 1, user 1 has sub-files $\{G_{12}, G_{13}, G_{14}\}$ that are requested by user 7. For $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$, let $\mathcal{H}_{i,j}$ denote the set of sub-files stored at high-level user i and requested by low-level user j, e.g., $\mathcal{H}_{1,7} \triangleq \{G_{12}, G_{13}, G_{14}\}$ in Example 1. We note that these sub-files are sent as unicast messages in the conventional coded delivery scheme. Similarly, for $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$, let $\mathcal{L}_{i,j}$ be the set of $\binom{K-2}{1} = K-2$ sub-files stored by low-level user j that are requested by high-level user i, e.g., $\mathcal{L}_{1,7} \triangleq \{A_{27}, A_{37}, A_{47}, A_{57}, A_{67}\}$ in Example 1. One can observe that a cross-level message, targeting high-level user i and low-level user j, can be constructed by taking one sub-file from each of $\mathcal{H}_{i,j}$ and $\mathcal{L}_{i,j}$, and bit-wise XORing them. For the given example, any three sub-files can be chosen from set $\mathcal{L}_{1,7}$ and paired with any of the sub-files in $\mathcal{H}_{1,7}$ to construct a cross-level message. To generalize, if there is a set of sub-files $\mathcal{F}_{i,j} \subseteq \mathcal{L}_{i,j}$ such that $|\mathcal{F}_{i,j}| = |\mathcal{H}_{i,j}|$, then we can easily construct cross-level messages that target high-level

user i and low-level user j, using any one-to-one mapping between these two sets. However, we remark that, if sub-file A_{37} is used to construct a cross-level message, i.e., $A_{37} \in \mathcal{F}_{1,7}$, then the multicast message $A_{37} \oplus C_{17}$ should also be decomposed, and the sub-file C_{17} should also be paired with a low-level file, i.e., $C_{17} \in \mathcal{F}_{3,7}$.

Hence, the main challenge in cross-level delivery is the construction of sets $\mathcal{F}_{i,j}$ in a joint manner. Before the construction process of these sets, we will define two more sets that will be useful. We note that, in the given example, sub-files $\{A_{27}, A_{37}, A_{47}, A_{57}\}$ are also cached by a high-level user, but sub-file A_{67} is cached by only low-level users. For $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$, we introduce the set $\Omega_{i,j} \subseteq \mathcal{L}_{i,j}$ of the sub-files that are requested by high-level user i, and cached by low-level user j as well as by a high-level user, e.g., $\Omega_{1,7} \triangleq \{A_{27}, A_{37}, A_{47}, A_{57}\}$. Furthermore, we introduce the set Λ_i, $i \in \mathcal{K}^h$, of the sub-files that are requested by high-level user i but cached by only low-level users, i.e., $\Lambda_i \triangleq \cup_{j \in \mathcal{K}^l} \left(\mathcal{L}_{i,j} \setminus \Omega_{i,j}\right)$. We have, in Example 1, $\Lambda_1 = \{A_{67}\}$.

In the third step of proposed CLCD scheme, our aim is to deliver all the remaining sub-files requested by the low-level users and the sub-files requested by the high-level users that are cached by only low-level users, via multicast messages, each destined for one high-level and one low-level user. More formally, we want low-level user j, $j \in \mathcal{K}^l$, to recover all the sub-files in $\cup_{i \in \mathcal{K}^h} \mathcal{H}_{i,j}$, and we want high-level user i, $i \in \mathcal{K}^h$, to recover all the sub-files in Λ_i. To this end, sets $\mathcal{F}_{i,j}$ must satisfy the following properties:

$$\Lambda_i \subseteq \cup_j \mathcal{F}_{i,j}, \ \forall i \in \mathcal{K}^h, \tag{4}$$

$$\mathcal{F}_{i,j} \cap \mathcal{F}_{i,k} = \emptyset, \ \forall i \in \mathcal{K}^h \text{ and } \forall j,k \in \mathcal{K}^l, \tag{5}$$

where (4) ensures that each high-level user collects its missing sub-files that are available only in the caches of the low-level users and (5) guarantees that high-level users do not receive the same sub-file multiple times.

Next, we show how to construct the sets $\mathcal{F}_{i,j}$, $i \in \mathcal{K}^h$, $j \in \mathcal{K}^l$. In order to ensure (4) and (5), Λ_i is partitioned into subsets $\left\{\Lambda_{i,j}\right\}_{j \in K^l}$ with approximately uniform cardinality, i.e., $|\Lambda_{i,k}| - |\Lambda_{i,j}| \leq 1$, $\forall j,k \in \mathcal{K}^l$, $j \neq k$, and such that $\Lambda_{i,j} \subseteq \mathcal{L}_{i,j}$ holds for all $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$. Further details on approximately uniform partitioning are provided in Appendix C. We note that, if $\Lambda_{i,j} \subseteq \mathcal{F}_{i,j}$, then (4) holds. We also assume that the same partitioning is applied to all Λ_i's. Partitions of Λ_i's for Example 1 are illustrated in Table III.

For given $\Omega_{i,j}$ and $\Lambda_{i,j}$, $\mathcal{F}_{i,j}$ can be constructed as follows: $\mathcal{F}_{i,j} = \Lambda_{i,j} \cup \left\{\Omega_{i,j} \setminus \Delta_{i,j}\right\}$, for some $\Delta_{i,j} \subseteq \Omega_{i,j}$. Then, from the construction, one can easily verify that (5) holds. We note that, in this construction, $\Delta_{i,j}$ simply denotes the sub-files in $\mathcal{L}_{i,j}$ that are not used in a cross-level message. In Example 1, this corresponds to the green multicast messages in Table II. In particular, if multicast message $A_{27} \oplus B_{17}$ is not decomposed for cross-level delivery, then $A_{27} \in \Delta_{17}$ and $B_{17} \in \Delta_{27}$. Hence, the ultimate aim is to find a proper way of constructing sets Δ_{ij}, which is equivalent to deciding which multicast messages to be decomposed.

In order to ensure the initial requirement of $|\mathcal{F}_{i,j}| = |\mathcal{H}_{i,j}|$, we need to show that it is possible to construct $\Delta_{i,j}$, $i \in \mathcal{K}^h$, $j \in \mathcal{K}^l$, which satisfy the following equality

$$|\mathcal{H}_{i,j}| = |\Lambda_{i,j}| + |\Omega_{i,j}| - |\Delta_{i,j}|. \tag{6}$$

We note that, if the following inequality holds

$$|\Lambda_{i,j}| \leq |\mathcal{H}_{i,j}| \leq |\Lambda_{i,j}| + |\Omega_{i,j}|, \tag{7}$$

$\Delta_{i,j}$ satisfying (6) can always be found.

From the construction, we know that $|\mathcal{H}_{i,j}| = \frac{K-1}{2}$; however, $|\Omega_{i,j}|$ and $|\Lambda_{i,j}|$ depend on the realization of the user demands, i.e., $|\Omega_{i,j}| = k^h - 1$ and $|\Lambda_{i,j}| = \left\lceil \frac{k^l-1}{2} \right\rceil$ or $|\Lambda_{i,j}| = \left\lfloor \frac{k^l-1}{2} \right\rfloor$ due to the approximately uniform partitioning. Accordingly,

$$|\Omega_{i,j}| + |\Lambda_{i,j}| = \left\lfloor \frac{K-1}{2} + \frac{k^h-2}{2} \right\rfloor, \text{ or} \tag{8}$$

$$|\Omega_{i,j}| + |\Lambda_{i,j}| = \left\lceil \frac{K-1}{2} + \frac{k^h-2}{2} \right\rceil. \tag{9}$$

One can observe that, when $k^h \geq 2$, $|\Omega_{i,j}| + |\Lambda_{i,j}| \geq |\mathcal{H}_{i,j}| = \frac{K-1}{2}$ in both cases. If $k^h = 1$, the high-level file is considered as a low-level file in the delivery phase, and the achievable rate becomes $(K-1)/2$. In the remainder of this section, we assume $k^h \geq 2$. One can also observe that $|\Lambda_{i,j}| \leq |\mathcal{H}_{i,j}|$, since $k^l \leq K$. Let $n_{i,j}$ be the cardinality of the set $\Delta_{i,j}$ that satisfies (6), i.e., $n_{i,j} \triangleq |\Omega_{i,j}| + |\Lambda_{i,j}| - |\mathcal{H}_{i,j}|$. We can consider any subset of $\Omega_{i,j}$ with cardinality $n_{i,j}$ as $\Delta_{i,j}$ to construct $\mathcal{F}_{i,j}$. Eventually, the overall problem can be treated as choosing the multicast messages that will be decomposed, colored with blue in Table II, so that the sub-files that are not used for cross-level delivery satisfy the constraint $|\Delta_{i,j}| = n_{i,j}$, $\forall\ i \in \mathcal{K}^h, j \in \mathcal{K}^l$. Once the decomposed files and $\Delta_{i,j}$ are decided, $\mathcal{F}_{i,j}$ can be constructed easily as illustrated in Table III, which also lists all the cross-level messages.

We remark that when $W_{d_i,\{k,j\}} \in \Delta_{i,j}$, for $i,k \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$, this also requires $W_{d_k,\{i,j\}} \in \Delta_{k,j}$. Hence, for a particular $j \in \mathcal{K}^l$ with $n_{i,j} = n$, for all $i \in \mathcal{K}^h$, construction of the sets $\Delta_{i,j}$ is equivalent to finding $k^h n/2$ number of (i,k) pairs such

(i,j)	$\Omega_{i,j}$	$\Lambda_{i,j}$	$\Delta_{i,j}$	$\mathcal{F}_{i,j}$	$\mathcal{H}_{i,j}$	Cross-level messages
(1,6)	$\{A_{26}, A_{36}, A_{46}, A_{56}\}$	$\emptyset$	$\{A_{46}\}$	$\{A_{26}, A_{36}, A_{56}\}$	$\{F_{12}, F_{13}, F_{14}\}$	$A_{26} \oplus F_{12},\ A_{36} \oplus F_{13},\ A_{56} \oplus F_{14}$
(1,7)	$\{A_{27}, A_{37}, A_{47}, A_{57}\}$	$\{A_{67}\}$	$\{A_{27}, A_{47}\}$	$\{A_{67}, A_{37}, A_{57}\}$	$\{G_{12}, G_{13}, G_{14}\}$	$A_{67} \oplus G_{12},\ A_{37} \oplus G_{13},\ A_{57} \oplus G_{14}$
(2,6)	$\{B_{16}, B_{36}, B_{46}, B_{56}\}$	$\emptyset$	$\{B_{56}\}$	$\{B_{16}, B_{36}, B_{46}\}$	$\{F_{23}, F_{24}, F_{25}\}$	$B_{16} \oplus F_{23},\ B_{36} \oplus F_{24},\ B_{46} \oplus F_{25}$
(2,7)	$\{B_{17}, B_{37}, B_{47}, B_{57}\}$	$\{B_{67}\}$	$\{B_{17}, B_{57}\}$	$\{B_{67}, B_{37}, B_{47}\}$	$\{G_{23}, G_{24}, G_{25}\}$	$B_{67} \oplus G_{23},\ B_{37} \oplus G_{24},\ B_{47} \oplus G_{25}$
(3,6)	$\{C_{16}, C_{26}, C_{46}, C_{56}\}$	$\emptyset$	$\{C_{46}\}$	$\{C_{16}, C_{26}, C_{56}\}$	$\{F_{34}, F_{35}, F_{36}\}$	$C_{16} \oplus F_{34},\ C_{26} \oplus F_{35},\ C_{56} \oplus F_{36}$
(3,7)	$\{C_{17}, C_{27}, C_{47}, C_{57}\}$	$\{C_{67}\}$	$\{C_{47}, C_{57}\}$	$\{C_{67}, C_{17}, C_{27}\}$	$\{G_{34}, G_{35}, G_{36}\}$	$C_{67} \oplus G_{34},\ C_{17} \oplus G_{35},\ C_{27} \oplus G_{36}$
(4,6)	$\{D_{16}, D_{26}, D_{36}, D_{56}\}$	$\emptyset$	$\{D_{36}\}$	$\{D_{16}, D_{26}, D_{56}\}$	$\{F_{45}, F_{46}, F_{47}\}$	$D_{16} \oplus F_{45},\ D_{26} \oplus F_{46},\ D_{56} \oplus F_{47}$
(4,7)	$\{D_{17}, D_{27}, D_{37}, D_{57}\}$	$\{D_{67}\}$	$\{D_{17}, D_{37}\}$	$\{D_{67}, D_{27}, D_{57}\}$	$\{G_{45}, G_{46}, G_{47}\}$	$D_{67} \oplus G_{45},\ D_{27} \oplus G_{46}, D_{57} \oplus G_{47}$
(5,6)	$\{E_{16}, E_{26}, E_{36}, E_{46}\}$	$\emptyset$	$\{E_{26}\}$	$\{E_{16}, E_{36}, E_{46}\}$	$\{F_{15}, F_{56}, F_{57}\}$	$E_{16} \oplus F_{15},\ E_{36} \oplus F_{56},\ E_{46} \oplus F_{57}$
(5,7)	$\{E_{17}, E_{27}, E_{37}, E_{47}\}$	$\{E_{67}\}$	$\{E_{27}, E_{37}\}$	$\{E_{67}, E_{17}, E_{47}\}$	$\{G_{15}, G_{56}, G_{57}\}$	$E_{67} \oplus G_{15},\ E_{17} \oplus G_{56},\ E_{47} \oplus G_{57}$

TABLE III: Sets $\Omega_{i,j}$, $\Lambda_{i,j}$, $\Delta_{i,j}$, $\mathcal{F}_{i,j}$, $\mathcal{H}_{i,j}$, and the cross-level messages formed in Step 3 of the delivery phase for Example 1.

that each $i \in \mathcal{K}^h$ appears in exactly n of them. This problem can be analyzed by constructing a graph whose vertices are the elements of $\mathcal{K}^h$, and each vertex has a degree of n so that each edge of the graph can be considered as a pair. To successfully construct such a graph we utilize the *matching* concept used in graph theory. The key graph theoretic definitions and lemmas used for this analysis are introduced in Appendix A, while the construction of sets $\Delta_{i,j}$ and $\mathcal{B}$ is detailed in Appendix B.

