Fundamental Limits of Coded Caching with Multiple Antennas, Shared Caches and Uncoded Prefetching

Emanuele Parrinello, Ayşe Ünsal and Petros Elia

***Abstract*—The work explores the fundamental limits of coded caching in the setting where a transmitter with potentially multiple (N_0) antennas serves different users that are assisted by a smaller number of caches. Under the assumption of uncoded cache placement, the work derives the exact optimal worst-case delay and DoF, for a broad range of user-to-cache association profiles where each such profile describes how many users are helped by each cache. This is achieved by presenting an information-theoretic converse based on index coding that succinctly captures the impact of the user-to-cache association, as well as by presenting a coded caching scheme that optimally adapts to the association profile by exploiting the benefits of encoding across users that share the same cache.**

The work reveals a powerful interplay between shared caches and multiple senders/antennas, where we can now draw the striking conclusion that, as long as each cache serves at least N_0 users, adding a single degree of cache-redundancy can yield a DoF increase equal to N_0, while at the same time — irrespective of the profile — going from 1 to N_0 antennas reduces the delivery time by a factor of N_0. Finally some conclusions are also drawn for the related problem of coded caching with multiple file requests.

***Index Terms*—Caching networks, coded caching, shared caches, delivery rate, uncoded cache placement, index coding, MISO broadcast channel, multiple file requests, network coding.**

I. INTRODUCTION

In the context of communication networks, the emergence of predictable content, has brought to the fore the use of caching as a fundamental ingredient for handling the exponential growth in data volumes. A recent information theoretic exposition of the cache-aided communication problem [1], has revealed the potential of caching in allowing for the elusive scaling of networks, where a limited amount of (bandwidth and time) resources can conceivably suffice to serve an ever increasing number of users.

A. Coded Caching

This exposition in [1] considered a shared-link broadcast channel (BC) scenario where a single-antenna transmitter has access to a database of N equally-sized files and is connected (via a bottleneck link) to K users, each having an isolated cache of size equal to the size of M files.

The authors are with the Communication Systems Department at EURECOM, Sophia Antipolis, 06410, France (email: parrinel@eurecom.fr, unsal@eurecom.fr, elia@eurecom.fr). The work is supported by the European Research Council under the EU Horizon 2020 research and innovation program / ERC grant agreement no. 725929 (project DUALITY).

This work appeared in part in the proceedings of ITW 2018. A preliminary version of this work, focusing only on the single-stream setting, can be found in [4].

Assuming that the shared-link connecting the transmitter to the users has a normalized capacity that equals 1 file per unit of time, the work in [1] showed that any set of K simultaneous demanded files (one file requested per user) can be delivered with a delay of at most $T = \frac{K(1-\gamma)}{1+K\gamma}$ where $\gamma \triangleq \frac{M}{N}$ denotes the per-user cache size normalized by the size of the library. This implied an ability to treat $K\gamma + 1$ users at a time; a number that is often referred to as the cache-aided sum *degrees of freedom* (DoF) $d_\Sigma \triangleq \frac{K(1-\gamma)}{T}$, corresponding to a caching gain of $K\gamma$ additional served users due to caching.

For this same shared-link setting, this performance was shown to be approximately optimal (cf. [1]), and under the basic assumption of uncoded cache placement where caches store uncoded content from the library, it was shown to be exactly optimal (cf. [2] as well as [3]).

The novel *coded* approach to caching has been studied in a variety of settings that include non-uniform files popularity distributions [5]–[7], uneven topologies [8], [9], a broad range of channels such as MIMO broadcast channels [10] and erasure channels [11], devide-to-device (D2D) networks [12], coded caching under secrecy constraints [13], and other settings as well [14]–[19]. Recently some progress has also been made in ameliorating the well known subpacketization bottleneck; for this see for example [22]–[28].

B. Cache-aided Heterogeneous Networks: Coded Caching with Shared Caches

Another step in further exploiting the use of caches, was to explore coded caching in the context of the so-called heterogeneous networks which better capture aspects of more realistic settings such as larger wireless networks. Here the term *heterogeneous* refers to scenarios where one or potentially many multi-antenna transmitters (base-stations) communicate to a set of users, with the assistance of smaller nodes. In our setting, these smaller helper nodes will serve as caches that will be shared among the users. This cache-aided heterogeneous topology nicely captures an evolution into denser networks where many wireless access points work in conjunction with bigger base stations, in order to better handle interference and to alleviate the backhaul load by replacing it with storage capacity at the communicating nodes.

The use of caching in such networks was famously explored in the *Femtocaching* work in [29], where wireless receivers are assisted by helper nodes of a limited cache size, whose main role is to bring content closer to the users. A transition to coded caching can be found in [30] which considered a similar shared-cache heterogeneous network as here, where each receiving user can have access to a main single-antenna base station (single-sender) and to different helper caches. In this context, under a uniform user-to-cache association where each cache serves an equal number of users, [30] proposes a coded caching scheme which was shown to perform to within a certain constant factor from the optimal. This uniform setting is addressed also in [31], again for the single antenna case. Interesting work can also be found in [32] which explores the single-stream (shared-link) coded caching scenario with shared caches, where the uniformity condition is lifted, and where emphasis is placed on designing schemes with centralized coded prefetching with small sum-size caches where the total cache size is smaller than the library size (i.e., where $KM < N$).

The study of the shared-caches setting is important because, in addition to applying to realistic scenarios where receivers (physically) share cache-aided helper nodes, the setting also reflects the effect of subpacketization constraints [22] that typically force cache-aided receivers to have identically-filled caches.

Coded caching with multiple file requests: Related work can also be found on the problem of coded caching with multiple file requests per receiver, which — for the single-stream, error free case — is closely related to the shared cache problem here. Such work — all in the shared-link case ($N_0 = 1$) — appears in [33], [34] in the context of single-layer coded caching. Somewhat related work also appears in [35], [36] in the context of hierarchical coded caching. Recent progress can also be found in [37] which establishes the exact optimal worst-case delay — under uncoded cache placement, for the shared-link case — for the uniform case where each user requests an equal number of files[1]. As a byproduct of our results here, in the context of worst-case demands, we establish the exact optimal performance of the multiple file requests problem for any (not necessarily uniform) user-to-file association profile.

Our contribution: In the heterogeneous setting with shared caches, we here explore the effect of user-to-cache association profiles and their non-uniformity, and we characterize how this effect is scaled in the presence of multiple antennas or multiple senders. Such considerations are motivated by realistic constraints in assigning users to caches, where these constraints may be due to topology, cache capacity or other factors. As it turns out, there is an interesting interplay between all these aspects, which is crisply revealed here as a result of a new scheme and an outer bound that jointly provide exact optimality results. Throughout the paper, emphasis will be placed on the shared-cache scenario, but some of the results will be translated directly to the multiple file request problem which will be described later on.

C. Notation

For n being a positive integer, $[n]$ refers to the following set $[n] \triangleq \{1, 2, \ldots, n\}$, and $2^{[n]}$ denotes the power set of $[n]$. The expression $\alpha|\beta$ denotes that integer α divides integer β. Permutation and binomial coefficients are denoted and defined by $P(n,k) \triangleq \frac{n!}{(n-k)!}$ and $\binom{n}{k} \triangleq \frac{n!}{(n-k)!k!}$, respectively. For a set $\mathcal{A}$, $|\mathcal{A}|$ denotes its cardinality. $\mathbb{N}$ represents the natural numbers. We denote the lower convex envelope of the points $\{(i, f(i))|i \in [n] \cup \{0\}\}$ for some $n \in \mathbb{N}$ by $Conv(f(i))$. The concatenation of a vector $\boldsymbol{v}$ with itself N times is denoted by $(\boldsymbol{v}\|\boldsymbol{v})_N$, while the concatenation of a vector $\boldsymbol{v}$ with another vector $\mathbf{w}$ is denoted by $\boldsymbol{v}\|\mathbf{w}$. For $n \in \mathbb{N}$, we denote the symmetric group of all permutations of $[n]$ by S_n. To simplify notation, we will also use such permutations $\pi \in S_n$ on vectors $\boldsymbol{v} \in \mathbb{R}^n$, where $\pi(\boldsymbol{v})$ will now represent the action of the permutation matrix defined by π, meaning that the first element of $\pi(\boldsymbol{v})$ is $\boldsymbol{v}_{\pi(1)}$ (the $\pi(1)$ entry of $\boldsymbol{v}$), the second is $\boldsymbol{v}_{\pi(2)}$, and so on. Similarly $\pi^{-1}(\cdot)$ will represent the inverse such function and $\pi_s(\boldsymbol{v})$ will denote the sorted version of a real vector $\boldsymbol{v}$ in descending order.

D. Paper Outline

In Section II we give a detailed description of the system model and the problem definition, followed by the main results in Section III, first for the shared-link setting with shared caches[2], and then for the multi-antenna/multi-sender setting. In Section IV, we introduce the scheme for a broad range of parameters. The scheme is further explained with an example in this section. We present the information theoretic converse along with an explanatory example for constructing the lower bound in Section V. Lastly, in Section VI we draw some basic conclusions, while in the Appendix Section VII we present some proof details.

II. System Model

We consider a basic broadcast configuration with a transmitting server having N_0 transmitting antennas and access to a library of N files $W^1, W^2, \ldots, W^N$, each of size equal to one unit of 'file', where this transmitter is connected via a broadcast link to K receiving users and to $\Lambda \leq K$ helper nodes that will serve as caches which store content from the library[3] . The communication process is split into a) the cache-placement phase, b) the user-to-cache assignment phase during which each user is assigned to a single cache, and c) the delivery phase where each user requests a single file independently and during which the transmitter aims to deliver these requested files, taking into consideration the cached content and the user-to-cache association.

[1]This work explores other cases as well, such as that where the performance measure is the average delivery delay, for which various bounds are presented.

[2]We also introduce a very brief parenthetical note that translates these results to the multiple file request scenario.