Step 4) Intra-high-level delivery with multicasting gain of two: In the last step, the server multicasts the remaining messages which are colored with green in Table II. In Example 1, this step is finalized with unicasting the sub-file A_{46}. Let $\mathcal{B}$ denote the set of multicast messages that are not decomposed to be used for cross-level delivery, and hence, are multicasted in the last step.

The required delivery rate for a particular demand vector $\mathbf{d}$ under $\mathrm{CL}(2, 1, \mathrm{N_h}, \mathrm{N_l}, 0)$ scheme is given by the following lemma.

Lemma 1. *For a cache size of* $M = \frac{2N_h+N_l}{2}$, *and demand realization* $\mathbf{d}$, *the following delivery rate is achievable by the proposed* $\mathrm{CL}(2, 1, \mathrm{N_h}, \mathrm{N_l}, 0)$ *scheme:*

$$R(\mathbf{d}) = \begin{cases} \frac{\binom{k^h}{3}+\left\lceil\left(L(\mathbf{d})-3\binom{k^h}{3}\right)/2\right\rceil}{\binom{K}{2}}, & \text{if } k^h \geq 2 \\ \frac{K-1}{2}, & \text{otherwise} \end{cases}, \tag{10}$$

where $L(\mathbf{d})$ *is the total number of missing sub-files for demand realization* $\mathbf{d}$, *and is given by*

$$L(\mathbf{d}) \triangleq (K-1)\left[\frac{K^2}{2} - k^h - \frac{k^l}{2}\right]. \tag{11}$$

Note that, in the centralized coded delivery scheme of [1], when all the files are cached at level t, the delivery rate is equal to $\frac{K-t}{t+1}$. We can observe from (10) that, when $k^l = K$, there are only low level users and $R(\mathbf{d}) = \frac{K-1}{2}$. Similarly, when $k^h = K$, there are only high level users and $R(\mathbf{d}) = \frac{K-2}{3}$. In general, Lemma 1 implies that for any demand realization $\mathbf{d}$, sub-files are served with a minimum multicasting gain of 2. We also remark that the delivery rate depends only on the vector $\mathbf{k} \triangleq [k^h, k^l]$; rather than the demand vector; hence, the dependence on $\mathbf{d}$ in (10) and (11) can be replaced by $\mathbf{k}$, i.e., we use $R(\mathbf{k})$ and $L(\mathbf{k})$, respectively. Note that k^h and k^l are random variables, and their distribution $P(\mathbf{k}|N_h, N_l)$ depends on N_h, N_l and the popularity distribution. Accordingly, the average delivery rate can now be written as follows:

$$\bar{R} \triangleq \sum_{\mathbf{k}} R(\mathbf{k}) P(\mathbf{k}|N_h, N_l). \tag{12}$$

In the next subsection, we extend our analysis to the case in which a certain subset of the files are not cached at all, that is, $N_h + N_l < N$.

C. $\mathrm{CL}(\mathrm{t_h} = 2, \mathrm{t_l} = 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$

We recall that when $N_h + N_l = N$, the cache memory requirement for $\mathrm{CL}(2, 1, \mathrm{N_h}, \mathrm{N_l}, 0)$ is given by

$$M\,(2, 1, N_h, N_l, 0) \triangleq \frac{N_h + N}{K}. \tag{13}$$

Hence, in this particular case, for a given cache memory size of M, N_h and N_l can be directly written as a parameter of N, M, and K. However, by choosing not to cache N_r number of least popular files, it is possible to consider different values for N_h and N_l under the constraint

$$\frac{2N_h + N_l}{K} \leq M, \tag{14}$$

(i,j)	$\Omega_{i,j}$	$\Pi_{i,j}$	$\Lambda_{i,j}$	$\Delta_{i,j}$	$\mathcal{F}_{i,j}$	$\mathcal{H}_{i,j}$	Cross-level messages
(1,5)	$\{A_{25}, A_{35}, A_{45}\}$	$\{A_{57}\}$	$\emptyset$	$\{A_{25}\}$	$\{A_{57}, A_{35}, A_{45}\}$	$\{E_{12}, E_{13}, E_{14}\}$	$A_{57} \oplus E_{12},\ A_{35} \oplus E_{13},\ A_{45} \oplus E_{14}$
(1,6)	$\{A_{26}, A_{36}, A_{46}\}$	$\{A_{67}\}$	$\{A_{56}\}$	$\{A_{26}, A_{36}\}$	$\{A_{67}, A_{46}, A_{56}\}$	$\{F_{12}, F_{13}, F_{14}\}$	$A_{67} \oplus F_{12},\ A_{46} \oplus F_{13},\ A_{56} \oplus F_{14}$
(2,5)	$\{B_{15}, B_{35}, B_{45}\}$	$\{B_{57}\}$	$\emptyset$	$\{B_{15}\}$	$\{B_{57}, B_{35}, B_{45}\}$	$\{E_{23}, E_{24}, E_{25}\}$	$B_{57} \oplus E_{23},\ B_{35} \oplus E_{24},\ B_{45} \oplus E_{25}$
(2,6)	$\{B_{16}, B_{46}, B_{36}\}$	$\{B_{67}\}$	$\{B_{56}\}$	$\{B_{16}, B_{46}\}$	$\{B_{67}, B_{36}, B_{56}\}$	$\{F_{23}, F_{24}, F_{25}\}$	$B_{67} \oplus F_{23},\ B_{36} \oplus F_{24},\ B_{56} \oplus F_{25}$
(3,5)	$\{C_{45}, C_{15}, C_{25}\}$	$\{C_{57}\}$	$\emptyset$	$\{C_{45}\}$	$\{C_{57}, C_{15}, C_{25}\}$	$\{E_{34}, E_{35}, E_{36}\}$	$C_{57} \oplus E_{34},\ C_{15} \oplus E_{35},\ C_{25} \oplus E_{36}$
(3,6)	$\{C_{16}, C_{46}, C_{36}\}$	$\{C_{67}\}$	$\{C_{56}\}$	$\{C_{16}, C_{46}\}$	$\{C_{67}, C_{36}, C_{56}\}$	$\{F_{34}, F_{35}, F_{36}\}$	$C_{67} \oplus F_{34},\ C_{36} \oplus F_{35},\ C_{56} \oplus F_{36}$
(4,5)	$\{D_{35}, D_{25}, D_{15}\}$	$\{D_{57}\}$	$\emptyset$	$\{D_{35}\}$	$\{D_{57}, D_{15}, D_{25}\}$	$\{E_{45}, E_{46}, E_{47}\}$	$D_{57} \oplus E_{45},\ D_{15} \oplus E_{46},\ D_{25} \oplus E_{47}$
(4,6)	$\{D_{26}, D_{36}, D_{16}\}$	$\{D_{67}\}$	$\{D_{56}\}$	$\{D_{26}, D_{36}\}$	$\{D_{67}, D_{16}, D_{56}\}$	$\{F_{45}, F_{46}, F_{47}\}$	$D_{67} \oplus F_{45},\ D_{16} \oplus F_{46},\ D_{56} \oplus F_{47}$

TABLE IV: Sets $\Omega_{i,j}$, $\Lambda_{i,j}$, $\Delta_{i,j}$, $\mathcal{F}_{i,j}$, $\mathcal{H}_{i,j}$, $\Pi_{i,j}$ and the multicasted messages in step 3 of the delivery phase for Example 2.

or, equivalently,

$$N_h - N_r \leq KM - N. \tag{15}$$

From Equation (15), one can observe that by playing with the parameter N_r it is possible to seek a different placement strategy in terms of the caching levels. We will later discuss how the placement strategy can be optimized, but here now we will focus on the delivery of the $\mathrm{CL}(2, 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$ scheme, where we have three set of users, $\mathcal{K}^h$, $\mathcal{K}^l$ and $\mathcal{K}^r$, and set $k^a \triangleq |\mathcal{K}^a|, a \in \{l, h, r\}$ as before.

We highlight the key modifications compared to the case $N_r = 0$. First, we introduce a new set $\mathcal{Z}_{i,j}$ of sub-files that are requested by a high-level user $i \in \mathcal{K}^h$, and cached by a low-level user $j \in \mathcal{K}^l$ and a zero-level user. The fundamental modification appears in the cross-level delivery step, i.e., in Step 3 of the delivery algorithm, particularly in the construction of set $\mathcal{F}_{i,j}$. When $\mathcal{K}^r \neq \emptyset$, $\mathcal{F}_{i,j}$ is constructed in the following way

$$\mathcal{F}_{i,j} = \Lambda_{i,j} \cup \Pi_{i,j} \cup \left\{\Omega_{i,j} \setminus \Delta_{i,j}\right\}, \tag{16}$$

where $\Pi_{i,j} \subseteq \mathcal{Z}_{i,j}$. Initially, we want all the sub-files in $\mathcal{Z}_{i,j}$ to be included in $\mathcal{F}_{i,j}$, i.e., $\mathcal{Z}_{i,j} \subseteq \mathcal{F}_{i,j}$; however, this is not possible if

$$|\Lambda_{i,j}| + |\mathcal{Z}_{i,j}| > |\mathcal{H}_{i,j}|. \tag{17}$$

In that case, $\sigma_{i,j}$ sub-files must be removed from $\mathcal{Z}_{i,j}$ to obtain $\Pi_{i,j}$, where $\sigma_{i,j}$ is found as

$$\begin{aligned}
\sigma_{i,j} &\triangleq \left(|\Lambda_{i,j}| + |\mathcal{Z}_{i,j}| - |\mathcal{H}_{i,j}|\right)^+ \\
&= \left(\left\lceil \frac{k^l - 1}{2} \right\rceil + k^r - \frac{K-1}{2}\right)^+, \text{or} \\
&= \left(\left\lfloor \frac{k^l - 1}{2} \right\rfloor + k^r - \frac{K-1}{2}\right)^+,
\end{aligned} \tag{18}$$

where $(x)^+$ is x if $x > 0$, and 0 otherwise.

We remark that when $\sigma_{i,j} \geq 0$, this means that $|\Lambda_{i,j}| + |\Pi_{i,j}| = |\mathcal{H}_{i,j}|$, thus $\Omega_{i,j} = \Delta_{i,j}$ and $n_{i,j} = |\Omega_{i,j}|$. In general, $n_{i,j}|\Omega_{i,j}| - \tilde{n}_{i,j}$, where $\tilde{n}_{i,j}$

$$\begin{aligned}
\tilde{n}_{i,j} &\triangleq \left(|\mathcal{H}_{i,j}| - |\Lambda_{i,j}| - |\mathcal{Z}_{i,j}|\right)^+ \\
&= \left(\frac{K-1}{2} - \left\lceil \frac{k^l - 1}{2} \right\rceil - k^r\right)^+, \text{or} \\
&= \left(\frac{K-1}{2} - \left\lfloor \frac{k^l - 1}{2} \right\rfloor - k^r\right)^+.
\end{aligned} \tag{19}$$

When $n_{i,j}$'s are obtained, the sets $\Delta_{i,j}$ are constructed as in $\mathrm{CL}(2, 1, \mathrm{N_h}, \mathrm{N_l}, 0)$. Overall, the delivery phase of $\mathrm{CL}(2, 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$ consists of five steps. We will use the following example to illustrate these steps.

Example 2. In this example, we use the same setup as in Example 1; however, now we assume that files A, B, C, D are high-level files, E and F are low-level files, while file G is not cached at all. We also assume that users $1, 2, 3, 4, 5, 6, 7$ request files A, B, C, D, E, F, G respectively. Hence, $\mathcal{K}^h = \{1, 2, 3, 4\}$, $\mathcal{K}^l = \{5, 6\}$ and $\mathcal{K}^r = \{7\}$.