[3]We notice that while the representation here is of a wireless setting, the results apply directly to the standard wired multi-sender setting. In the high-SNR regime of interest, when $N_0 = 1$ and $\Lambda = K$ (where each cache is associated to one user), the setting matches identically the original single-stream setting in [1]. In particular, the file size and log(SNR) are here scaled so that, as in [1], each point-to-point link has (ergodic) capacity of 1 file per unit of time. When $N_0 > 1$ and $\Lambda = K$ (again each cache is associated to one user), the setting matches the multi-server *wireline* setting of [38] with a fully connected linear network, which we now explore in the presence of fewer caches serving potentially different numbers of users.

a) Cache placement phase: During this phase, helper nodes store content from the library without having knowledge of the users' requests. Each helper cache has size $M \leq N$ units of file, and no coding is applied to the content stored at the helper caches; this corresponds to the common case of *uncoded cache placement*. We will denote by $\mathcal{Z}_\lambda$ the content stored by helper node λ during this phase. The cache-placement algorithm is oblivious of the subsequent user-to-cache association $\mathcal{U}$.

b) User-to-cache association: After the caches are filled, each user is assigned to exactly *one* helper node/cache, from which it can download content at zero cost. Specifically, each cache $\lambda = 1, 2, \ldots, \Lambda$, is assigned to a set of users $\mathcal{U}_\lambda$, and all these disjoint sets

$$\mathcal{U} \triangleq \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_\Lambda\}$$

form a partition of the set of users $\{1, 2, \ldots, K\}$, describing the overall association of the users to the caches.

This cache assignment is independent of the cache content and independent of the file requests to follow. We here consider any arbitrary user-to-cache association $\mathcal{U}$, thus allowing the results to reflect both an ability to choose/design the association, as well as to reflect possible association restrictions due to randomness or topology. Similarly, having the user-to-cache association being independent of the requested files, is meant to reflect the fact that such associations may not be able to vary as quickly as a user changes the requested content.

c) Content delivery: The delivery phase starts when each user $k = 1, \ldots, K$ requests from the transmitter, any *one* file W^{d_k}, $d_k \in \{1, \ldots, N\}$ out of the N files of the library. Upon notification of the entire *demand vector* $\boldsymbol{d} = (d_1, d_2, \ldots, d_K) \in \{1, \ldots, N\}^K$, the transmitter aims to deliver the requested files, each to their intended receiver, and the objective is to design a *caching and delivery scheme* χ that does so with limited (delivery phase) duration T, where the delivery algorithm has full knowledge of the user-to-cache association $\mathcal{U}$.

For each transmission, the received signals at user k, take the form

$$y_k = \boldsymbol{h}_k^T \boldsymbol{x} + w_k, \quad k = 1, \ldots, K \tag{1}$$

where $\boldsymbol{x} \in \mathbb{C}^{N_0 \times 1}$ denotes the transmitted vector satisfying a power constraint $\mathbb{E}(||\boldsymbol{x}||^2) \leq P$, $\boldsymbol{h}_k \in \mathbb{C}^{N_0 \times 1}$ denotes the channel gain of receiver k, and w_k represents the AWGN noise with unit power at user k. We will assume that the maximum allowed average power P is high (i.e., we will assume high signal-to-noise ratio (SNR)), that there exists perfect channel state information throughout the active users, that fading is statistically symmetric, and that each link (one antenna to one receiver) has ergodic capacity $\log(SNR) + o(log(SNR))$.

d) User-to-cache association profiles, and performance measure: As one can imagine, some user-to-cache association instances $\mathcal{U}$ may allow for higher performance than others; for instance, one can suspect that more uniform profiles may be preferable. Part of the objective of this work is to explore the effect of such associations on the overall performance. Toward this, for any given $\mathcal{U}$, we define the association *profile* (sorted histogram)

$$\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, \ldots, \mathcal{L}_\Lambda)$$

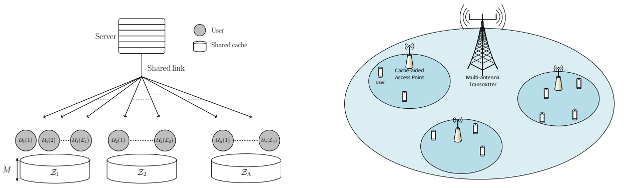

Fig. 1: Shared-link or multi-antenna broadcast channel with shared caches.

where $\mathcal{L}_\lambda$ is the number of users assigned to the λ-th *most populated* helper node/cache[4]. Naturally, $\sum_{\lambda=1}^{\Lambda} \mathcal{L}_\lambda = K$. Each profile $\boldsymbol{\mathcal{L}}$ defines a *class* $\mathcal{U}_{\boldsymbol{\mathcal{L}}}$ comprising all the user-to-cache associations $\mathcal{U}$ that share the same[5] profile $\boldsymbol{\mathcal{L}}$.

As in [1], the measure of interest T is the number of time slots, per file served per user, needed to complete delivery of any file-request vector[6] $\boldsymbol{d}$. We use $T(\mathcal{U}, \boldsymbol{d}, \chi)$ to define the delay required by some generic caching-and-delivery scheme χ to satisfy demand $\boldsymbol{d}$ in the presence of a user-to-cache association described by $\mathcal{U}$. To capture the effect of the user-to-cache association, we will characterize the optimal worst-case delivery time

$$T^*(\boldsymbol{\mathcal{L}}) \triangleq \min_{\chi} \max_{(\mathcal{U}, \boldsymbol{d}) \in (\mathcal{U}_{\boldsymbol{\mathcal{L}}}, \{1, \ldots, N\}^K)} T(\mathcal{U}, \boldsymbol{d}, \chi) \tag{2}$$

for each class. Our interest is in the regime of $N \geq K$ where there are more files than users.

III. Main Results

We first describe the main results for the single antenna case[7] (shared-link BC), and then generalize to the multi-antenna/multi-sender case.

A. Shared-Link Coded Caching with Shared Caches

The following theorem presents the main result for the shared-link case ($N_0 = 1$).

Theorem 1. *In the K-user shared-link broadcast channel with Λ shared caches of normalized size γ, the optimal delivery time within any class/profile $\boldsymbol{\mathcal{L}}$ is*

$$T^*(\boldsymbol{\mathcal{L}}) = Conv\left(\frac{\sum_{r=1}^{\Lambda - \Lambda\gamma} \mathcal{L}_r \binom{\Lambda - r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}}\right) \tag{3}$$

at points $\gamma \in \{\frac{1}{\Lambda}, \frac{2}{\Lambda}, \ldots, 1\}$.

Proof. The achievability part of the proof is given in Section IV, and the converse is proved in Section V after setting $N_0 = 1$.

[4]Here $\boldsymbol{\mathcal{L}}$ is simply the vector of the cardinalities of $\mathcal{U}_\lambda$, $\forall \lambda \in \{1, \ldots, \Lambda\}$, sorted in descending order. For example, $\mathcal{L}_1 = 6$ states that the highest number of users served by a single cache, is 6.

[5]An example of a user-to-cache assignment could have that users $\mathcal{U}_1 = (14, 15)$ are assigned to helper node 1, users $\mathcal{U}_2 = (1, 2, 3, 4, 5, 6, 7, 8)$ are assigned to helper node 2, and users $\mathcal{U}_3 = (9, 10, 11, 12, 13)$ to helper node 3. This corresponds to a profile $\boldsymbol{\mathcal{L}} = (8, 5, 2)$. The assignment $\mathcal{U}_1 = (1, 3, 5, 7, 9, 11, 13, 15)$, $\mathcal{U}_2 = (2, 4)$, $\mathcal{U}_3 = (6, 8, 10, 12, 14)$ would have the same profile, and the two resulting $\mathcal{U}$ would belong to the same class labeled by $\boldsymbol{\mathcal{L}} = (8, 5, 2)$.

[6]The time scale is normalized such that one time slot corresponds to the optimal amount of time needed to send a single file from the transmitter to the receiver, had there been no caching and no interference.

[7]This is also presented in the preliminary version of this work in [4].

Remark 1. *We note that the converse that supports Theorem 1, encompasses the class of all caching-and-delivery schemes χ that employ uncoded cache placement under a general sum cache constraint $\frac{1}{\Lambda}\sum_{\lambda=1}^{\Lambda}|\mathcal{Z}_\lambda| = M$ which does not* necessarily *impose an individual cache size constraint. The converse also encompasses all scenarios that involve a library of size $\sum_{n\in[N]}|W^n| = N$ but where the files may be of different size. In the end, even though the designed optimal scheme will consider an individual cache size M and equal file sizes, the converse guarantees that there cannot exist a scheme (even in settings with uneven cache sizes or uneven file sizes) that exceeds the optimal performance identified here.*

From Theorem 1, we see that in the uniform case[8] where $\boldsymbol{\mathcal{L}} = (\frac{K}{\Lambda}, \frac{K}{\Lambda}, \dots, \frac{K}{\Lambda})$, the expression in (3) reduces to

$$T^*(\boldsymbol{\mathcal{L}}) = \frac{K(1-\gamma)}{\Lambda\gamma + 1}$$

matching the achievable delay presented in [31]. It also matches the recent result by [37] which proved that this performance — in the context of the multiple file request problem — is optimal under the assumption of uncoded cache placement.

The following corollary relates to this uniform case.

Corollary 1. *In the uniform user-to-cache association case where $\boldsymbol{\mathcal{L}} = (\frac{K}{\Lambda}, \frac{K}{\Lambda}, \dots, \frac{K}{\Lambda})$, the aforementioned optimal delay $T^*(\boldsymbol{\mathcal{L}}) = \frac{K(1-\gamma)}{\Lambda\gamma+1}$ is smaller than the corresponding delay $T^*(\boldsymbol{\mathcal{L}})$ for any other non-uniform class.*

Proof. The proof that the uniform profile results in the smallest delay among all profiles, follows directly from the fact that in (3), both $\mathcal{L}_r$ and $\binom{\Lambda-r}{\Lambda\gamma}$ are non-increasing with r. □

B. Multi-antenna/Multi-sender Coded Caching with Shared Caches

The following extends Theorem 1 to the case where the transmitter is equipped with multiple ($N_0 > 1$) antennas. The results hold for any $\boldsymbol{\mathcal{L}}$ as long as any non zero $\mathcal{L}_\lambda$ satisfies $\mathcal{L}_\lambda \geq N_0,\ \forall\lambda \in [\Lambda]$.