Step 1) Intra-high-level delivery: The first step of the delivery phase is identical to that in (2) for $t = 2$. The only difference is that, now we consider only the users in $\mathcal{K}^h \cup \mathcal{K}^r$, instead of $[K]$, and a multicast message is not transmitted for subsets of $\mathcal{K}^r$, i.e., $\mathcal{S} \subseteq \mathcal{K}^r$.

Step 2) Intra-low-level delivery: The second step also follows (2) with $t = 1$, targeting low-level users in $\mathcal{K}^l$. In Example 2, the messages delivered by the server corresponding to the first two steps are listed in Table V.

Step 3) Cross-level delivery: We construct sets $\Lambda_{i,j}$ as illustrated in Table IV for Example 2. One can easily observe that $\sigma_{i,j} = 0$, $\forall i \in \mathcal{K}^h$ and $\forall j \in \mathcal{K}^l$, which implies that $\Pi_{i,j} = \mathcal{Z}_{i,j}$. Then, we evaluate the value of $n_{i,j}$ for all $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$. Subsequently, we construct the sets $\Delta_{i,j}$ and $\mathcal{F}_{i,j}$ for the evaluated values of $n_{i,j}$. Eventually, sets $\mathcal{F}_{i,j}$ and $\mathcal{H}_{i,j}$ are used to construct the set of multicast messages $\mathcal{B}$ similarly to the case $\mathcal{K}^r = \emptyset$. We again refer the reader to Appendix B for further

High-level users	Multicast message
1 2 3	$A_{23} \oplus B_{13} \oplus C_{12}$
1 2 4	$A_{24} \oplus B_{14} \oplus D_{12}$
1 2	$A_{27} \oplus B_{17}$
1 3 4	$A_{34} \oplus C_{14} \oplus D_{13}$
1 3	$A_{37} \oplus C_{17}$
1 4	$A_{47} \oplus D_{17}$
2 3 4	$B_{34} \oplus C_{24} \oplus D_{23}$
2 3	$B_{37} \oplus C_{27}$
2 4	$B_{47} \oplus D_{27}$
3 4	$C_{47} \oplus D_{37}$
Low-level users	Multicast message
5 6	$E_{16} \oplus F_{15}$
5 6	$E_{26} \oplus F_{56}$
5 6	$E_{67} \oplus F_{57}$

TABLE V: Multicast messages in the first two steps of the delivery phase of $\mathrm{CL}(2,1,\mathrm{N_h},\mathrm{N_l},0)$ scheme in Example 2.

details. Sets $\Delta_{i,j}$, $\mathcal{F}_{i,j}$, and all the multicasted messages in the third step for Example 2 are given in Table IV.

Step 4) Intra-high-level delivery with multicasting gain of two: In this step, the server multicasts the messages

$$\begin{aligned}\mathcal{B} = \{&A_{25} \oplus B_{15}, C_{45} \oplus D_{35}, A_{26} \oplus B_{16}, D_{36} \oplus C_{46} \\ &A_{36} \oplus C_{16}, B_{46} \oplus D_{26}\},\end{aligned} \tag{20}$$

each of which is destined for two high-level users.

Step 5) Unicasting: The remaining sub-files are sent as unicast messages in the last step. The sub-files that are sent in this step can be categorized under three groups: the first group consists of the sub-files that are requested by zero-level users, the second group consists of the sub-files that are requested by low-level users and cached by zero-level users, finally the third group is formed by the sub-files in set $\cup_{i\in\mathcal{K}^h, j\in\mathcal{K}^l}(\mathcal{Z}_{i,j} \setminus \Pi_{i,j})$.

Let $N_T(\mathbf{k})$ be the number of transmitted messages (unicast and multicast) for a given demand realization $\mathbf{k} \triangleq [k^h, k^l, k^r]$, i.e.,

$$\begin{aligned}N_T(\mathbf{k}) = &\underbrace{\binom{k^h + k^r}{3} - \binom{k^r}{3}}_{\text{Step 1}} + \underbrace{\frac{K-1}{2}\binom{k^l}{2}}_{\text{Step 2}} + \underbrace{\frac{K-1}{2}k^h k^l}_{\text{Step 3}} \\ &+ \underbrace{\left\lceil \frac{\sum_{i,j} n_{i,j}}{2} \right\rceil}_{\text{Step 4}} + \underbrace{k^l k^r \frac{K-1}{2} + \sum_{i,j} \sigma_{i,j}}_{\text{Step 5}}.\end{aligned} \tag{21}$$

Then, the normalized delivery rate for $\mathrm{CL}(2,1,\mathrm{N_h},\mathrm{N_l},0)$ is given by $R(\mathbf{k}) = \frac{N_T(\mathbf{k})}{\binom{K}{2}} + k^r$. The average delivery rate can be written as follows

$$\bar{R}(N_h, N_l, N_r) \triangleq \sum_{\mathbf{k}} R(\mathbf{k}) P(\mathbf{k}|N_h, N_l, N_r), \tag{22}$$

where $\mathbf{k} \triangleq [k^h, k^l, k^r]$ denotes the demand realization. Hence, the placement strategy can be written as an optimization problem;

$$\begin{aligned}&\textbf{P1:} \quad \min \bar{R}(N_h, N_l, N_r) \\ &\text{subject to: (15)}\end{aligned} \tag{23}$$

To solve the optimization problem **P1**, we first need to compute the required delivery rate $R(\mathbf{k})$ for each possible demand realization $\mathbf{k}$, which scales with $O(K^2)$. The cache memory constraint in (15) can be rewritten in the following way;

$$N_r^{max} \geq N_r \geq N_r^{min} \tag{24}$$

where N_r^{max} and N_r^{min} are defined as the number of files that are not cached in order to cache all the remaining files at level 2 and 1, respectively. Hence, solving the optimization problem **P1** is equivalent to searching for N_r, that gives the minimum $\bar{R}(N_h, N_l, N_r)$. Hence, for each value of N_r, $P(\mathbf{k}|N_h, N_l, N_r)$ must be calculated for each possible realization of $\mathbf{k}$, which scales with $O(K^2)$. Hence, the optimization of the placement phase has an overall complexity of $O(NK^2 + K^2)$.

IV. Extension to CL($t_h, t_l = 1, N_h, N_l, N_r$)

In this section, we extend our analysis to the more general CLCD scheme where t_h can take any integer value, although we still fix $t_h = 1$. We emphasize here that when $t_h = 2$, it is possible to establish a link between the process of constructing multicast messages, particularly forming of the pair of sub-files, and the concept of *matchings* from graph theory. However, this correspondence does not generalize with $t_h = 2$, we particularly focus on a multicasting gain of two, whereas in this section, our aim is provide a more general perspective on CLCD and show how the delivery phase of the proposed scheme can be analyzed as an optimization problem.

The placement phase of the CL($t_h, 1, N_h, N_l, N_r$) scheme is similar to that of CL($2, 1, N_h, N_l, N_r$). Each file is divided into $\binom{K}{t_h}$ sub-files, and the high-level files are cached at level t_h; that is each user caches $\binom{K-1}{t_h-1}$ sub-files of each high-level file. On the other hand, for the low-level files, the sub-files are grouped into K disjoint and equal-size subsets and each user caches the sub-files in a different subset which corresponds to caching at level 1.

Now, to analyze the delivery phase with cross-level coded delivery, let $\mathcal{P}(\mathcal{K})$ be the power set of $\mathcal{K}$, and $\mathbb{S} \in \mathcal{P}(\mathcal{K})$ be the set of sets $\mathcal{S}$ with cardinality $t+1$, i.e., $|\mathbb{S}| = \binom{K}{t+1}$, and $\tilde{\mathbb{S}} \subseteq \mathbb{S}$ defined as $\tilde{\mathbb{S}} = \left\{\mathcal{S} \in \mathbb{S} : \mathcal{K}^h \cap \mathcal{S} \neq 0\right\}$. Recall that, with the conventional coded delivery scheme, the low-level and high-level users are served separately. Hence, for the delivery of the high-level files, for each set $\mathcal{S} \subseteq \tilde{\mathbb{S}}$ the server transmits

$$\oplus_{i \in \mathcal{K}^h \cap \mathcal{S}} W_{d_i, \mathcal{S}\setminus\{i\}} \tag{25}$$

in order to deliver all the sub-files requested by the high-level users. However, although the high-level files are cached at level t, a message destined for high-level users has a multicasting gain of $\mu_{\mathcal{S}} = |\mathcal{K}^h \cap \mathcal{S}|$, instead of $t+1$. In particular, when $\mu_{\mathcal{S}} = 1$ the corresponding message is a unicast message.

We call a multicast message in the form of (25) as *decomposable* if in the corresponding set $\mathcal{S}$, there is at least one high-level and one low-level user. The server can utilize those high-level sub-files for cross-level delivery. Let $\mathbb{S}_d$ be the set of sets $\mathcal{S}$ that correspond to a decomposable multicast message. In the CL($t_h, 1, N_h, N_l, N_r$) scheme, for each demand realization $\mathbf{k}$, we look for a set $\mathbb{S}_{cl} \subseteq \mathbb{S}_d$ to use for pairing sub-files of high-level users and sub-files of low-level users. We introduce variable $x_{\mathcal{S}} \in \{1, 0\}$, where $x_{\mathcal{S}} = 0$ if the conventional coded delivery scheme, as given in (25), is used to deliver the sub-files corresponding to set $\mathcal{S}$, while $x_{\mathcal{S}} = 1$ if the corresponding multicast message is decomposed and the sub-files in the message are delivered by cross-level coded delivery. We further introduce variables $x_{(\mathcal{S},i,j)}$, for $i \in K^h \cap \mathcal{S}$ and $j \in K^l \cap \mathcal{S}$, where $x_{(\mathcal{S},i,j)} = 1$, if sub-file $W_{d_i,\mathcal{S}\setminus\{i\}}$ is paired with a sub-file $W_{d_j,\{i,***\}}$ that is requested by low-level user j and cached by high-level user i. Then, the server multicasts $W_{d_i,\mathcal{S}\setminus\{i\}} \oplus W_{d_j,\{i,***\}}$. Hence, for each set $\mathcal{S}$ there are $\mu_{\mathcal{S}} \times |\mathcal{K}^l \cap \mathcal{S}|$ variables in the form $x_{(\mathcal{S},i,j)}$.

To illustrate the multicast message decomposition process, consider the CL($t_h = 3, 1, N_h, N_l, N_r$) scheme, and assume that there are $K = 7$ users with a demand realization $\mathbf{k}$, $\mathcal{K}^h = \{1, 2, 3, 4\}$ and $\mathcal{K}^l = \{5, 6, 7\}$. Consider a particular set $\mathcal{S} = \{1, 2, 6, 7\}$. According to (25), the server multicasts $W_{d_1,\{2,6,7\}} \oplus W_{d_2,\{1,6,7\}}$ for high-level users 1 and 2. With a multicast message decomposition specified by $x_{(\mathcal{S},1,6)} = 1$, $x_{(\mathcal{S},1,7)} = 0$, $x_{(\mathcal{S},2,6)} = 0$, $x_{(\mathcal{S},2,7)} = 1$, and $x_{\mathcal{S}} = 1$, the server multicasts the messages $W_{d_6,\{1,***\}} \oplus W_{d_1,\{2,6,7\}}$ and $W_{d_7,\{2,***\}} \oplus W_{d_2,\{1,6,7\}}$ in the cross-level delivery phase. We remark that, if the conventional coded delivery scheme is used, sub-files $W_{d_6,\{1,***\}}$ and $W_{d_7,\{2,***\}}$ are sent as unicast messages. Hence, in total three messages would be transmitted with the conventional coded delivery scheme, i.e., $W_{d_1,\{2,6,7\}} \oplus W_{d_2,\{1,6,7\}}$, $W_{d_6,\{1,***\}}$, and $W_{d_7,\{2,***\}}$; however, via CLCD this can be reduced to two, i.e., $W_{d_6,\{1,***\}} \oplus W_{d_1,\{2,6,7\}}$ and $W_{d_7,\{2,***\}} \oplus W_{d_2,\{1,6,7\}}$. One can observe that if a multicast message is decomposed and used by the CLCD scheme, then the number of multicast messages is reduced by one.