Theorem 2. *In the N_0-antenna K-user broadcast channel with Λ shared caches of normalized size γ, the optimal delivery time within any class/profile $\boldsymbol{\mathcal{L}}$ is*

$$T^*(\boldsymbol{\mathcal{L}}, N_0) = \frac{1}{N_0} Conv\left(\frac{\sum_{r=1}^{\Lambda-\Lambda\gamma}\mathcal{L}_r\binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}}\right) \quad (4)$$

for $\gamma \in \left\{\frac{1}{\Lambda}, \frac{2}{\Lambda}, \dots, 1\right\}$. This reveals a multiplicative gain of N_0 with respect to the single antenna case.

Proof. The scheme that achieves (4) is presented in Section IV, and the converse is presented in Section V.

The following extends Corollary 1 to the multi-antenna case, and the proof is direct from Theorem 2.

Corollary 2. *In the uniform user-to-cache association case of $\boldsymbol{\mathcal{L}} = \left(\frac{K}{\Lambda}, \frac{K}{\Lambda}, \dots, \frac{K}{\Lambda}\right)$ where $N_0 \leq \frac{K}{\Lambda}$, the optimal delay is*

$$T^*(\boldsymbol{\mathcal{L}}) = \frac{K(1-\gamma)}{N_0(\Lambda\gamma + 1)} \quad (5)$$

and it is smaller than the corresponding delay $T^*(\boldsymbol{\mathcal{L}})$ for any other non-uniform class.

Remark 2 (Shared-link coded caching with multiple file requests)**.** *In the error-free shared-link case ($N_0 = 1$), with file-independence and worst-case demand assumptions, i.e., when each requests a different file, the shared-cache problem here is closely related to the coded caching problem with multiple file requests per user, where now Λ users with their own cache, request in total $K \geq \Lambda$ different files. In particular, a small format modification would now yield each demand vector $\boldsymbol{d} = (d_1, d_2, \dots, d_K)$ which would represent the vector of the indices of the K requested files, and each user $\lambda = \{1, 2, \dots, \Lambda\}$, would request those files from this vector* **d***, whose indices[9] form the set $\mathcal{U}_\lambda \subset [K]$. At this point, as before, the problem is now defined by the user-to-file association $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_\Lambda\}$ which describes — given a fixed demand vector $\boldsymbol{d}$ — the files requested by any user. From this point on, the equivalence with the original shared cache problem is complete. As before, each such $\mathcal{U}$ again has a corresponding (sorted) profile $\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_\Lambda)$, and belongs to a class $\mathcal{U}_{\boldsymbol{\mathcal{L}}}$ with all other associations $\mathcal{U}$ that share the same profile $\boldsymbol{\mathcal{L}}$. As we quickly show in the Appendix Section VII-H, our scheme and converse can be adapted to the multiple file request problem, and thus directly from Theorem 1 we conclude that for this multiple file request problem, the optimal delay $T^*(\boldsymbol{\mathcal{L}}) \triangleq \min_\chi \max_{(\mathcal{U},\boldsymbol{d})\in(\mathcal{U}_{\boldsymbol{\mathcal{L}}},\{1,\dots,N\}^K)} T(\mathcal{U}, \boldsymbol{d}, \chi)$ corresponding to any user-to-file association profile $\boldsymbol{\mathcal{L}}$, takes the form $T^*(\boldsymbol{\mathcal{L}}) = Conv\left(\frac{\sum_{r=1}^{\Lambda-\Lambda\gamma}\mathcal{L}_r\binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}}\right)$. At this point we close the parenthesis regarding multiple file requests, and we refocus exclusively on the problem of shared caches.*

C. Interpretation of Results

1) Capturing the effect of the user-to-cache association profile: In a nutshell, Theorems 1,2 quantify how profile non-uniformities bring about increased delays. What we see is that, the more skewed the profile is, the larger is the delay. This is reflected in Figure 2 which shows — for a setting with $K = 30$ users and $\Lambda = 6$ caches — the memory-delay trade-off curves for different user-to-cache association profiles. As expected, Figure 2 demonstrates that when all users are connected to the same helper cache, the only gain arising from caching is the well known *local caching gain*. On the other hand, when users are assigned uniformly among the caches (i.e., when $\mathcal{L}_\lambda = \frac{K}{\Lambda}, \forall\lambda \in [\Lambda]$) the caching gain is maximized and the delay is minimized.

2) A multiplicative reduction in delay: Theorem 2 states that, as long as each cache is associated to at least N_0 users,

[8]Here, this uniform case, naturally implies that $\Lambda | K$.

[9]For example, having $\mathcal{U}_2 = \{3, 5, 7\}$, means that user 2 has requested files $W^{d_3}, W^{d_5}, W^{d_7}$.

	Dedicated Caches		Shared Caches/ Multiple File Requests		Shared Caches	
	Single Antenna	Multiple Antennas	Single Antenna		Multiple Antennas	
User-to-cache association profile			uniform	not uniform	uniform	not uniform
Achievable delay	$\frac{K(1-\gamma)}{K\gamma+1}$ [1]	$\frac{K(1-\gamma)}{K\gamma+N_0}$ [38]	$\frac{K(1-\gamma)}{\Lambda\gamma+1}$ [37]	$\frac{\sum_{r=1}^{\Lambda-\Lambda\gamma}\mathcal{L}_r\binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}}$ (*)	$\frac{K(1-\gamma)}{N_0(\Lambda\gamma+1)}$ (*)	$\frac{\sum_{r=1}^{\Lambda-\Lambda\gamma}\mathcal{L}_r\binom{\Lambda-r}{\Lambda\gamma}}{N_0\binom{\Lambda}{\Lambda\gamma}}$ (*)
Optimality	gap of 2 [39] and optimality under uncoded prefetching [2], [3]	gap of 2 under one-shot linear schemes [40]	optimal under uncoded prefetching [37]	optimal under uncoded prefetching (*)	optimal under uncoded prefetching and $L_\lambda \geq N_0$ (*)	

TABLE I: The table above provides an overview of the optimal worst-case performance of the state-of-the-art algorithms for the cache-aided shared-link setting with dedicated and shared caches. The cells with the symbol (*) show the contributions of this paper.

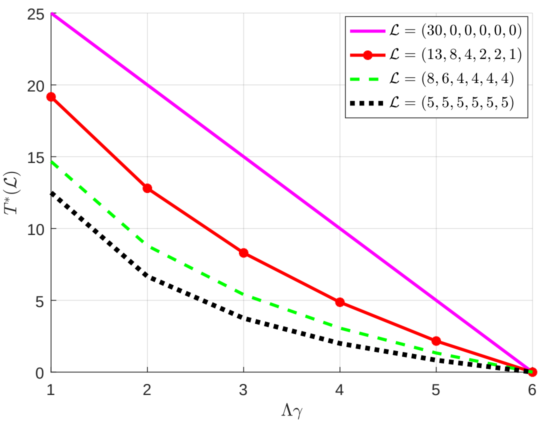

Fig. 2: Optimal delay for different user-to-cache association profiles $\boldsymbol{\mathcal{L}}$, for $K = 30$ users and $\Lambda = 6$ caches.

we can achieve a delay $T(\boldsymbol{\mathcal{L}}, N_0) = \frac{1}{N_0}\frac{\sum_{r=1}^{\Lambda-\Lambda\gamma}\mathcal{L}_r\binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}}$. The resulting reduction

$$\frac{T(\boldsymbol{\mathcal{L}}, N_0 = 1)}{T(\boldsymbol{\mathcal{L}}, N_0)} = N_0 \quad (6)$$

as compared to the single-stream case, comes in strong contrast to the case of $\Lambda = K$ where, as we know from [38], this same reduction takes the form

$$\frac{T(\Lambda = K, N_0 = 1)}{T(\Lambda = K, N_0)} = \frac{\frac{K(1-\gamma)}{1+\Lambda\gamma}}{\frac{K(1-\gamma)}{N_0+\Lambda\gamma}} = \frac{N_0 + \Lambda\gamma}{1 + \Lambda\gamma} \quad (7)$$

which approaches N_0 only when $\gamma \to 0$, and which decreases as γ increases.

In the uniform case ($\mathcal{L}_\lambda = \frac{K}{\Lambda}$) with $\Lambda \leq \frac{K}{N_0}$, Corollary 2 implies a sum-DoF

$$d_\Sigma(\gamma) = \frac{K(1-\gamma)}{T} = N_0(1 + \Lambda\gamma)$$

which reveals that every time we add a single degree of cache-redundancy (i.e., every time we increase $\Lambda\gamma$ by one), we gain N_0 degrees of freedom. This is in direct contrast to the case of $\Lambda = K$ (for which case we recall from [38] that the DoF is $N_0 + \Lambda\gamma$) where the same unit increase in the cache redundancy yields only one additional DoF.

3) Impact of encoding over users that share the same cache: As we know, both the MN algorithm in [1] and the multi-antenna algorithm in [38], are designed for users with different caches, so — in the uniform case where $\mathcal{L}_\lambda = K/\Lambda$ — one conceivable treatment of the shared-cache problem would have been to apply these algorithms over Λ users at a time, all with different caches[10]. As we see, in the single antenna case, this implementation would treat $1 + \Lambda\gamma$ users at a time thus yielding a delay of $T = \frac{K(1-\gamma)}{1+\Lambda\gamma}$, while in the multi-antenna case, this implementation would treat $N_0 + \Lambda\gamma$ users at a time (see [38]) thus yielding a delay of $T = \frac{K(1-\gamma)}{N_0+\Lambda\gamma}$. What we see here is that while this direct implementation is optimal (this is what we also do here in the uniform-profile case) in the single antenna case (see [37], see also Corollary 1), in the multi-antenna case, this same approach can have an unbounded performance gap

$$\frac{\frac{K(1-\gamma)}{N_0+\Lambda\gamma}}{\frac{K(1-\gamma)}{N_0(1+\Lambda\gamma)}} = \frac{N_0(1 + \Lambda\gamma)}{N_0 + \Lambda\gamma} \quad (8)$$

from the derived optimal performance from Corollary 2. These conclusions also apply when the user-to-cache association profiles are not uniform; again there would be a direct implementation of existing multi-antenna coded caching algorithms, which would though again have an unbounded performance gap from the optimal performance achieved here.