Note that variables $x_{(\mathcal{S},i,j)}$ and $x_{(\mathcal{S},i,k)}$ cannot be 1 at the same time, since the sub-file $W_{d_i,\mathcal{S}\setminus\{i\}}$ can be paired with only one low-level sub-file. Hence, for the message decomposition we have the following constraint

$$\sum_{j \in \mathcal{K}^l \cap \mathcal{S}} x_{(\mathcal{S},i,j)} \leq 1, \quad \forall i \in \mathcal{K}^h \cap \mathcal{S} \text{ and } \forall \mathcal{S} \in \mathbb{S}_d. \tag{26}$$

Another constraint, from the construction, is that each high-level user i contains $\binom{K-1}{t-1}/t$ sub-files that are missing at low-level user j; thus, the total number of sub-file pairings between a low-level user j and a high-level user i is at most $\binom{K-1}{t-1}/t$ i.e.,

$$\sum_{S \in \mathbb{S}_d} x_{(\mathcal{S},i,j)} \leq \frac{\binom{K-1}{t-1}}{t} \quad \forall i \in \mathcal{K}^h, j \in \mathcal{K}^l. \tag{27}$$

Finally, if $x_S = 1$, then all the corresponding sub-files must be sent via CLCD, i.e.,

$$\sum_{i \in \mathcal{K}^h \cap \mathcal{S}, j \in \mathcal{K}^l \cap \mathcal{S}} x_{(\mathcal{S},i,j)} = \mu_{\mathcal{S}} x_S, \quad \forall \mathcal{S} \in \mathbb{S}_d. \tag{28}$$

For the given values of the variables $x_{(\mathcal{S},i,j)}$ and $x_{\mathcal{S}}$, the delivery phase of CL($t_1, 1, N_h, N_l, N_r$) is given in Algorithm 2. According to Algorithm 2, the number of transmitted messages, $N_T(\mathbf{k})$, for a given demand realization $\mathbf{k} = \left[k^h, k^l, k^r\right]$ is

Algorithm 2: Overall Delivery phase of $\mathrm{CL}(\mathrm{t_l}, 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$

```
for S ∈ S̃ \ S_cl do
    multicast ⊕_{i∈K^h∩S} W_{d_i,S\{i}};
end
for S ∈ S_cl do
    for i ∈ K^h ∩ S do
        for j ∈ K^l ∩ S do
            if x_(S,i,j) = 1 then
                multicast W_{d_i,S\{i}} ⊕ W_{d_j,{i,***}};
            end
        end
    end
end
for j, k ∈ K^l do
    send W_{d_k,{j,***}} ⊕ W_{d_j,{k,***}};
end
Remaining sub-packets are send via unicast transmission;
```

given by

$$N_T(\mathbf{k}) = \binom{K}{t+1} - \binom{k^r + k^l}{t+1} - \sum_{\mathcal{S}\in\mathbb{S}_d} x_{\mathcal{S}} \tag{29}$$

$$+ \frac{\binom{K-1}{t-1}}{t} k^h k^l + \frac{\binom{K-1}{t-1}}{t} k^r k^l + \binom{k^l}{2}\frac{\binom{K-1}{t-1}}{t}. \tag{30}$$

Hence, the main objective of $\mathrm{CL}(\mathrm{t_h}, \mathrm{t_l} = 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$ scheme is to maximize $\sum_{\mathcal{S}\in\mathbb{S}_d} x_{\mathcal{S}}$ in order to minimize the delivery rate for each possible demand realization. Equivalently, the minimum delivery rate problem can be converted to the following optimization problem:

$$\textbf{P2:} \quad \max \sum_{\mathcal{S}\in\mathbb{S}_d} x_{\mathcal{S}}$$

$$\text{subject to:} \sum_{j\in\mathcal{K}^l\cap\mathcal{S}} x_{(\mathcal{S},i,j)} \le 1, \quad \forall \mathcal{S} \in \mathbb{S}_d, \tag{31}$$

$$\sum_{\mathcal{S}\in\mathbb{S}_d} x_{(\mathcal{S},i,j)} \le \binom{K-1}{t-1}/t, \quad \forall i \in \mathcal{K}^h, j \in \mathcal{K}^l, \tag{32}$$

$$\sum_{i\in\mathcal{K}^h\cap\mathcal{S}, \forall j\in\mathcal{K}^l\cap\mathcal{S}} x_{(\mathcal{S},i,j)} = \mu_{\mathcal{S}} x_{\mathcal{S}}, \quad \forall \mathcal{S} \in \mathbb{S}_d. \tag{33}$$

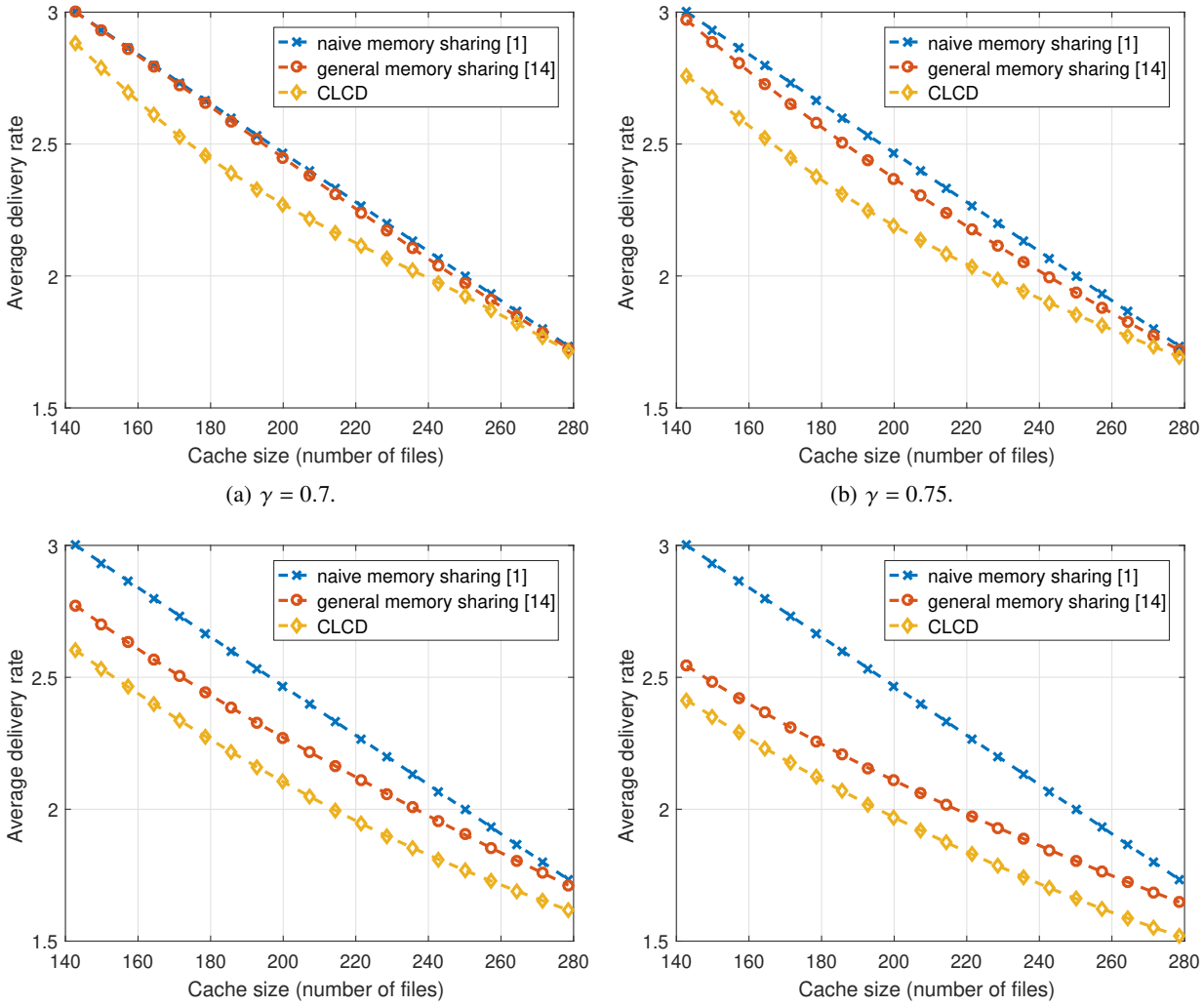
By solving **P2** for each demand realization $\mathbf{k}$, we can construct the optimal coded delivery scheme for $\mathrm{CL}(\mathrm{t_h}, 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$. We remark that **P2** is a *binary integer programming* problem, and unlike $\mathrm{CL}(2, 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$, the optimization of the delivery phase cannot be solved in polynomial time. Nevertheless, complexity of **P2** does not depend on the number of files, N, but depends on the number of users, K; hence, for moderate number of users the CLCD scheme can be optimized even for a very large file library. Once the required delivery rate for each possible demand realization is computed, optimal placement strategy; that is, the number of files not to cache, can be found following the same procedure in Section III. We want to remark that instead of solving **P2**, it is also possible to design a greedy algorithm to construct the multicast messages.

In Section VI, we will study the most general scheme $\mathrm{CL}(\mathrm{t_h}, \mathrm{t_l}, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$ and show that the delivery phase can be still analyzed as an optimization problem with a reduced computational complexity. But before extending our analysis, we would like to present the numerical results comparing the CLCD schemes presented so far with the alternative approaches in the literature.

V. Numerical Results

In this section, we compare the performance of the proposed CLCD scheme with that of the conventional centralized coded delivery scheme with two different content placement strategies. The first content placement strategy is called *naive memory sharing*, introduced in [1], in which all the files are cached at the same level according to a single parameter $t = MK/N$. Note that, when parameter t is not an integer, then the files in the library are divided into two disjoint fragments identically, and these fragments are cached according to parameters $\lfloor t \rfloor$ and $\lceil t \rceil$.

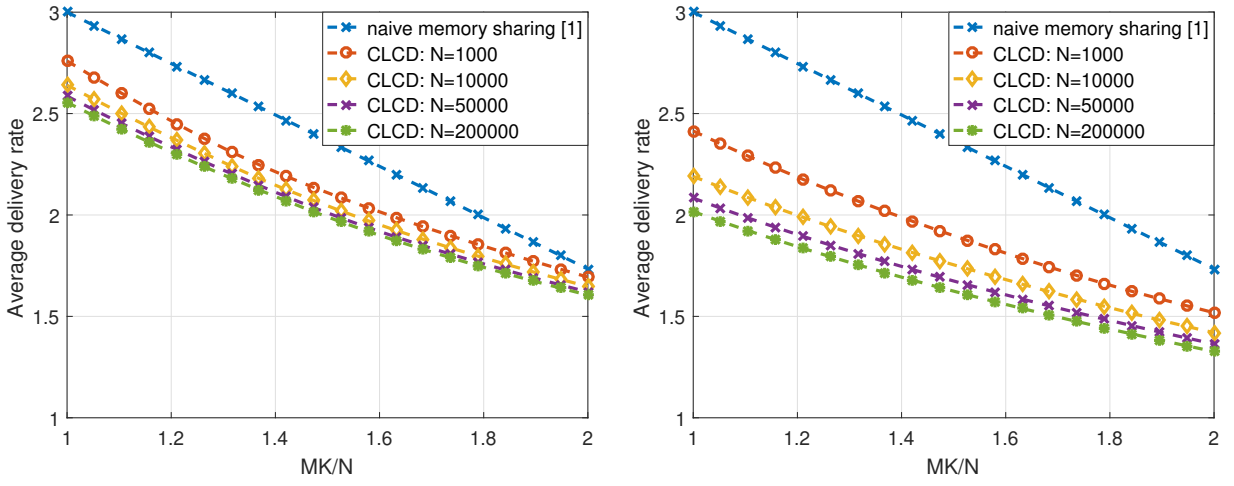
The second benchmark strategy is the *general memory sharing* scheme proposed in [14], which is shown to outperform other coded delivery techniques under non-uniform demand distributions. In this scheme, each file is divided into $K+1$ disjoint fragments, and the kth fragment, $1 < k$, is cached according to parameter $t = k-1$, while the first fragment is not cached. Thus, the overall system can be considered as a combination of $K+1$ sequential coded delivery phases with different multicasting



(a) $\gamma = 0.7$. (b) $\gamma = 0.75$.

(c) $\gamma = 0.8$. (d) $\gamma = 0.85$.

Fig. 1: Average delivery rate vs. the normalized cache size M for different Zipf parameter (γ) values.



(a) $\gamma = 0.75$. (b) $\gamma = 0.85$.

Fig. 2: Average delivery rate vs. MK/N for different Zipf parameter (γ) values.

Algorithm 3: $\mathrm{CL}(\mathrm{t_h}, \mathrm{t_l}, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r} = 0)$ scheme

```
Input : K^h, K^l
for all S ⊆ K : |S| = t_h + 1 do
    if S ⊆ K^h then
        serve high-level users with multicasting gain t_h + 1
    end
    if S ⊄ K^h and S ⊄ K^l then
        for all Ŝ ⊆ S : |Ŝ| = t_l + 1 and Ŝ ⊄ K^l do
            use CLCD with multicasting gain of t_l + 1
        end
    end
end
for all S ⊆ K^l : |S| = t_l + 1 do
    serve low-level users with multicasting gain of t_l + 1
end
```

gains, i.e., the kth delivery phase is executed with multicasting gain of $k+1$. The size of each fragment of each file can be obtained via solving a linear optimization problem [14].