IV. Coded Caching Scheme

This section is dedicated to the description of the placement-and-delivery scheme achieving the performance presented in the general Theorem 2 (and hence also in Theorem 1 and the corollaries). The formal description of the optimal scheme in the upcoming subsection will be followed by two clarifying examples in Section IV-D and Section IV-E that demonstrate the main idea behind the design.

A. Description of the General Scheme

The placement phase, which uses exactly the algorithm developed in [1] for the case of $(\Lambda = K, M, N)$, is independent of $\mathcal{U}, \boldsymbol{\mathcal{L}}$, while the delivery phase is designed for any given $\mathcal{U}$, and will achieve the optimal worst-case delivery times stated in (3) and (4). As mentioned, we will assume that any non zero $\mathcal{L}_\lambda$ satisfies $\mathcal{L}_\lambda \geq N_0, \forall\lambda \in [\Lambda]$.

1) Cache Placement Phase : The placement phase employs the original cache-placement algorithm of [1] corresponding to the scenario of having only Λ users, each with their own cache. Hence — recalling from [1] — first each file W^n is split into $\binom{\Lambda}{\Lambda\gamma}$ disjoint subfiles $W^n_{\mathcal{T}}$, for

[10]This would then require $\frac{K}{\Lambda}$ such rounds in order to cover all K users.

each $\mathcal{T} \subset [\Lambda]$, $|\mathcal{T}| = \Lambda\gamma$, and then each cache stores a fraction γ of each file, as follows

$$\mathcal{Z}_\lambda = \{W^n_{\mathcal{T}} : \mathcal{T} \ni \lambda, \ \forall n \in [N]\}. \tag{9}$$

2) Delivery Phase: For the purpose of the scheme description only, we will assume without loss of generality that $|\mathcal{U}_1| \geq |\mathcal{U}_2| \geq \cdots \geq |\mathcal{U}_\Lambda|$ (any other case can be handled by simple relabeling of the caches), and we will use the notation $\mathcal{L}_\lambda \triangleq |\mathcal{U}_\lambda|$. Furthermore, in a slight abuse of notation, we will consider here each $\mathcal{U}_\lambda$ to be an *ordered vector* describing, in order, the users associated to cache λ. We will also use

$$\boldsymbol{s_\lambda} = (\mathcal{U}_\lambda \| \mathcal{U}_\lambda)_{N_0}, \lambda \in [\Lambda] \tag{10}$$

to denote the N_0-fold concatenation of each $\mathcal{U}_\lambda$. Each such $N_0\mathcal{L}_\lambda$-length vector $\boldsymbol{s_\lambda}$ can be seen as the concatenation of $\mathcal{L}_\lambda$ different N_0-tuples $\boldsymbol{s_{\lambda,j}}$, $j = 1, 2, \ldots, \mathcal{L}_\lambda$, i.e., each $\boldsymbol{s_\lambda}$ takes the form[11]

$$\boldsymbol{s_\lambda} = \boldsymbol{s_{\lambda,1}} \| \boldsymbol{s_{\lambda,2}} \| \ldots \| \underbrace{\boldsymbol{s_{\lambda,\mathcal{L}_\Lambda}}}_{N_0-\text{length}} .$$

The delivery phase commences with the demand vector $\boldsymbol{d}$ being revealed to the server. Delivery will consist of $\mathcal{L}_1$ rounds, where each round $j \in [\mathcal{L}_1]$ serves users

$$\mathcal{R}_j = \bigcup_{\lambda \in [\Lambda]} \big(\boldsymbol{s_{\lambda,j}} : \mathcal{L}_\lambda \geq j\big). \tag{11}$$

Transmission scheme: Once the demand vector $\boldsymbol{d}$ is revealed to the transmitter, each requested subfile $W^n_{\mathcal{T}}$ (for any n found in $\boldsymbol{d}$) is further split into N_0 mini-files $\{W^n_{\mathcal{T},l}\}_{l \in [N_0]}$. During round j, serving users in $\mathcal{R}_j$, we create $\binom{\Lambda}{\Lambda\gamma+1}$ sets $\mathcal{Q} \subseteq [\Lambda]$ of size $|\mathcal{Q}| = \Lambda\gamma + 1$, and for each set $\mathcal{Q}$, we pick the set of users

$$\chi_{\mathcal{Q}} = \bigcup_{\lambda \in \mathcal{Q}} \big(\boldsymbol{s_{\lambda,j}} : \mathcal{L}_\lambda \geq j\big). \tag{12}$$

If $\chi_{\mathcal{Q}} = \emptyset$, then there is no transmission, and we move to the next $\mathcal{Q}$. If $\chi_{\mathcal{Q}} \neq \emptyset$, the server — during this round j — transmits the following vector[12]

$$\boldsymbol{x}_{\chi_{\mathcal{Q}}} = \sum_{\lambda \in \mathcal{Q}: \mathcal{L}_\lambda \geq j} \mathbf{H}^{-1}_{\boldsymbol{s_{\lambda,j}}} \times \Big[W^{d_{\boldsymbol{s_{\lambda,j}}(1)}}_{\mathcal{Q}\backslash\{\lambda\},l_1} \ \ldots \ W^{d_{\boldsymbol{s_{\lambda,j}}(k)}}_{\mathcal{Q}\backslash\{\lambda\},l_k} \ \ldots \ W^{d_{\boldsymbol{s_{\lambda,j}}(N_0)}}_{\mathcal{Q}\backslash\{\lambda\},l_{N_0}}\Big]^T \tag{13}$$

where $W^{d_{\boldsymbol{s_{\lambda,j}}(k)}}_{\mathcal{Q}\backslash\{\lambda\},l_k}$ is a mini-file intended for user $\boldsymbol{s_{\lambda,j}}(k)$, i.e., for the user labelled by the kth entry of vector $\boldsymbol{s_{\lambda,j}}$. The choice of each $l_k \in [N_0]$ is sequential, guaranteeing that no mini-file $W^{d_{\boldsymbol{s_{\lambda,j}}(k)}}_{\mathcal{Q}\backslash\{\lambda\},l_k}$ of the same subfile $W^{d_{\boldsymbol{s_{\lambda,j}}(k)}}_{\mathcal{Q}\backslash\{\lambda\}}$ is transmitted twice. In the above, $\mathbf{H}^{-1}_{\boldsymbol{s_{\lambda,j}}}$ denotes the inverse of the channel matrix between the N_0 transmit antennas and the users in vector $\boldsymbol{s_{\lambda,j}}$.

Since each user appears in $\boldsymbol{s_\lambda}$ (and consequently in $\bigcup_{j \in [\mathcal{L}_1]} \mathcal{R}_j$) exactly N_0 times, at the end of the $\mathcal{L}_1$ rounds, all the N_0 mini-files $W^{d_{\boldsymbol{s_{\lambda,j}}(k)}}_{\mathcal{Q}\backslash\{\lambda\},l_k}$, $l_k \in [N_0]$ will be sent once. This, along with the fact that the sets $\mathcal{Q}$ follow the standard MAN scheme in [1], guarantees that the scheme serves all subfiles requested by the users.

Decoding: Directly from (13), we see that each receiver $\boldsymbol{s_{\lambda,j}}(k)$ obtains a received signal whose noiseless version takes the form

$$y_{\boldsymbol{s_{\lambda,j}}(k)} = W^{d_{\boldsymbol{s_{\lambda,j}}(k)}}_{\mathcal{Q}\backslash\{\lambda\},l_k} + \iota_{\boldsymbol{s_{\lambda,j}}(k)}$$

where $\iota_{\boldsymbol{s_{\lambda,j}}(k)}$ is the kth entry of the interference vector

$$\sum_{\substack{\lambda' \in \mathcal{Q}\backslash\{\lambda\}: \\ \mathcal{L}_{\lambda'} \geq j}} \mathbf{H}^{-1}_{\boldsymbol{s_{\lambda',j}}} \cdot \Big[W^{d_{\boldsymbol{s_{\lambda',j}}(1)}}_{\mathcal{Q}\backslash\{\lambda'\},l_1} \ldots W^{d_{\boldsymbol{s_{\lambda,j}}(k)}}_{\mathcal{Q}\backslash\{\lambda\},l_k} \ldots W^{d_{\boldsymbol{s_{\lambda',j}}(N_0)}}_{\mathcal{Q}\backslash\{\lambda'\},l_{N_0}}\Big]^T. \tag{14}$$

In the above, we see that the entire interference term $\iota_{\boldsymbol{s_{\lambda,j}}(k)}$ experienced by receiver $\boldsymbol{s_{\lambda,j}}(k)$, can be removed (cached-out) because all appearing subfiles $W^{d_{\boldsymbol{s_{\lambda',j}}(1)}}_{\mathcal{Q}\backslash\{\lambda'\},l_1}, \ldots, W^{d_{\boldsymbol{s_{\lambda',j}}(N_0)}}_{\mathcal{Q}\backslash\{\lambda'\},l_{N_0}}$, for all $\lambda' \in \mathcal{Q} \setminus \{\lambda\}, \mathcal{L}_{\lambda'} \geq j$, can be found in cache λ associated to this user, simply because $\lambda \in \mathcal{Q}\backslash\{\lambda'\}$.

This completes the proof of the scheme for the multi-antenna case.