In general, it has been observed that the popularity of video files for on-demand streaming applications approximately follows a Zipf distribution with parameter $1 > \gamma > 0.65$ [16], [17]. In our simulations, we consider $\gamma = 0.7, 0.75, 0.8, 0.85$. We note that γ, represents the skewness of the distribution of video popularity, where smaller γ values indicates a more uniform popularity distribution. In realistic scenarios, number of files in the video library is considered to be on the order of 10^4. However, due to the complexity of the general memory sharing scheme, we will consider $N = 1000$ and $K = 7$ in our initial simulations. We let the cache size M vary from 140 to 280, which corresponds to $1 < t < 2$. We are particularly interested in this regime as all three strategies considered in this paper converge to the same performance for larger t values. The delivery rates achieved by the naive memory sharing, the general memory sharing, and the CLCD schemes are illustrated in Figure 1. We note that, for each value of the Zipf parameter γ we calculate the achievable delivery rate of the scheme $\mathrm{CL}(\mathrm{t_h}, 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$, for $t_h = 2, 3, \ldots, 6$, and take the minimum of them.

We observe that when the number of files N is large, the general memory sharing algorithm tends to divide the library into two groups according popularities, such that the most popular files are cached according to the naive memory sharing scheme while the less popular files are not cached at all. One can observe from Fig. 1 that when $\gamma = 0.7$ or $\gamma = 0.75$ the optimal memory sharing scheme performs very similarly to the naive memory sharing scheme, while the proposed CLCD scheme can provide a significant reduction in the achievable delivery rate. In addition, we observe that CLCD outperforms both the naiive and general memory sharing schemes at all γ values considered here.

We also perform simulations to illustrate how the number of files N affects the performance of the CLCD scheme when $t = MK/N$ is fixed, i.e., the cache memory size scales with the number of files in the library. In the naive memory sharing scheme, all the files are cached at level $t = MK/N$, thus the performance of this scheme does not change with the size of the file library as long as M/N is fixed. In the simulations, we consider $N = 10^3, 10^4, 5\times10^4$, and 2×10^5. Due to its complexity it is not possible to run the general memory sharing scheme for large libraries, thus we compare CLCD scheme only with the naive memory sharing scheme. We observe in Fig. 2 that the CLCD scheme performs better for larger file libraries. In particular, when $\gamma = 0.85$ the performance gap between the CLCD scheme and the naive memory sharing scheme for $N = 2 \times 10^5$ is almost two times of the performance gap for $N = 10^3$.

VI. General $\mathrm{CL}(\mathrm{t_h}, \mathrm{t_l}, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$ Scheme

Having shown the superiority of the proposed $\mathrm{CL}(\mathrm{t_h}, 1, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$ scheme to its alternatives in the literature, in this section, we present the most generic form of the CLCD scheme. In the previously introduced versions with $t_l = 1$, the placement phase is designed to prevent any miss-alignment between the high-level and low-level sub-file sizes. Hence, the fundamental modification in the $\mathrm{CL}(\mathrm{t_h}, \mathrm{t_l}, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r})$ scheme with respect to the previous versions is that the alignment of the sub-file sizes and the construction of the multicast messages are done jointly during the deliver phase. In this section, for sake of clarity, we start our presentation by setting $N_r = 0$. This constraint will later be removed. The general structure of the $\mathrm{CL}(\mathrm{t_h}, \mathrm{t_l}, \mathrm{N_h}, \mathrm{N_l}, 0)$ scheme is given in Algorithm 3.

Here, the main design issue is how to overlap the bits of high-level and low-level files that are delivered in the CLCD phase. The key point here is that, for a subset $\mathcal{S}$ of different types of users, i.e., $\mathcal{S} \not\subseteq \mathcal{K}^h$ and $\mathcal{S} \not\subseteq \mathcal{K}^l$ with $|\mathcal{S}| = t_h + 1$, any high level user $i \in \mathcal{S}$ appears in exactly $\binom{t_h}{t_l}$ of the subsets $\hat{\mathcal{S}}$ of $\mathcal{S}$ with $|\hat{\mathcal{S}}| = t_l + 1$. Hence, the requested sub-file $W_{d_i,\mathcal{S}\setminus\{i\}}$ should be divided into $\binom{t_h}{t_l}$ smaller sub-files. Similarly, from the perspective of low-level users, each $\hat{\mathcal{S}}$ appears as a subset of exactly $\binom{K-(t_l+1)}{t_h-t_l}$ different $\mathcal{S}$ sets, thus the sub-file requested by low-level user j, $W_{d_j,\hat{\mathcal{S}}\setminus\{j\}}$, should be divided into $\binom{K-(t_l+1)}{t_h-t_l}$ smaller sub-files. Let F^h_{cl} and F^l_{cl} be the sizes of the sub-files used for the aforementioned high-level and low-level sub-files,

respectively, in the cross-delivery phase, i.e.,

$$F_{cl}^{h} = \left(\binom{K}{t_h} \binom{t_h}{t_l} \right)^{-1}, \tag{34}$$

and

$$F_{cl}^{l} = \left(\binom{K}{t_l} \binom{K-(t_l+1)}{t_h-t_l} \right)^{-1}. \tag{35}$$

By expanding (34) and (35) one can observe that

$$F_{cl}^{l} = F_{cl}^{h} \times \frac{K-t_l}{K-t_h}, \tag{36}$$

which implies that it is possible to overlap the high-level files with the low-level files in the cross-delivery step as long as there is at least one low-level user in $\hat{\mathcal{S}}$. Accordingly, the overall rate, for given k^h, k^l can be written as

$$\begin{aligned} R(k^h, k^l) = & \binom{k^h}{t_h+1} F_h + \left(\binom{K}{t_l+1} - \binom{k^h}{t_l+1} \right) F_l \\ & + \underbrace{\binom{k^h}{t_l+1} \left(\binom{K-(t_l+1)}{t_h-t_l} - \binom{k^h-(t_l+1)}{t_h-t_l} \right) F_{cl}^{h}}_{R_{no}\text{: non-overlapped bits of high level files}}. \end{aligned} \tag{37}$$

Here, the CLCD scheme has two objectives: one is to achieve a multicasting gain of at least t_l+1 in the delivery phase, and the other is to overlap the delivery of the high-level and low-level files as much as possible. Hence, we want to minimize the term R_{no} in (37). This can be achieved by dividing the high-level files in a non-uniform manner. To clarify the non-uniform file division, consider the case with $t_h = 4$, $t_l = 3$, and assume that there are $K = 7$ users with a demand realization $\mathbf{k} = [4, 3, 0]$, where $\mathcal{K}^h = \{1, 2, 3, 4\}$ and $\mathcal{K}^l = \{5, 6, 7\}$.

According to the proposed CLCD design, for set $\mathcal{S} = \{1, 2, 3, 4, 5\}$, delivery of the requested sub-files are realized over 5 steps, with a multicasting gain of $t_l + 1 = 4$, corresponding to the subsets $\hat{\mathcal{S}}_1 = \{1, 2, 3, 4\}$, $\hat{\mathcal{S}}_2 = \{1, 2, 3, 5\}$, $\hat{\mathcal{S}}_3 = \{1, 2, 4, 5\}$, $\hat{\mathcal{S}}_4 = \{1, 3, 4, 5\}$, $\hat{\mathcal{S}}_5 = \{2, 3, 4, 5\}$. We recall that since user 1 receives its required file $W_{d_1,\{2,3,4,5\}}$, part by part in four of these steps, the corresponding sub-file is further divided into 4 smaller sub-files. Let $W_{d_1,\{2,3,4,5\},\hat{\mathcal{S}}_j}$ denote the one delivered in the step corresponding to subset $\hat{\mathcal{S}}_j$. Subset $\hat{\mathcal{S}}_1 = \{1, 2, 3, 4\}$ consists of only high-level users. We also recall that, as given in (36), the low-level sub-files are larger than the high-level ones. Hence, to reduce the amount of non overlapping bits R_{no}, one can reduce the size of $W_{d_1,\{2,3,4,5\},\hat{\mathcal{S}}_1}$ and, accordingly, increase the size of $W_{d_1,\{2,3,4,5\},\hat{\mathcal{S}}_2}$, $W_{d_1,\{2,3,4,5\},\hat{\mathcal{S}}_3}$,$W_{d_1,\{2,3,4,5\},\hat{\mathcal{S}}_4}$ until they match with that of F_{cl}^{l}.

The main question arising from this observation is how to adjust the size of the high-level sub-files assigned to each $\hat{\mathcal{S}}$ so that the high-level and low-level sub-files overlap as much as possible. Let us focus on a particular subset $\mathcal{S} \nsubseteq \mathcal{K}^h$, $\mathcal{S} \nsubseteq \mathcal{K}^l$ and a particular high-level user $i \in \mathcal{S}$. Let the number of subsets $\hat{\mathcal{S}} : i \in \hat{\mathcal{S}} \subseteq \mathcal{K}^h$, be denoted by $n_h^i(\mathcal{S})$, and the number of subsets $\hat{\mathcal{S}} : i \in \hat{\mathcal{S}} \nsubseteq \mathcal{K}^h$ be denoted by $n_{cl}^i(\mathcal{S})$. Since we want to reduce the size of sub-files $W_{d_i,\mathcal{S}\setminus\{i\},\hat{\mathcal{S}}}$, $i \in \hat{\mathcal{S}} \subseteq \mathcal{K}^h$, and increase that of sub-files $W_{d_i,\mathcal{S}\setminus\{i\},\hat{\mathcal{S}}}$, $i \in \hat{\mathcal{S}} \nsubseteq \mathcal{K}^h$, we have the following constraint based on (36):

$$\left(\frac{n_h^i(\mathcal{S})\alpha_i(\mathcal{S})}{n_{cl}^i(\mathcal{S})} + 1 \right) F_{cl}^{h} = F_{cl}^{h} \frac{K-t_l}{K-t_h}, \tag{38}$$

where $\alpha_i(\mathcal{S})$ denotes the ratio of reduction in the size of the high-level sub-files. Accordingly, $\alpha_i(\mathcal{S})$ parameter can be written as

$$\alpha_i(\mathcal{S}) = \frac{(t_h-t_l) n_{cl}^i(\mathcal{S})}{(K-t_h) n_h^i(\mathcal{S})}. \tag{39}$$

We note that, parameters $n_{cl}^i(\mathcal{S})$, $n_h^i(\mathcal{S})$ and $\alpha_i(\mathcal{S})$ are identical for all the high-level users $i \in \mathcal{S}$; hence, we drop the user index for simplicity. Once the size of the high-level sub-files are adjusted, R_{no} can be rewritten as follows:

$$R_{no} = \sum_{\mathcal{S}\subseteq\mathcal{K}:\ \mathcal{S}\cap\mathcal{K}^h\neq\emptyset,\ \mathcal{S}\cap\mathcal{K}^l\neq\emptyset} n_h(\mathcal{S}) \min(1-\alpha(\mathcal{S}), 0) F_{cl}^{h}. \tag{40}$$

Consequently, by inserting R_{no} above into (37), one can obtain the required delivery rate for Algorithm 3.

We remark that with the CLCD scheme all the sub-files are served with a minimum multicasting gain of $t_l + 1$. If $R_{no} = 0$, one can claim the optimality of the delivery phase, under the same placement scheme, since the low-level files cannot be served with a multicasting gain greater than t_l+1. On the other hand, if $R_{no} > 0$, then the non-overlapped bits of the high-level sub-files are delivered with a multicasting gain of t_l+1, although, in principle, they could be delivered with a multicasting gain of $t + 1$, where $t_h > t > t_l$; thus, when $t_h >> t_l$ this approach might be inefficient. Hence, the main question at this point is

Algorithm 4: CL($\mathrm{t_h}, \mathrm{t_l}, \mathrm{N_h}, \mathrm{N_l}, \mathrm{N_r} = 0$) scheme with threshold t_{th}

Input : $\mathcal{K}^h$, $\mathcal{K}^l$, t_{th}
Construct set $\mathbb{S}_{cl} = \left\{\mathcal{S} : |\mathcal{S} \cap \mathcal{K}^h| \leq t_{th} + 1, \quad \mathcal{S} \nsubseteq \mathcal{K}^l\right\}$;
Based on $\mathbb{S}_{cl}$ compute initial values of F^l_{cl} and F^h_{cl};
for *all* $\mathcal{S} \subseteq \mathbb{S}_{cl}$ **do**
  Compute $n_h(\mathcal{S})$, $n_l(\mathcal{S})$, $\alpha(\mathcal{S})$ to seek maximum overlapping
end
for *all* $\mathcal{S} \subseteq \mathcal{K} : |\mathcal{S}| = t_h + 1$ **do**
  if $|\mathcal{S} \cap \mathcal{K}^h| > t_{th} + 1$ **then**
    serve high-level users with multicasting gain $|\mathcal{S} \cap \mathcal{K}^h|$
  end
  if $\mathcal{S} \subseteq \mathbb{S}_{cl}$ **then**
    for *all* $\hat{\mathcal{S}} \subseteq \mathcal{S} : |\hat{\mathcal{S}}| = t_l + 1$ *and* $\hat{\mathcal{S}} \nsubseteq \mathcal{K}^l$ **do**
      use CLCD with multicasting gain $t_l + 1$
    end
  end
end
for *all* $\mathcal{S} \subseteq \mathcal{K}^l : |\mathcal{S}| = t_l + 1$ **do**
  serve low-level users with multicasting gain of $t_l + 1$
end

how to decide which $\mathcal{S}$ subsets with $|\mathcal{S}| = t_h + 1$ will be used for CLCD. To this end, we introduce another parameter, which denoted by t_{th}, where $t_h > t_{th} \geq t_l$, such that $\mathcal{S} : |\mathcal{S}| = t_h + 1$ is used for CLCD if $|\mathcal{S} \cap \mathcal{K}^h| \leq t_{th} + 1$, otherwise high-level users in $\mathcal{S}$ are served with multicasting gain of $|\mathcal{S} \cap \mathcal{K}^h|$ as in Algorithm 3.

The CLCD scheme with t_{th} is given in Algorithm 4. The delivery rate, $R(\mathbf{k})$, for given demand realization $\mathbf{k}$ with Algorithm 4 can be written as,

$$R(\mathbf{k}) = \sum_{\mathcal{S}\subseteq\mathcal{K}:\ |\mathcal{S}\cap\mathcal{K}^h|>t_{th}+1} F_h + \left(\binom{K}{t_l+1} - \binom{k^h}{t_l+1}\right) F_l + \sum_{\mathcal{S}\subseteq\mathbb{S}_{cl}} n_h(\mathcal{S}) \min(1-\alpha(\mathcal{S}), 0) F^h_{cl}. \tag{41}$$

The second term on the right hand side of (41) does not depend on parameter t_{th}; whereas the first term decreases with t_{th}, while the last term increases, which implies that t_{th} seeks a balance between the first and the last terms. Therefore, for each $\mathbf{k}$, a different t_{th} value may minimize $R(\mathbf{k})$. The average delivery rate is given by

$$\bar{R} = \sum_{\mathbf{k}} P(\mathbf{k}) R(\mathbf{k}). \tag{42}$$

In general, given t_h and t_l, one should find the best t_{th} value for each $\mathbf{k}$.

In the most generic form of the CLCD scheme we also allow $N_r > 0$. We slightly modify the delivery scheme by simply considering the zero-level users (users with uncached file requests) as low-level users for cross-level delivery and serve them with unicast transmissions at the end. Accordingly, the delivery rate is given by as

$$R(\mathbf{k}) = \sum_{\mathcal{S}\subseteq\mathcal{K}:\ |\mathcal{S}\cap\mathcal{K}^h|>t_{th}+1} F_h + \left(\binom{K}{t_l+1} - \binom{k^h}{t_l+1} - \binom{k^r}{t_l+1}\right) F_l + \sum_{\mathcal{S}\subseteq\mathbb{S}_{cl}} n_h(\mathcal{S}) \min(1-\alpha(\mathcal{S}), 0) F^h_{cl} + k^r. \tag{43}$$

We note that it might be possible to further reduce the delivery rate by revisiting the sub-file size alignment strategy taking into account the zero-level users at the expense of additional complexity.

VII. Conclusions

We introduced a novel centralized coded caching delivery scheme, called CLCD, for cache-aided content delivery under non-uniform demand distributions. The proposed caching scheme uses a different placement strategy for the files depending on their popularities, such that the cache memory allocated to the sub-files belonging to more popular files are larger compared to the low popular ones. The main novelty of the proposed approach is in the delivery phase, where the messages multicasted to the group of users are carefully designed to satisfy as many users as possible with minimal delivery rate while in the case where the files are cached in a non-uniform manner based on their popularities. For a certain special case of the proposed algorithm we are able to provide a closed form expression for the delivery rate. We also showed via numerical simulations

that the proposed CLCD scheme can provide up to 10% reduction in the average delivery rate compared to the state-of-the-art. We expect this gain to grow considerably as the number of files increases, while the state-of-the-art schemes in the literature cannot be implemented for a large file library due to their formidable complexity.

APPENDIX A
SET PARTITIONING

In this section, we will introduce some useful lemmas and definitions related to graph theory that will be utilized in the process of multicast message construction, particularly for pairing the sub-files. We use G, $\mathcal{V}(G)$ and $\mathcal{E}(G)$ to denote a graph, its set of vertices and edges, respectively. For simplicity, we enumerate the vertices, and set $\mathcal{V}(G) = [V]$, so that V represents the number of vertices.

Definition 1. *A graph G is a complete graph if each pair of distinct vertices is connected by a unique edge.*

Definition 2. *A matching D of graph G is a subgraph of G whose edges share no vertex; that is, each vertex in matching D has degree one.*

Definition 3. *A matching is maximum when it has the largest possible size.*

Definition 4. *A matching of a graph G is perfect if it contains all of G's vertices. From definition a perfect matching is a maximum matching although reverse is not necessarily true.*

Definition 5. *1-factorization of a graph G is a collection $\mathcal{D}$ of edge-disjoint perfect matchings (also referred to as 1-factors) whose union is $\mathcal{E}(G)$. Graph G is 1-factorable if it admits 1-factorization.*

Lemma 2. *A complete graph G, with $V = 2k$ for some $k \in \mathbb{Z}^+$, is 1-factorable and $|\mathcal{D}| = V$*

Definition 6. *For a given edge $e_{i,j}$, $I(e_{i,j})$ is the index set of adjacent vertices $I(e_{i,j}) = \{i, j\}$.*

Definition 7. *A* partition *of a set $\mathcal{S}$ is a collection of nonempty and mutually disjoint subsets of $\mathcal{S}$, called blocks, whose union is $\mathcal{S}$. For instance given set $\mathcal{S} = \{1, 2, 3, 4\}$; $\{1\}$,$\{2, 3, 4\}$ is a partition of $\mathcal{S}$ with blocks of sizes 1 and 3.*

Lemma 3. *Consider a graph G with $V = 2k$ for some $k \in \mathbb{Z}^+$. Then, for any perfect matching D of G, $\mathcal{P} \triangleq \left\{I(e_{i,j}) : e_{i,j} \in \mathcal{E}(D)\right\}$ is a partition of set $[V]$ with blocks of size two.*

Lemma 4. *Given set $[V]$, with $V = 2k$ for some $k \in \mathbb{Z}^+$, it is possible to obtain V different disjoint partitions[3] of set $[V]$ with blocks of size two.*

Proof. Consider a complete graph G, such that $\mathcal{V}(G) = [V]$ with $V = 2k$. From Lemma 2, $\mathcal{E}(G)$ can be written as a collection $\mathcal{D}$ of edge-disjoint perfect matchings. Lemma 3 implies that for any $D \in \mathcal{D}$, one can obtain $\mathcal{P} \triangleq \left\{I(e_{i,j}) : e_{i,j} \in D\right\}$ which is a partition of set $[V]$ with blocks of size two. Further, since any $D_k, D_l \in \mathcal{D}$ edge-disjoint perfect matching, corresponding partitions, $\mathcal{P}_k \triangleq \left\{I(e_{i,j}) : e_{i,j} \in \mathcal{E}(D_k)\right\}$ and $\mathcal{P}_l \triangleq \left\{I(e_{i,j}) : e_{i,j} \in \mathcal{E}(D_l)\right\}$ are disjoint i.e., $\mathcal{P}_k \cap \mathcal{P}_l = \emptyset$. □

In Fig. 3, all disjoint partitions for set $[6] = \{1, 2, 3, 4, 5, 6\}$, and the corresponding edge-disjoint perfect matchings are illustrated. We note that, although Lemma 4 is introduced for set $[V]$, with $V = 2k$ for some $k \in \mathbb{Z}^+$, it is valid for any set $\mathcal{K}$, $|\mathcal{K}| = V$, and we use the notation $\mathcal{P}^{\mathcal{K}}$ to refer a particular set $\mathcal{K}$.

Lemma 5. *Consider a complete graph G with $V = 2k + 1$ for some $k \in \mathbb{Z}^+$. Then, for any maximum matching D of G, $\mathcal{P} \triangleq \left\{I(e_{i,j}) : e_{i,j} \in \mathcal{E}(D)\right\}$ is a partition of set $[V] \setminus \{v\}$ with blocks of size two, where $v \in \mathcal{V}(G) \setminus \mathcal{V}(D)$.*

Lemma 6. *A complete graph G, with $V = 2k + 1$ for some $k \in \mathbb{Z}^+$, can be decomposed into collection of V edge-disjoint maximum matchings $\mathcal{D}$.*

Proof. Assume that all the vertices in $\mathcal{V}(G)$ are located on a circle according to their index with an increasing order. Then, for any vertex $j \in \mathcal{V}(G)$, a sub-graph $\tilde{G}$ with $\mathcal{V}(\tilde{G}) = [V]\setminus\{j\}$ and $\mathcal{E}(\tilde{G}) = \left\{e_{\ \mathrm{mod}\ (j-1,V),\ \mathrm{mod}\ (j+1,V)}, \ldots, e_{\ \mathrm{mod}\ (j-k,V),\ \mathrm{mod}\ (j+k,V)}\right\}$, where $k = (V-1)/2$, is a maximum mathching. By following the aforementioned method, $\forall i \in \mathcal{V}(G)$, it is possible to generate a maximum matching D_i. Then one can easily observe that for any i, j

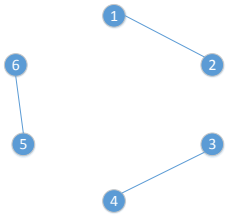
$$\mathcal{E}(D_i) \cap \mathcal{E}(D_j) = \emptyset, \tag{44}$$

and,

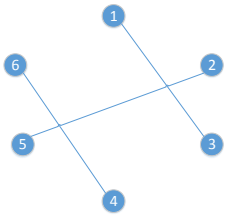
$$\cup_{i\in[V]}\, \mathcal{E}(D_i) = \mathcal{E}(G). \tag{45}$$

□

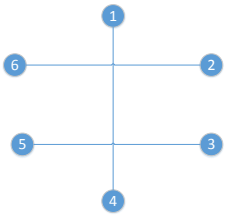
[3]By a disjoint partition, we mean that the distinct partitions, represented as sets, are disjoint.



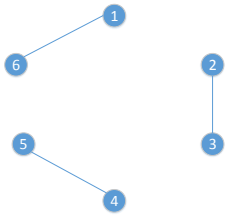
(a) $\{\{1,2\},\{3,4\},\{5,6\}\}$



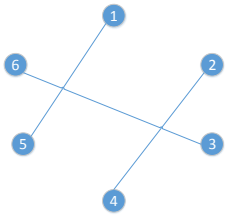
(b) $\{\{1,3\},\{2,5\},\{4,6\}\}$



(c) $\{\{1,4\},\{3,5\},\{2,6\}\}$



(d) $\{\{1,6\},\{2,3\},\{4,5\}\}$



(e) $\{\{1,5\},\{2,4\},\{3,6\}\}$

Fig. 3: All perfect matchings and corresponding partitions for $\{1,2,3,4,5,6\}$

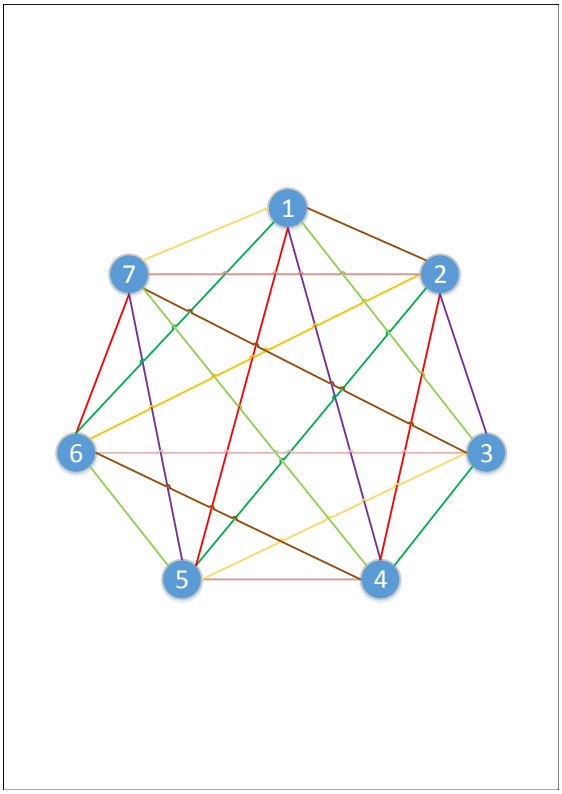


Fig. 4: Each color represents a maximum matching.