3) Small modification for the single antenna case: For the single-antenna case, the only difference is that now $\boldsymbol{s_\lambda} = \mathcal{U}_\lambda$, and that each transmitted vector in (13) during round j, becomes a scalar of the form

$$x_{\chi_{\mathcal{Q}}} = \bigoplus_{\lambda \in \mathcal{Q}: \mathcal{L}_\lambda \geq j} W^{d_{\boldsymbol{s_{\lambda,j}}}}_{\mathcal{Q}\backslash\{\lambda\},1}. \tag{15}$$

The rest of the details from the general scheme, as well as the subsequent calculation of the delay, follow directly. We also notice that this same scheme for the single antenna case has been already proposed for a decentralized coded caching setting with reduced subpacketization in [41], [42].

B. Calculation of Delay

To first calculate the delay needed to serve the users in $\mathcal{R}_j$ during round j, we recall that there are $\binom{\Lambda}{\Lambda\gamma+1}$ sets

$$\chi_{\mathcal{Q}} = \bigcup_{\lambda \in \mathcal{Q}} \big(\mathcal{U}_\lambda(j) : \mathcal{L}_\lambda \geq j\big), \mathcal{Q} \subseteq [\Lambda]$$

of users, and we recall that $|\mathcal{U}_1| \geq |\mathcal{U}_2| \geq \cdots \geq |\mathcal{U}_\Lambda|$. For each such non-empty set, there is a transmission. Furthermore we see that for $a_j \triangleq \Lambda - \frac{|\mathcal{R}_j|}{N_0}$, there are $\binom{a_j}{\Lambda\gamma+1}$ such sets $\chi_{\mathcal{Q}}$ which are empty, which means that round j consists of

$$\binom{\Lambda}{\Lambda\gamma+1} - \binom{a_j}{\Lambda\gamma+1} \tag{16}$$

transmissions.

Since each file is split into $\binom{\Lambda}{\Lambda\gamma}N_0$ subfiles, the duration of each such transmission is

$$\frac{1}{\binom{\Lambda}{\Lambda\gamma}N_0} \tag{17}$$

and thus summing over all $\mathcal{L}_1$ rounds, the total delay takes the form

$$T = \frac{\sum_{j=1}^{\mathcal{L}_1} \binom{\Lambda}{\Lambda\gamma+1} - \binom{a_j}{\Lambda\gamma+1}}{\binom{\Lambda}{\Lambda\gamma}N_0} \tag{18}$$

[11] Note also that having $\mathcal{L}_\lambda \geq N_0, \forall\lambda \in [\Lambda]$ guarantees that in any given $\boldsymbol{s_{\lambda,j}}, j \in [\mathcal{L}_\lambda]$, a user appears at most once.

[12] The transmitted-vector structure below draws from the structure in [28], in the sense that it involves the linear combination of one or more *Zero Forcing* precoded (ZF-precoded) vectors of subfiles that are labeled (as we see below) in the spirit of [1].

which, after some basic algebraic manipulation (see Appendix VII-G for the details), takes the final form

$$T = \frac{1}{N_0} \frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \mathcal{L}_r \binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} \tag{19}$$

which concludes the achievability part of the proof. $\square$

C. Intuition on the scheme: separability and the parallel version of the multi-transmitter BC with shared caches

The scheme described in Section IV-A makes use of a *non-separable* approach that is based on multicast messages that simultaneously exploit the caches and the channel. To gain some insight into this scheme, we will briefly look at a simpler (but separable) solution to the easier parallel version of the multi-transmitter BC with shared caches.

N_0-transmitter parallel BC with shared caches: This parallel version of the problem consists of N_0 transmitters, each connected to all K receivers via a broadcast link of normalized unit capacity. The main difference in this parallel version is that naturally each receiver enjoys N_0 simultaneous signal observations, one from each transmitter. As in our setting, each receiver has access to one of Λ caches, as defined by the association profiles.

In this parallel version, an optimal scheme would consist of a cache placement that is identical to ours as in Section IV-A1, and a delivery phase which begins by generating the same set of XORs that we would generate for the single-antenna case as in Section IV-A3, then partitioning this set into any N_0 equal-sized subsets[13], and finally the delivery phase would conclude by sending each of the N_0 subsets, simultaneously, from each of the transmitters. Decoding is done exactly as in the single-antenna case. This achieves a speed-up factor of N_0 over the single-antenna case, for any association profile $\boldsymbol{\mathcal{L}}$ (including for the case where $L_\lambda \leq N_0$). As it will become clear later, the lower bound in Section V applies also to this parallel channel, and we can thus conclude that the optimal performance for the parallel channel coincides with the derived performance for the MISO BC for $L_\lambda \geq N_0$.

a) Non-separability of the scheme for the MISO-BC with shared caches: To draw the connection to our scheme, we first note that the above described *parallel* version is separable; the scheme consists of a) a cache placement and multicast message generation that are independent of the network/channel and are identical to the single-transmitter case, and b) an efficient delivery (now as a function of the network) of these multicast messages. However, this is not the case for the MISO BC setting of interest here, where the multicast messages themselves are a function of the channel. The difference here is that, to the best of our understanding, we could not have used the single-stream XORs generated in Section IV-A3, and then simply found a way (e.g. via beamforming) to deliver these over a parallelized channel, where this channel was parallelized let's say by use of ZF[14]. Instead, what we do is that we consider the same sequence of XORs as in the parallel channel setting, we split these into the same N_0 subsets, but then instead of considering XORs, we decompose each XOR, and linearly combine the subfiles within each such XOR, as a function of the channel, in a way that the created multicast messages — in addition to exploiting the caches of the users addressed by each such multicast message — also simultaneously orthogonalize the channels to another set of users that share the same cache. This approach makes the scheme non-separable. Whether it is possible or not to develop a fully separable optimal solution for the MISO BC with shared caches, remains an open question.

[13] Without loss of generality, we here assume that the number of XORs is a multiple of the number of antennas. The general case can be handled with a small increase in the subpacketization.

[14] This separable approach though can indeed work (cf. [38]) if each user has their own cache.

D. Scheme Example: $K = N = 15$, $\Lambda = 3$, $N_0 = 2$ and $\boldsymbol{\mathcal{L}} = (8, 5, 2)$

Consider a scenario with $K = 15$ users $\{1, 2, \ldots, 15\}$, a server equipped with $N_0 = 2$ transmitting antennas that stores a library of $N = 15$ equally-sized files $W^1, W^2, \ldots, W^{15}$, and consider $\Lambda = 3$ helper caches, each of size equal to $M = 5$ units of file.

In the cache placement phase, we split each file W^n into 3 equally-sized disjoint subfiles denoted by W_1^n, W_2^n, W_3^n and as in [1], each cache λ stores $W_\lambda^n, \forall n \in [15]$.

We assume that in the subsequent cache assignment, users $\mathcal{U}_1 = (1, 2, 3, 4, 5, 6, 7, 8)$ are assigned to helper node 1, users $\mathcal{U}_2 = (9, 10, 11, 12, 13)$ to helper node 2 and users $\mathcal{U}_3 = (14, 15)$ to helper node 3. This corresponds to a profile $\boldsymbol{\mathcal{L}} = (8, 5, 2)$. We also assume without loss of generality that the demand vector is $\boldsymbol{d} = (1, 2, \ldots, 15)$.

Delivery takes place in $|\mathcal{U}_1| = 8$ rounds, and each round will serve either $N_0 = 2$ users or no users from each of the following three ordered user groups

$$\begin{aligned}
\boldsymbol{s_1} &= \mathcal{U}_1 || \mathcal{U}_1 = (1, 2, \ldots, 7, 8, 1, 2, \ldots, 7, 8), \\
\boldsymbol{s_2} &= \mathcal{U}_2 || \mathcal{U}_2 = (9, 10, 11, 12, 13, 9, 10, 11, 12, 13), \\
\boldsymbol{s_3} &= \mathcal{U}_3 || \mathcal{U}_3 = (14, 15, 14, 15).
\end{aligned}$$

Specifically, rounds 1 through 8, will respectively serve the following sets of users

$$\begin{aligned}
\mathcal{R}_1 &= \{1, 2, 9, 10, 14, 15\} \\
\mathcal{R}_2 &= \{3, 4, 11, 12, 14, 15\} \\
\mathcal{R}_3 &= \{5, 6, 13, 9\} \\
\mathcal{R}_4 &= \{7, 8, 10, 11\} \\
\mathcal{R}_5 &= \{1, 2, 12, 13\} \\
\mathcal{R}_6 &= \{3, 4\} \\
\mathcal{R}_7 &= \{5, 6\} \\
\mathcal{R}_8 &= \{7, 8\}.
\end{aligned}$$

Before transmission, each requested subfile $W_{\mathcal{T}}^n$ is further split into $N_0 = 2$ mini-files $W_{\mathcal{T},1}^n$ and $W_{\mathcal{T},2}^n$. As noted in the general description of the scheme, the transmitted vector structure within each round, draws from [28] as it employs the linear combination of ZF-precoded vectors. In the first round, the server transmits, one after the other, the following 3 vectors

$$\boldsymbol{x}_{\{1,2,9,10\}} = \mathbf{H}^{-1}_{\{1,2\}} \begin{bmatrix} W^1_{2,1} \\ W^2_{2,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{9,10\}} \begin{bmatrix} W^9_{1,1} \\ W^{10}_{1,1} \end{bmatrix} \quad (20)$$

$$\boldsymbol{x}_{\{1,2,14,15\}} = \mathbf{H}^{-1}_{\{1,2\}} \begin{bmatrix} W^1_{3,1} \\ W^2_{3,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{14,15\}} \begin{bmatrix} W^{14}_{1,1} \\ W^{15}_{1,1} \end{bmatrix} \quad (21)$$

$$\boldsymbol{x}_{\{9,10,14,15\}} = \mathbf{H}^{-1}_{\{9,10\}} \begin{bmatrix} W^9_{3,1} \\ W^{10}_{3,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{14,15\}} \begin{bmatrix} W^{14}_{2,1} \\ W^{15}_{2,1} \end{bmatrix} \quad (22)$$

where $\mathbf{H}^{-1}_{\{i,j\}}$ is the zero-forcing (ZF) precoder[15] that inverts the channel $\mathbf{H}_{\{i,j\}} = [\mathbf{h}_i^T \mathbf{h}_j^T]$ from the transmitter to users i and j. To see how decoding takes place, let us first focus on users 1 and 2 during the transmission of $\boldsymbol{x}_{\{1,2,9,10\}}$, where we see that, due to ZF precoding, the users' respective received signals take the form

$$y_1 = W^1_{2,1} + \underbrace{\mathbf{h}_1^T \mathbf{H}^{-1}_{\{9,10\}} \begin{bmatrix} W^9_{1,1} \\ W^{10}_{1,1} \end{bmatrix}}_{\text{interference}} + w_1 \quad (23)$$

$$y_2 = W^2_{2,1} + \underbrace{\mathbf{h}_2^T \mathbf{H}^{-1}_{\{9,10\}} \begin{bmatrix} W^9_{1,1} \\ W^{10}_{1,1} \end{bmatrix}}_{\text{interference}} + w_2. \quad (24)$$

Users 1 and 2 use their cached content in cache node 1, to remove files $W^9_{1,1}, W^{10}_{1,1}$, and can thus directly decode their own desired subfiles. The same procedure is applied to the remaining users served in the first round.

Similarly, in the second round, we have

$$\boldsymbol{x}_{\{3,4,11,12\}} = \mathbf{H}^{-1}_{\{3,4\}} \begin{bmatrix} W^3_{2,1} \\ W^4_{2,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{11,12\}} \begin{bmatrix} W^{11}_{1,1} \\ W^{12}_{1,1} \end{bmatrix} \quad (25)$$

$$\boldsymbol{x}_{\{3,4,14,15\}} = \mathbf{H}^{-1}_{\{3,4\}} \begin{bmatrix} W^3_{3,1} \\ W^4_{3,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{14,15\}} \begin{bmatrix} W^{14}_{1,2} \\ W^{15}_{1,2} \end{bmatrix} \quad (26)$$

$$\boldsymbol{x}_{\{11,12,14,15\}} = \mathbf{H}^{-1}_{\{11,12\}} \begin{bmatrix} W^{11}_{3,1} \\ W^{12}_{3,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{14,15\}} \begin{bmatrix} W^{14}_{2,2} \\ W^{15}_{2,2} \end{bmatrix} \quad (27)$$

and again in each round, each pair of users can cache-out some of the files, and then decode their own file due to the ZF precoder.

The next three transmissions, corresponding to the third round, are as follows

$$\boldsymbol{x}_{\{5,6,13,9\}} = \mathbf{H}^{-1}_{\{5,6\}} \begin{bmatrix} W^5_{2,1} \\ W^6_{2,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{13,9\}} \begin{bmatrix} W^{13}_{1,1} \\ W^9_{1,2} \end{bmatrix} \quad (28)$$

$$\boldsymbol{x}_{\{5,6\}} = \mathbf{H}^{-1}_{\{5,6\}} \begin{bmatrix} W^5_{3,1} \\ W^6_{3,1} \end{bmatrix} \quad (29)$$

$$\boldsymbol{x}_{\{13,9\}} = \mathbf{H}^{-1}_{\{13,9\}} \begin{bmatrix} W^{13}_{3,1} \\ W^9_{3,2} \end{bmatrix} \quad (30)$$

where the transmitted vectors $\boldsymbol{x}_{\{5,6\}}$ and $\boldsymbol{x}_{\{13,9\}}$ simply use zero-forcing. Similarly round 4 serves the users in $\mathcal{R}_4$ by sequentially sending

$$\boldsymbol{x}_{\{7,8,10,11\}} = \mathbf{H}^{-1}_{\{7,8\}} \begin{bmatrix} W^7_{2,1} \\ W^8_{2,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{10,11\}} \begin{bmatrix} W^{10}_{1,2} \\ W^{11}_{1,2} \end{bmatrix} \quad (31)$$

$$\boldsymbol{x}_{\{7,8\}} = \mathbf{H}^{-1}_{\{7,8\}} \begin{bmatrix} W^7_{3,1} \\ W^8_{3,1} \end{bmatrix} \quad (32)$$

$$\boldsymbol{x}_{\{10,11\}} = \mathbf{H}^{-1}_{\{10,11\}} \begin{bmatrix} W^{10}_{3,2} \\ W^{11}_{3,2} \end{bmatrix} \quad (33)$$

[15]Instead of ZF, one can naturally use a similar precoder with potentially better performance in different SNR ranges.

and round 5 serves the users in $\mathcal{R}_5$ by sequentially sending

$$\boldsymbol{x}_{\{1,2,12,13\}} = \mathbf{H}^{-1}_{\{1,2\}} \begin{bmatrix} W^1_{2,2} \\ W^2_{2,2} \end{bmatrix} + \mathbf{H}^{-1}_{\{12,13\}} \begin{bmatrix} W^{12}_{1,2} \\ W^{13}_{1,2} \end{bmatrix} \quad (34)$$

$$\boldsymbol{x}_{\{1,2\}} = \mathbf{H}^{-1}_{\{1,2\}} \begin{bmatrix} W^1_{3,2} \\ W^2_{3,2} \end{bmatrix} \quad (35)$$

$$\boldsymbol{x}_{\{12,13\}} = \mathbf{H}^{-1}_{\{12,13\}} \begin{bmatrix} W^{12}_{3,2} \\ W^{13}_{3,2} \end{bmatrix}. \quad (36)$$

Finally, for the remaining rounds $6, 7, 8$ which respectively involve user sets $\mathcal{R}_6, \mathcal{R}_7$ and $\mathcal{R}_8$ that are connected to the same helper cache 1, data is delivered using the following standard ZF-precoded transmissions

$$\boldsymbol{x}_{\{3,4\}} = \mathbf{H}^{-1}_{\{3,4\}} \begin{bmatrix} W^3_{2,2} || W^3_{3,2} \\ W^4_{2,2} || W^4_{3,2} \end{bmatrix} \quad (37)$$

$$\boldsymbol{x}_{\{5,6\}} = \mathbf{H}^{-1}_{\{5,6\}} \begin{bmatrix} W^5_{2,2} || W^5_{3,2} \\ W^6_{2,2} || W^6_{3,2} \end{bmatrix} \quad (38)$$

$$\boldsymbol{x}_{\{7,8\}} = \mathbf{H}^{-1}_{\{7,8\}} \begin{bmatrix} W^7_{2,2} || W^7_{3,2} \\ W^8_{2,2} || W^8_{3,2} \end{bmatrix}. \quad (39)$$

The overall delivery time required to serve all users is

$$T = \frac{1}{6} \cdot 15 + \frac{1}{3} \cdot 3 = \frac{21}{6}$$

where the first summand is for rounds 1 through 5, and the second summand is for rounds 6 through 8.

It is very easy to see that this delay remains the same — given again worst-case demand vectors — for any user-to-cache association $\mathcal{U}$ with the same profile $\boldsymbol{\mathcal{L}} = (8, 5, 2)$. Every time, this delay matches the converse

$$T^*((8,5,2)) \geq \frac{\sum_{r=1}^{2} \mathcal{L}_r \binom{3-r}{1}}{2\binom{3}{1}} = \frac{8 \cdot 2 + 5 \cdot 1}{6} = \frac{21}{6} \quad (40)$$

of Theorem 2.

We proceed with a second example which gives further insight. This example can be later combined with the example in Section V-B which describes the information theoretic converse for the same set of parameters.

E. Scheme Example: $K = N = 9$, $\Lambda = 3$, $N_0 = 2$ and $\boldsymbol{\mathcal{L}} = (4, 3, 2)$

Consider a setting with $K = 9$ users $\{1, 2, \ldots, 9\}$, a server equipped with $N_0 = 2$ transmitting antennas that has access to a library of $N = 9$ equally-sized files $W^1, W^2, \ldots, W^9$. Let us also consider $\Lambda = 3$ different caches, each of size equal to $M = 6$ units of file.

In the cache placement phase, we split each file W^n into 3 equally-sized subfiles denoted by $W^n_{12}, W^n_{13}, W^n_{23}$ such that each cache λ stores $W^n_{\mathcal{T}}, \forall n \in [9]$ and $\lambda \in \mathcal{T}$.

In the subsequent cache assignment, we assume that users $\mathcal{U}_1 = (1, 2, 3, 4)$ are assigned to cache 1, users $\mathcal{U}_2 = (5, 6, 7)$ to cache 2 and users $\mathcal{U}_3 = (8, 9)$ to cache 3, resulting in the profile $\boldsymbol{\mathcal{L}} = (4, 3, 2)$. We also assume without loss of generality that the demand vector is $\boldsymbol{d} = (1, 2, \ldots, 9)$.

Following the general delivery scheme in Section IV-A2, the server will satisfy all users' requests in $|\mathcal{U}_1| = 4$ rounds.