Algorithm 5: Set construction for the case of even k^h

Input : $\mathcal{K}^h$, $\{n_j\}_{j\in\mathcal{K}^l}$
Output: $\{\Delta_{i,j}\}_{i\in\mathcal{K}^h, j\in\mathcal{K}^l}$, $\mathcal{B}$

$\Delta_{i,j} \leftarrow \{\}$, $\forall i \in \mathcal{K}^h, j \in \mathcal{K}^l$;
$\mathcal{B} \leftarrow \{\}$;
for *all* $j \in \mathcal{K}^l$ **do**
  construct $\mathcal{Q}^{n_j}$
  for *all* $\{i,k\} \in \mathcal{Q}^{n_j}$ **do**
    $\Delta_{i,j} \leftarrow \Delta_{i,j} \bigcup \{W_{d_i,\{k,j\}}\}$;
    $\Delta_{k,j} \leftarrow \Delta_{k,j} \bigcup \{W_{d_k,\{i,j\}}\}$;
    $\mathcal{B} \leftarrow \mathcal{B} \bigcup \{W_{d_i,\{k,j\}} \oplus W_{d_k,\{i,j\}}\}$;
  end
end

Decomposition of a complete graph G, with $V = 7$, into edge-disjoint maximum matchings is illustrated in Figure 4. From Lemma 5 and Lemma 6, we can conclude that for a given set $\mathcal{K}$, for each $i \in \mathcal{K}$, it is possible to have a partition of $\mathcal{K} \setminus \{i\}$ with blocks of size two, denoted by $\mathcal{P}^{\mathcal{K}\setminus\{i\}}$ such that

$$\mathcal{P}^{\mathcal{K}\setminus\{i\}} \cap \mathcal{P}^{\mathcal{K}\setminus\{j\}} = \emptyset \text{ for any } i, j \in \mathcal{K}.$$

Appendix B
Construction procedure for $\Delta_{i,j}$ and $\mathcal{B}$

In this section, we will explain the procedure for constructing the set of multicast messages, $\mathcal{B}$, that are sent in the last step of the delivery phase. The main concern of this procedure is to satisfy the constraint $|\Delta_{i,j}| = n_{i,j}$[4], $\forall i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$, while constructing the set $\mathcal{B}$. In the construction procedure, we consider two cases, where k^h is even and odd, separately.

A. Even number of high-level users

Before presenting the algorithm for an even number of high-level users, we will briefly explain how the concept of *partitions* can be utilized for the construction of set $\mathcal{B}$. Consider $\mathcal{K}^h = \{1,2,3,4,5,6\}$ and a partition of $\mathcal{K}^h$ with blocks of size two[5], e.g., $\mathcal{P}^{\mathcal{K}^h} = \{\{1,2\},\{3,4\},\{5,6\}\}$. Then, for a particular $j \in \mathcal{K}^l$, each $\{i,k\} \in \mathcal{P}^{\mathcal{K}^h}$ can be converted into the following multicast message[6]: $W_{d_i,\{k,j\}} \oplus W_{d_k,\{i,j\}}$. To this end, the specified partition $\mathcal{P}^{\mathcal{K}^h}$ corresponds to the following set of multicast messages;

$$\left\{W_{d_1,\{2,j\}} \oplus W_{d_2,\{1,j\}}, W_{d_3,\{4,j\}} \oplus W_{d_4,\{3,j\}}, W_{d_5,\{6,j\}} \oplus W_{d_6,\{5,j\}}\right\}. \tag{46}$$

Recall that, $\mathcal{B}$ is the set of multicast-messages that are not decomposed and are sent in the last step of the delivery phase. Hence, if the multicast messages in (46) are added to $\mathcal{B}$, then sub-files $W_{d_1,\{2,j\}}$, $W_{d_2,\{1,j\}}$, $W_{d_3,\{4,j\}}$, $W_{d_4,\{3,j\}}$,$W_{d_5,\{6,j\}}$ and $W_{d_6,\{5,j\}}$ are added to sets $\Delta_{1,j}$, $\Delta_{2,j}$, $\Delta_{3,j}$, $\Delta_{4,j}$, $\Delta_{5,j}$, and $\Delta_{6,j}$, respectively. One can observe that, for a particular $j \in \mathcal{K}^l$, if a partition $\mathcal{P}^{\mathcal{K}^h}$ is used for adding multicast messages to set $\mathcal{B}$, then exactly one sub-file is added to the set $\Delta_{i,j}$ for each $i \in \mathcal{K}^h$. Hence, for a particular $j \in \mathcal{K}^l$, we consider any n_j disjoint partitions $\mathcal{P}_1^{\mathcal{K}^h}, \ldots, \mathcal{P}_{n_j}^{\mathcal{K}^h}$ to determine the multicast messages to be included in set $\mathcal{B}$, and the constraint $|\Delta_{i,j}| = n_j$ would be satisfied for all $i \in \mathcal{K}^h$. Hence, we define $\mathcal{Q}^{n_j} = \cup_{i=1:n_j} \mathcal{P}_i^{\mathcal{K}^h}$, and use it to add multicast messages to set $\mathcal{B}$ in Algorithm 5. We note that $\mathcal{Q}^{n_j}$ can be considered as a set of user pairings, where each user appears in exactly n_j pairings. Further details on the construction of the disjoint partitions $\mathcal{P}_1^{\mathcal{K}^h}, \ldots, \mathcal{P}_n^{\mathcal{K}^h}$ are explained in Appendix A.

B. Odd number of high-level users

Above, we first obtained a set of node pairings $\mathcal{Q}^{n_j}$ for each $j \in \mathcal{K}^l$, and then constructed the set of multicast messages $\mathcal{B}$ using these node pairings in Algorithm 5. We recall that $\mathcal{Q}^{n_j}$ is the union of n_j partitions generated from $\mathcal{K}^h$. However, when k^h is odd, it is not possible to obtain partition of $\mathcal{K}^h$ with blocks of size two. We remark that, if k^h is odd, k^l must be even, which means that for $k^l/2$ low-level users, n_j will be an odd number n_{odd}, and for the remaining low-level users, n_j will be an even number n_{even}. Let $\mathcal{K}^l_{odd}$ and $\mathcal{K}^l_{even}$ be the subset of low-level users with n_{odd} and n_{even}, respectively. Furthermore,

[4]Since the same partitioning is applied to all Λ_i's, $\forall i \in \mathcal{K}^h$, $n_{i,j} = n_j$ for all $i \in \mathcal{K}^h$.

[5]In the scope of this work, we use the term partition to refer only a particular type of partition with blocks of size two. Hence, from this point on we use the term partition without further specifying the block sizes.

[6]Throughout the paper, we often refer to the subset $\{i,k\}$ as a node pairing.

Algorithm 6: Construction of $\mathcal{Q}^{n_{even}}$ for odd k^h

Input : $\mathcal{K}^h, n_{even}$
Output: $\mathcal{Q}^{n_{even}}$
$\tilde{\mathcal{K}}^h \subset \mathcal{K}^h$ with $|\tilde{\mathcal{K}}^h| = k^h - 1$;
for $j = 1 : n_{even}/2$ **do**
  $\{i, k\} \leftarrow \mathcal{P}^{\tilde{\mathcal{K}}^h}(j)$;
  $\mathcal{Q}^{n_{even}} \leftarrow \mathcal{Q}^{n_{even}} \bigcup (\mathcal{P}^{\mathcal{K}^h \setminus \{i\}} \bigcup \mathcal{P}^{\mathcal{K}^h \setminus \{k\}} \bigcup \{i, k\})$;
end

Partition	Corresponding set
$\mathcal{P}^{\mathcal{K}^h \setminus \{1\}}$	$\{\{2,7\},\{3,6\},\{4,5\}\}$
$\mathcal{P}^{\mathcal{K}^h \setminus \{6\}}$	$\{\{2,3\},\{1,4\},\{5,7\}\}$
$\mathcal{P}^{\mathcal{K}^h \setminus \{2\}}$	$\{\{1,3\},\{4,7\},\{5,6\}\}$
$\mathcal{P}^{\mathcal{K}^h \setminus \{5\}}$	$\{\{4,6\},\{3,7\},\{1,2\}\}$

TABLE VI: Partitions used to construct $\mathcal{Q}^4$ for $\mathcal{K}^h = \{1,2,3,4,5,6,7\}$.

let k^l_{odd} and k^l_{even} denote the cardinality of the sets $\mathcal{K}^l_{odd}$ and $\mathcal{K}^l_{even}$, respectively.

For the case of n_{even}, we introduce a new method, Algorithm 6, to construct $\mathcal{Q}^{n_{even}}$ for each $j \in \mathcal{K}^l$. In Algorithm 6, we use the notation $\mathcal{A}(i)$ to denote the ith element of set $\mathcal{A}$ for an arbitrary ordering of its elements[7]. We also want to remark that the partitions used in Algorithm 6 are disjoint, i.e.,

$$\mathcal{P}^{\mathcal{K}^h \setminus \{i\}} \cap \mathcal{P}^{\mathcal{K}^h \setminus \{k\}} = \emptyset, \tag{47}$$

where $i, k \in \mathcal{K}^h$, and $i \neq k$. The process of obtaining these disjoint partitions are explained in Appendix A. Now, one can easily observe that, each high-level user appears in exactly one pairing in $\mathcal{P}^{\mathcal{K}^h \setminus \{i\}}$, except i, and in $\mathcal{P}^{\mathcal{K}^h \setminus \{i\}} \bigcup \mathcal{P}^{\mathcal{K}^h \setminus \{k\}} \cup \{i, k\}$ each high-level user appears exactly in two pairings. To clarify, assume that we want to construct $\mathcal{Q}^4$ for $\mathcal{K}^h = \{1,2,3,4,5,6,7\}$. We can set $\tilde{\mathcal{K}}^h = \{1,2,3,4,5,6\}$ and use partition $\mathcal{P}^{\tilde{\mathcal{K}}^h} = \{\{1,6\},\{2,5\},\{3,4\}\}$, so that $\mathcal{P}^{\tilde{\mathcal{K}}^h}(1) = \{1,6\}$ and $\mathcal{P}^{\tilde{\mathcal{K}}^h}(2) = \{2,5\}$. Then, Algorithm 6 uses the partitions given in Table VI to construct $\mathcal{Q}^4$ as follows:

$$\begin{aligned}\mathcal{Q}^4 = \{&\{2,7\},\{3,6\},\{4,5\},\{2,3\},\{1,4\},\{5,7\},\{1,6\}\\ &\{1,3\},\{4,7\},\{5,6\},\{4,6\},\{3,7\},\{1,2\},\{2,5\}\}.\end{aligned} \tag{48}$$

Eventually, in $\mathcal{Q}^{n_{even}}$ each high-level user appears exactly in n_{even} pairings. Hence, as in Algorithm 5, $\mathcal{Q}^{n_{even}}$ can be used to construct sets $\Delta_{i,j}$ as well the set of multicast messages $\mathcal{B}$.

We remark that, for each $j \in \mathcal{K}^l_{even}$ the same set of node pairings $\mathcal{Q}^{n_{even}}$ is used, thus the process is identical for each $j \in \mathcal{K}^l_{even}$. However, for low-level users $j \in \mathcal{K}^l_{odd}$ the process will not be identical since it is not possible to construct a single $\mathcal{Q}^{n_{odd}}$ for all $j \in \mathcal{K}^l_{odd}$. Nevertheless, we follow a similar procedure to construct the multicast messages for the low-level users in $\mathcal{K}^l_{odd}$.

The overall procedure for the case of odd number of high-level users is illustrated in Algorithm 7. In the algorithm, the set of node pairings $\mathcal{Q}_j$ for each $j \in \mathcal{K}^l_{odd}$ is constructed separately, and is used to decide the multicast message to be placed in $\mathcal{B}$, and the sub-files to be placed in sets $\Delta_{i,j}$, as in Algorithm 1. From the construction, one can easily observe that in $\mathcal{Q}_j$, each high-level user appears exactly in n_{odd} pairings except a particular k that appears in $n_{odd} - 1$ pairings. Thus, when $\mathcal{Q}_j$ is used to construct multicast messages, constraint $|\Delta_{i,j}| = n_{odd}$ is satisfied for all $i \in \mathcal{K}^h$, except $i = k$. Therefore, at line 11 and 16 in Algorithm 7, we add a sub-file to set $\Delta_{k,j}$, where the high-level user k appears in $n_{odd} - 1$ pairings in $\mathcal{Q}_j$, and at line 17 we XOR these sub-files. Hence, eventually we ensure that, at the end of the algorithm the equality constraint $|\Delta_{i,j}| = n_{odd}$ is satisfied for all $i \in \mathcal{K}^h$ and for all $j \in \mathcal{K}^l_{odd}$.

With the proposed set construction algorithms we are ensuring that, in the third and fourth steps of the delivery phase all the sub-files are delivered with a multicasting gain of two[8], which explains the achievable delivery rate.