Specifically, rounds 1 through 4, will respectively serve the following sets of users

$$\mathcal{R}_1 = \{1,2,5,6,8,9\}$$
$$\mathcal{R}_2 = \{3,4,7,5,8,9\}$$
$$\mathcal{R}_3 = \{1,2,6,7\}$$
$$\mathcal{R}_4 = \{3,4\}.$$

Before proceeding with the transmission, each demanded subfile $W^n_{\mathcal{T}}$ is further split into $N_0 = 2$ mini-files $W^n_{\mathcal{T},1}$ and $W^n_{\mathcal{T},2}$. In the first round, the server transmits the following vector

$$\boldsymbol{x}_{\{1,2,5,6,8,9\}} = \mathbf{H}^{-1}_{\{1,2\}} \begin{bmatrix} W^1_{23,1} \\ W^2_{23,1} \end{bmatrix} + \\ + \mathbf{H}^{-1}_{\{5,6\}} \begin{bmatrix} W^5_{13,1} \\ W^6_{13,1} \end{bmatrix} + \mathbf{H}^{-1}_{\{8,9\}} \begin{bmatrix} W^8_{12,1} \\ W^9_{12,1} \end{bmatrix} \quad (41)$$

where $\mathbf{H}^{-1}_{\{i,j\}}$ is the zero-forcing (ZF) precoder that inverts the channel matrix $\mathbf{H}_{\{i,j\}} = [\mathbf{h}_i^T \mathbf{h}_j^T]$. Let us now focus on users 1 and 2 which, after the transmission of $\boldsymbol{x}_{\{1,2,5,6,8,9\}}$, respectively receive the signals

$$y_1 = W^1_{23,1} + \underbrace{\mathbf{h}_1^T \mathbf{H}^{-1}_{\{5,6\}} \begin{bmatrix} W^5_{13,1} \\ W^6_{13,1} \end{bmatrix} + \mathbf{h}_1^T \mathbf{H}^{-1}_{\{8,9\}} \begin{bmatrix} W^8_{12,1} \\ W^9_{12,1} \end{bmatrix}}_{\text{interference}} \quad (42)$$

$$y_2 = W^2_{23,1} + \underbrace{\mathbf{h}_2^T \mathbf{H}^{-1}_{\{5,6\}} \begin{bmatrix} W^5_{13,1} \\ W^6_{13,1} \end{bmatrix} + \mathbf{h}_2^T \mathbf{H}^{-1}_{\{8,9\}} \begin{bmatrix} W^8_{12,1} \\ W^9_{12,1} \end{bmatrix}}_{\text{interference}} \quad (43)$$

where the noise was neglected. Users 1 and 2 use their cached content in helper node 1, to remove subfiles $W^5_{13,1}, W^6_{13,1}, W^8_{12,1}, W^9_{12,1}$, and can thus directly decode their own desired subfiles. The exact same procedure is used by the remaining users served in the first round.

Similarly, in the second round, we have

$$\boldsymbol{x}_{\{3,4,7,5,8,9\}} = \mathbf{H}^{-1}_{\{3,4\}} \begin{bmatrix} W^3_{23,1} \\ W^4_{23,1} \end{bmatrix} + \\ + \mathbf{H}^{-1}_{\{7,5\}} \begin{bmatrix} W^7_{13,1} \\ W^5_{13,2} \end{bmatrix} + \mathbf{H}^{-1}_{\{8,9\}} \begin{bmatrix} W^8_{12,2} \\ W^9_{12,2} \end{bmatrix}. \quad (44)$$

The decoding follows again the previously described procedure.

The next transmission, corresponding to the third round, takes the form

$$\boldsymbol{x}_{\{1,2,6,7\}} = \mathbf{H}^{-1}_{\{1,2\}} \begin{bmatrix} W^1_{23,2} \\ W^2_{23,2} \end{bmatrix} + \mathbf{H}^{-1}_{\{6,7\}} \begin{bmatrix} W^6_{13,2} \\ W^7_{13,2} \end{bmatrix}. \quad (45)$$

Similarly the last round serves the users in $\mathcal{R}_4$ by simply sending

$$\boldsymbol{x}_{\{3,4\}} = \mathbf{H}^{-1}_{\{3,4\}} \begin{bmatrix} W^3_{23,2} \\ W^4_{23,2} \end{bmatrix} \quad (46)$$

which does not require any cached content to be successfully decoded by users 3 and 4.

The overall delivery time required to serve all users is

$$T = \frac{1}{6} \cdot 4 = \frac{2}{3}$$

where $\frac{1}{6}$ is the duration of each transmission and 4 are the number of transmitted vectors during the 4 rounds.

It is again easy to see that this delay remains the same — given worst-case demand vectors — for any user-to-cache association $\mathcal{U}$ with the same profile $\boldsymbol{\mathcal{L}} = (4,3,2)$. This delay matches the converse

$$T^*((4,3,2)) \geq \frac{\sum_{r=1}^{1} \mathcal{L}_r \binom{3-r}{2}}{2\binom{3}{2}} = \frac{4 \cdot 1}{6} = \frac{2}{3} \quad (47)$$

of Theorem 2.

V. Information Theoretic Converse

Toward proving Theorems 1 and 2, we develop a lower bound on the normalized delivery time in (2) for each given user-to-cache association profile $\boldsymbol{\mathcal{L}}$. The proof technique is based on the breakthrough in [2] which — for the case of $\Lambda = K$, where each user has their own cache — employed index coding to bound the performance of coded caching. Part of the challenge here will be to account for having shared caches, and mainly to adapt the index coding approach to reflect non-uniform user-to-cache association classes.

We will begin with lower bounding the normalized delivery time $T(\mathcal{U}, \boldsymbol{d}, \chi)$, for any user-to-cache association $\mathcal{U}$, demand vector $\boldsymbol{d}$ and a generic caching-delivery strategy χ.

Identifying the distinct problems: The caching problem is defined when the user-to-cache association $\mathcal{U} = \{\mathcal{U}_\lambda\}_{\lambda=1}^{\Lambda}$ and demand vector $\boldsymbol{d}$ are revealed. What we can easily see is that there are many combinations of $\{\mathcal{U}_\lambda\}_{\lambda=1}^{\Lambda}$ and $\boldsymbol{d}$ that jointly result in the same coded caching problem. After all, any permutation of the file indices requested by users assigned to the same cache, will effectively result in the same coded caching problem. As one can see, every *distinct* coded caching problem is fully defined by $\{\boldsymbol{d_\lambda}\}_{\lambda=1}^{\Lambda}$, where $\boldsymbol{d_\lambda}$ denotes the vector of file indices requested by the users in $\mathcal{U}_\lambda$, i.e., requested by the $|\mathcal{U}_\lambda|$ users associated to cache λ. The analysis is facilitated by reordering the demand vector $\boldsymbol{d}$ to take the form

$$\boldsymbol{d}(\mathcal{U}) \triangleq (\boldsymbol{d_1}, \ldots, \boldsymbol{d_\Lambda}). \quad (48)$$

Based on this, we define the set of worst-case demands associated to a given profile $\boldsymbol{\mathcal{L}}$, to be

$$\mathcal{D}_{\boldsymbol{\mathcal{L}}} = \{\boldsymbol{d}(\mathcal{U}) : \boldsymbol{d} \in \mathcal{D}_{wc}, \mathcal{U} \in \mathcal{U}_{\boldsymbol{\mathcal{L}}}\}$$

where $\mathcal{D}_{wc}$ is the set of demand vectors $\boldsymbol{d}$ whose K entries are all different (i.e., where $d_i \neq d_j,\ i,j \in [\Lambda],\ i \neq j$, corresponding to the case where all users request different files). We will convert each such coded caching problem into an index coding problem.

The corresponding index coding problem: To make the transition to the index coding problem, each requested file $W^{\boldsymbol{d_\lambda}(j)}$ is split into 2^Λ disjoint subfiles $W^{\boldsymbol{d_\lambda}(j)}_{\mathcal{T}}, \mathcal{T} \in 2^{[\Lambda]}$ where $\mathcal{T} \subset [\Lambda]$ indicates the set of helper nodes in which $W^{\boldsymbol{d_\lambda}(j)}_{\mathcal{T}}$ is cached[16]. Then — in the context of index coding — each subfile $W^{\boldsymbol{d_\lambda}(j)}_{\mathcal{T}}$ can be seen as being requested by a different user that has as side information

[16]Notice that by considering a subpacketization based on the power set $2^{[\Lambda]}$, and by allowing for any possible size of these subfiles, the generality of the result is preserved. Naturally, this does not impose any subpacketization related performance issues because this is done only for the purpose of creating a converse.

all the content $\mathcal{Z}_\lambda$ of the same helper node λ. Naturally, no subfile of the form $W_{\mathcal{T}}^{\boldsymbol{d}_{\boldsymbol{\lambda}}(j)}$, $\mathcal{T} \ni \lambda$ is requested, because helper node λ already has this subfile. Therefore the corresponding index coding problem is defined by $K2^{\Lambda-1}$ requested subfiles, and it is fully represented by the side-information graph $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$, where $\mathcal{V}_\mathcal{G}$ is the set of vertices (each vertex/node representing a different subfile $W_{\mathcal{T}}^{\boldsymbol{d}_{\boldsymbol{\lambda}}(j)}, \mathcal{T} \not\ni \lambda$) and $\mathcal{E}_\mathcal{G}$ is the set of direct edges of the graph. Following standard practice in index coding, a directed edge from node $W_{\mathcal{T}}^{\boldsymbol{d}_{\boldsymbol{\lambda}}(j)}$ to $W_{\mathcal{T}'}^{\boldsymbol{d}_{\boldsymbol{\lambda}'}(j')}$ exists if and only if $\lambda' \in \mathcal{T}$. For any given $\mathcal{U}$, $\boldsymbol{d}$ (and of course, for any scheme χ) the total delay T required for this index coding problem, is the completion time for the corresponding coded caching problem.

Lower bounding $T(\mathcal{U}, \boldsymbol{d}, \chi)$: We are interested in lower bounding $T(\mathcal{U}, \boldsymbol{d}, \chi)$ which represents the total delay required to serve the users for the index coding problem corresponding to the side-information graph $\mathcal{G}_{\mathcal{U},\boldsymbol{d}}$ defined by $\mathcal{U}, \boldsymbol{d}, \chi$ or equivalently by $\boldsymbol{d}(\mathcal{U}), \chi$.

In the next lemma, we remind the reader — in the context of our setting — the useful index-coding converse from [43].

Lemma 1. *(Cut-set-type converse [43]) For a given $\mathcal{U}, \boldsymbol{d}, \chi$, in the corresponding side information graph $\mathcal{G}_{\mathcal{U},\boldsymbol{d}} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ of the N_0-antenna MISO broadcast channel with $\mathcal{V}_\mathcal{G}$ vertices/nodes and $\mathcal{E}_\mathcal{G}$ edges, the following inequality holds*

$$T \geq \frac{1}{N_0} \sum_{\mathcal{V} \in \mathcal{V}_\mathcal{J}} |\mathcal{V}| \tag{49}$$

for every acyclic induced subgraph $\mathcal{J}$ of $\mathcal{G}_{\mathcal{U},\boldsymbol{d}}$, where $\mathcal{V}_\mathcal{J}$ denotes the set of nodes of the subgraph $\mathcal{J}$, and where $|\mathcal{V}|$ is the size of the message/subfile/node $\mathcal{V}$.