Now, we go back to Example 1, where we have an odd number of high-level files. In particular, we have $\mathcal{K}^h = \{1,2,3,4,5\}$, $\mathcal{K}^l_{odd} = \{6\}$ and $\mathcal{K}^l_{even} = \{7\}$, thus $n_{odd} = 1$ and $n_{even} = 2$. Further, partition are given as $\mathcal{P}^{\mathcal{K}^h \setminus \{1\}} = \{\{2,5\},\{3,4\}\}$, $\mathcal{P}^{\mathcal{K}^h \setminus \{2\}} = \{\{1,3\},\{4,5\}\}$, $\mathcal{P}^{\mathcal{K}^h \setminus \{3\}} = \{\{1,5\},\{2,4\}\}$, $\mathcal{P}^{\mathcal{K}^h \setminus \{4\}} = \{\{3,5\},\{1,2\}\}$, $\mathcal{P}^{\mathcal{K}^h \setminus \{5\}} = \{\{1,4\},\{3,2\}\}$. Then, Algorithm 7 takes the set $\tilde{\mathcal{K}}^h = \{1,2,3,4\}$ and use the partition $\mathcal{P}^{\mathcal{K}^h} = \mathcal{P}^{\mathcal{K}^h \setminus \{5\}} = \{\{1,4\},\{2,3\}\}$. Accordingly, Algorithm 7 constructs $\mathcal{Q}^{n_{odd}-1} = \emptyset$ and $\mathcal{Q}^{n_{even}} = \{\{1,2\},\{1,4\},\{2,5\},\{3,4\},\{3,5\}\}$. Thereafter, sets $\mathcal{Q}_6$ and $\mathcal{Q}_7$ are constructed as $\mathcal{Q}_6 = \{\{2,5\},\{3,4\}\}$ and

[7]Although we use index for the set $\mathcal{P}^{\tilde{\mathcal{K}}^h}$, Algorithm 6 does not require a particular ordering for this set.

[8]When $|\cup_{i \in \mathcal{K}^h, j \in \mathcal{K}^l} \Delta_{i,j}|$ is odd, exactly one sub-file is unicasted, while the remaining sub-files achieve a multicasting gain of two, as in Example 1.

Algorithm 7: Set construction for the case of odd k^h

Input : $\mathcal{K}^l_{odd}, \mathcal{K}^l_{even}, \mathcal{K}^h, \{n_{even}, n_{odd}\}$
Output: $\mathcal{B}, \left\{\Delta_{i,j}\right\}_{i\in K^h, j\in K^l}$

$\Delta_{i,j} \leftarrow \{\}\ \forall\ i \in K^h, j \in K^l$, $Q_j \leftarrow \{\}\ \forall j \in K^l$;
$\tilde{\mathcal{K}}^h \subset \mathcal{K}^h$ with $|\tilde{\mathcal{K}}^h| = k^h - 1$;
construct $Q^{n_{odd}-1}$ and construct $Q^{n_{even}}$;
$Q_j \leftarrow Q^{n_{even}}, \forall j \in \mathcal{K}^l_{even}$;
$Q_j \leftarrow Q^{n_{odd}-1}, \forall j \in \mathcal{K}^l_{odd}$;
$\tilde{n} \leftarrow \lceil n_{odd}/2 \rceil$;
for $m = 1 : k^l_{odd}$ **do**
  if *m is odd* **then**
    $\{i,k\} \leftarrow \mathcal{P}^{\tilde{\mathcal{K}}^h}(\tilde{n})$;
    $Q_{\mathcal{K}^l_{odd}(m)} \leftarrow Q_{\mathcal{K}^l_{odd}(m)} \bigcup \mathcal{P}^{\mathcal{K}^h\setminus\{i\}}$;
    $\Delta_{i,\mathcal{K}^l_{odd}(m)} \leftarrow \Delta_{i,\mathcal{K}^l_{odd}(m)} \bigcup \left\{W_{d_i,\{k,\mathcal{K}^l_{odd}(m)\}}\right\}$;
  end
  if *m is even* **then**
    $\{i,k\} \leftarrow \mathcal{P}^{\tilde{\mathcal{K}}^h}(\tilde{n})$;
    $Q_{\mathcal{K}^l_{odd}(m)} \leftarrow Q_{\mathcal{K}^l_{odd}(m)} \bigcup \mathcal{P}^{\mathcal{K}^h\setminus\{k\}}$;
    $\Delta_{k,\mathcal{K}^l_{odd}(m)} \leftarrow \Delta_{k,\mathcal{K}^l_{odd}(m)} \bigcup \left\{W_{d_k,\{i,\mathcal{K}^l_{odd}(m)\}}\right\}$;
    $\mathcal{B} \leftarrow \mathcal{B} \bigcup \left\{W_{d_i,\{k,\mathcal{K}^l_{odd}(m-1)\}} \oplus W_{d_k,\{i,\mathcal{K}^l_{odd}(m)\}}\right\}$;
  end
end
for *all* $j \in K^l$ **do**
  for *all* $(i,k) \in Q_j$ **do**
    $\Delta_{i,j} \leftarrow \Delta_{i,j} \bigcup \left\{W_{d_i,\{k,j\}}\right\}$;
    $\Delta_{k,j} \leftarrow \Delta_{k,j} \bigcup \left\{W_{d_k,\{i,j\}}\right\}$;
    $\mathcal{B} \leftarrow \mathcal{B} \bigcup \left\{W_{d_i,\{k,j\}} \oplus W_{d_k,\{i,j\}}\right\}$;
  end
end

$Q_7 = \{\{1,2\},\{1,4\},\{2,5\},\{3,4\},\{3,5\}\}$. The corresponding set of multicast messages, $\mathcal{B}$, are already illustrated with blue in Table II.

Appendix C
Partition of Λ

In this part, we will show how $\Lambda_i = \left\{W_{d_i,\{k,j\}} : k, j \in \mathcal{K}^l\right\}$ can be approximately partitioned. Recall that the main concern is to assign sub-file $W_{d_i,\{k,j\}}$ to either $\Lambda_{i,k}$ or $\Lambda_{i,j}$ in order to achieve approximately uniform cardinality sets $\left\{\Lambda_{i,j}\right\}_{j\in\mathcal{K}^l}$. We consider two cases, where k^l is even and odd, respectively. We start with the case where k^l is an odd number. Let the set of partitions $\left\{\mathcal{P}^{\mathcal{K}^l\setminus\{i\}}\right\}_{i\in\mathcal{K}^l}$ are given for $\mathcal{K}^l$. Then Algorithm 8 is used to construct $\left\{\Lambda_{i,j}\right\}_{j\in\mathcal{K}^l}$. Note that, in each step of Algorithm 8 the size of set $\mathcal{W}$ is equal to k^l and low-level user index j appears in exactly two sub-files in the $\mathcal{W}$. By "sequential distribution" we mean that we start with some low-level user j and take the two sub-files $W_{d_i,\{k,j\}}, W_{d_i,\{\acute{k},j\}} \in \mathcal{W}$. We start with assigning $W_{d_i,\{k,j\}}$ to set $\Lambda_{i,j}$, then the other sub-file $W_{d_i,\{\acute{k},j\}}$ is assigned to set $\Lambda_{i,\acute{k}}$. Thereafter, we take the file $W_{d_i,\{\acute{k},\acute{j}\}}$ and assign it to set $\Lambda_{i,\acute{j}}$ and we continue the process in the same way. Formally speaking, each Q in the algorithm resembles a Hamiltonian cycle, and in Algorithm 8 we are assigning each edge in the cycle to a node. To clarify, consider Hamiltonian cycle corresponding to $Q = \{\{1,7\},\{7,5\},\{2,6\},\{2,3\},\{1,4\},\{6,4\},\{3,5\}\}$ is illustrated in Fig. 5. Assume that, we start from node 1 and assign $e_{1,7}$ to node 1, and edges $e_{7,5}, e_{3,5}, e_{2,3}, e_{2,6}, e_{6,4}$,and $e_{1,4}$ are assigned to nodes $7, 5, 3, 2, 6, 4$, respectively. Note that, each edge corresponds to a sub-file and each node corresponds to a set $\Lambda_{i,j}$. We further remark that a complete graph G with odd order can be decomposed into Hamiltonian cycles (as done in Algorithm 8); hence, Λ_i can be partitioned uniformly where $|\Lambda_{i,j}| = (k_l - 1)/2,\ \forall i \in \mathcal{K}^h,\ \forall j \in \mathcal{K}^h$.

For the case of even k^l we can use a similar approach, but this time we construct the Hamiltonian cycles via combining two perfect matchings. Different from the previous case complete graph G with even order cannot be decomposed to Hamiltonian cycles. Hence, in the end we will have a remaining perfect matching. Those edges in the last perfect matching are assigned

Algorithm 8: Partition of Λ_i

Input : $\left\{\mathcal{P}^{\mathcal{K}^l\setminus\{j\}}\right\}_{j\in\mathcal{K}^l}$,$\Lambda_i$

Output: $\left\{\Lambda_{i,j}\right\}_{j\in K^l}$

Pick an element $s \in \mathcal{K}^l$ randomly
for $m = 1 : (k^l - 1)/2$ **do**
  $\{j, k\} \leftarrow \mathcal{P}^{\mathcal{K}^l\setminus\{s\}}(m)$;
  $\mathcal{Q} \leftarrow \mathcal{P}^{\mathcal{K}^l\setminus\{j\}} \bigcup \mathcal{P}^{\mathcal{K}^l\setminus\{k\}} \bigcup \{j, k\}$;
  $\mathcal{W} \leftarrow \left\{W_{d_i,\{k,j\}} : \{k, j\} \in \mathcal{Q}\right\}$;
  Distribute W to sets $\left\{\Lambda_{i,j}\right\}_{j\in K^l}$ sequentially
end

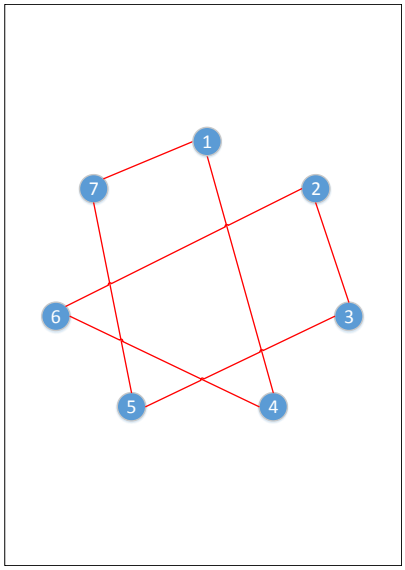


Fig. 5: Hamiltonian cycle corresponding to $\mathcal{Q} = \{\{1, 7\}, \{7, 5\}, \{2, 6\}, \{2, 3\}, \{1, 4\}, \{6, 4\}, \{3, 5\}\}$.

to nodes randomly. Therefore, in the end, $k^l/2$ of the partitions will have cardinality $k^l/2 - 1$, while the remaining will have cardinality $k^l/2$.

REFERENCES

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, May 2014.
[2] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, Aug 2015.
[3] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, Feb 2018.
[4] M. M. Amiri, Q. Yang, and D. Gündüz, "Decentralized caching and coded delivery with distinct cache capacities," *IEEE Transactions on Communications*, vol. 65, no. 11, pp. 4657–4669, Nov 2017.
[5] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Coded caching for heterogeneous systems: An optimization perspective," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5321–5335, 2019.
[6] ——, "Device-to-device coded-caching with distinct cache sizes," *IEEE Transactions on Communications*, pp. 1–1, 2020.
[7] A. Tang, S. Roy, and X. Wang, "Coded caching for wireless backhaul networks with unequal link rates," *IEEE Transactions on Communications*, vol. PP, no. 99, pp. 1–1, 2017.
[8] M. M. Amiri and D. Gündüz, "Cache-aided content delivery over erasure broadcast channels," *IEEE Transactions on Communications*, vol. PP, no. 99, pp. 1–1, 2017.
[9] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, Feb 2017.
[10] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, June 2017.
[11] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 349–366, Jan 2018.
[12] J.Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, May 2017.
[13] A. M. Daniel and W. Yu, "Optimization of heterogeneous coded caching," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1893–1919, 2020.
[14] S. Jin, Y. Cui, H. Liu, and G. Caire, "Structural properties of uncoded placement optimization for coded delivery," *CoRR*, vol. abs/1707.07146, 2017.
[15] A. Ramakrishnan, C. Westphal, and A. Markopoulou, "An efficient delivery scheme for coded caching," in *Proceedings of the 2015 27th International Teletraffic Congress*, ser. ITC '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 46–54.

[16] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *2008 16th Int. Workshop on Quality of Service*, June 2008.
[17] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, Oct 2009.
[18] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network - measurements, models, and implications," *Comput. Netw.*, vol. 53, no. 4, pp. 501–514, Mar. 2009.
[19] J. Lin, Z. Li, G. Xie, Y. Sun, K. Salamatian, and W. Wang, "Mobile video popularity distributions and the potential of peer-assisted video delivery," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 120–126, November 2013.
[20] K. Wan, D. Tuninetti, and P. Piantanida, "Novel delivery schemes for decentralized coded caching in the finite file size regime," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 1183–1188.
[21] N. Zhang and M. Tao, "Fitness-aware coded multicasting for decentralized caching with finite file packetization," *IEEE Wireless Communications Letters*, pp. 1–1, 2018.
[22] M. S. Heydar Abad, E. Ozfatura, O. Ercetin, and D. Gündüz, "Dynamic content updates in heterogeneous wireless networks," in *2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 2019, pp. 107–110.