Proof. The above lemma draws from [43, Corollary 1] (see also [44, Corollary 2] for a simplified version), and is easily proved in the Appendix Section VII-A.

Creating large acyclic subgraphs: Lemma 1 suggests the need to create (preferably large) acyclic subgraphs of $\mathcal{G}_{\mathcal{U},\boldsymbol{d}}$. The following lemma describes how to properly choose a set of nodes to form a large acyclic subgraph.

Lemma 2. *Consider a subgraph $\mathcal{J}$ of $\mathcal{G}_{\mathcal{U},\boldsymbol{d}}$ corresponding to the index coding problem defined by $\mathcal{U}, \boldsymbol{d}, \chi$ for any $\mathcal{U}$ with profile $\boldsymbol{\mathcal{L}}$, consisting of all subfiles $W_{\mathcal{T}_\lambda}^{\boldsymbol{d}_{\boldsymbol{\sigma_s(\lambda)}}(j)}$, $\forall j \in [\mathcal{L}_\lambda]$, $\forall \lambda \in [\Lambda]$ for all $\mathcal{T}_\lambda \subseteq [\Lambda] \backslash \{\sigma_s(1), \dots, \sigma_s(\lambda)\}$ where $\sigma_s \in S_\Lambda$ is the permutation such that $|\mathcal{U}_{\sigma_s(1)}| \geq |\mathcal{U}_{\sigma_s(2)}| \geq \dots \geq |\mathcal{U}_{\sigma_s(\Lambda)}|$. This subgraph is acyclic.*

Proof. The proof, which can be found in the Appendix Section VII-B, is an adaptation of [2, Lemma 1] to the current setting.

Remark 3. *The choice of the permutation σ_s is critical for the development of a tight converse. Any other choice $\sigma \in S_\Lambda$ may result — in some crucial cases — in an acyclic subgraph with a smaller number of nodes and therefore a looser bound. This approach here deviates from the original approach in [2, Lemma 1], which instead considered — for each $\boldsymbol{d}, \chi$, for the uniform user-to-cache association case of $K = \Lambda$ — the set of* all *possible permutations, that jointly resulted in a certain symmetry that is crucial to that proof. Here in our case, such symmetry would not serve the same purpose as it would dilute the non-uniformity in $\boldsymbol{\mathcal{L}}$ that we are trying to capture. Our choice of a single carefully chosen permutation, allows for a bound which — as it turns out — is tight even in non-uniform cases. The reader is also referred to Section V-B for an explanatory example.*

Having chosen an acyclic subgraph according to Lemma 2, we return to Lemma 1 and form — by adding the sizes of all subfiles associated to the chosen acyclic graph — the following lower bound

$$T(\mathcal{U}, \boldsymbol{d}, \chi) \geq T^{LB}(\mathcal{U}, \boldsymbol{d}, \chi) \tag{50}$$

where

$$\begin{aligned} T^{LB}(\mathcal{U}, \boldsymbol{d}, \chi) \triangleq \frac{1}{N_0} \Bigg(&\sum_{j=1}^{\mathcal{L}_1} \sum_{\mathcal{T}_1 \subseteq [\Lambda] \backslash \{\sigma_s(1)\}} |W_{\mathcal{T}_1}^{\boldsymbol{d}_{\boldsymbol{\sigma_s(1)}}(j)}| \\ &+ \sum_{j=1}^{\mathcal{L}_2} \sum_{\mathcal{T}_2 \subseteq [\Lambda] \backslash \{\sigma_s(1), \sigma_s(2)\}} |W_{\mathcal{T}_2}^{\boldsymbol{d}_{\boldsymbol{\sigma_s(2)}}(j)}| + \dots \\ &+ \sum_{j=1}^{\mathcal{L}_\Lambda} \sum_{\mathcal{T}_\Lambda \subseteq [\Lambda] \backslash \{\sigma_s(1), \dots, \sigma_s(\Lambda)\}} |W_{\mathcal{T}_\Lambda}^{\boldsymbol{d}_{\boldsymbol{\sigma_s(\Lambda)}}(j)}| \Bigg). \end{aligned} \tag{51}$$

Our interest lies in a lower bound for the worst-case delivery time/delay associated to profile $\boldsymbol{\mathcal{L}}$. Such a worst-case naturally corresponds to the scenario where all users request different files, i.e., where all the entries of the demand vector $\boldsymbol{d}(\mathcal{U})$ are different. The corresponding lower bound can be developed by averaging over worst-case demands. Recalling our set $\mathcal{D}_{\boldsymbol{\mathcal{L}}}$, the worst-case delivery time can thus be written as

$$T^*(\boldsymbol{\mathcal{L}}) \triangleq \min_\chi \overbrace{\max_{(\mathcal{U},\boldsymbol{d}) \in (\mathcal{U}_{\boldsymbol{\mathcal{L}}}, [N]^K)} T(\mathcal{U}, \boldsymbol{d}, \chi)}^{T(\boldsymbol{\mathcal{L}}, \chi)} \tag{52}$$

$$\stackrel{(a)}{\geq} \min_\chi \frac{1}{|\mathcal{D}_{\boldsymbol{\mathcal{L}}}|} \sum_{\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}} T(\boldsymbol{d}(\mathcal{U}), \chi) \tag{53}$$

where in step (a), we used the following change of notation $T(\boldsymbol{d}(\mathcal{U}), \chi) \stackrel{\triangle}{=} T(\mathcal{U}, \boldsymbol{d}, \chi)$ and averaged over worst-case demands.

With a given class/profile $\boldsymbol{\mathcal{L}}$ in mind, in order to construct $\mathcal{D}_{\boldsymbol{\mathcal{L}}}$ (so that we can then average over it), we will consider all demand vectors $\boldsymbol{d} \in \mathcal{D}_{wc}$ for all permutations $\pi \in S_\Lambda$. Then for each $\boldsymbol{d}$, we create the following set of Λ vectors

$$\begin{aligned} \boldsymbol{d_1'} &= (d_1 : d_{\mathcal{L}_1}), \\ \boldsymbol{d_2'} &= (d_{\mathcal{L}_1+1} : d_{\mathcal{L}_1+\mathcal{L}_2}), \\ &\vdots \\ \boldsymbol{d_\Lambda'} &= (d_{\sum_{i=1}^{\Lambda-1} \mathcal{L}_i \, +1} : d_K) \end{aligned}$$

and for each permutation $\pi \in S_\Lambda$ applied to the set $\{1, 2, \dots, \Lambda\}$, a demand vector $\boldsymbol{d}(\mathcal{U})$ is constructed as follows

$$\boldsymbol{d}(\mathcal{U}) \stackrel{\triangle}{=} (\boldsymbol{d_1}, \boldsymbol{d_2}, \dots, \boldsymbol{d_\Lambda}) \tag{54}$$

$$= (\boldsymbol{d'_{\pi^{-1}(1)}}, \boldsymbol{d'_{\pi^{-1}(2)}}, \dots, \boldsymbol{d'_{\pi^{-1}(\Lambda)}}). \tag{55}$$

This procedure is repeated for all $\Lambda!$ permutations $\pi \in S_\Lambda$ and all $P(N,K)$ worst-case demands $\boldsymbol{d} \in \mathcal{D}_{wc}$. This implies that the cardinality of $\mathcal{D}_{\boldsymbol{\mathcal{L}}}$ is $|\mathcal{D}_{\boldsymbol{\mathcal{L}}}| = P(N,K) \cdot \Lambda!$.

Using this designed set $\mathcal{D}_{\boldsymbol{\mathcal{L}}}$, now the optimal worst-case delivery time in (53) is bounded as

$$T^*(\boldsymbol{\mathcal{L}}) = \min_{\chi} T(\boldsymbol{\mathcal{L}}, \chi) \tag{56}$$

$$\geq \min_{\chi} \frac{1}{P(N,K)\Lambda!} \sum_{\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}} T^{LB}(\boldsymbol{d}(\mathcal{U}), \chi) \tag{57}$$

where $T^{LB}(\boldsymbol{d}(\mathcal{U}), \chi)$ is given by (51) for each reordered demand vector $\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}$. Rewriting the summation in (57), we get

$$\sum_{\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}} T^{LB}(\boldsymbol{d}(\mathcal{U}), \chi) =$$

$$= \frac{1}{N_0} \sum_{i=0}^{\Lambda} \sum_{n \in [N]} \sum_{\mathcal{T} \subseteq [\Lambda]: |\mathcal{T}| = i} |W_{\mathcal{T}}^n| \cdot \underbrace{\sum_{\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}} \mathbb{1}_{\mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}(\mathcal{U})}}}(W_{\mathcal{T}}^n)}_{\triangleq Q_i(W_{\mathcal{T}}^n)}, \tag{58}$$

where $\mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}(\mathcal{U})}}$ is the set of vertices in the acyclic subgraph chosen according to Lemma 2 for a given $\boldsymbol{d}(\mathcal{U})$. In the above, $\mathbb{1}_{\mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}(\mathcal{U})}}}(W_{\mathcal{T}}^n)$ denotes the indicator function which takes the value of 1 only if $W_{\mathcal{T}}^n \subset \mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}(\mathcal{U})}}$, else it is set to zero.

A crucial step toward removing the dependence on $\mathcal{T}$, comes from the fact that

$$Q_i = Q_i(W_{\mathcal{T}}^n) \triangleq \sum_{\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}} \mathbb{1}_{\mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}(\mathcal{U})}}}(W_{\mathcal{T}}^n)$$

$$= \binom{N-1}{K-1} \sum_{r=1}^{\Lambda} P(\Lambda - i - 1, r-1)(\Lambda - r)! \mathcal{L}_r$$

$$\times P(K-1, \mathcal{L}_r - 1)(K - \mathcal{L}_r)!(\Lambda - i) \tag{59}$$

where we can see that the total number of times a specific subfile appears — in the summation in (58), over the set of all possible $\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}$, and given our chosen permutation σ_s — is not dependent on the subfile itself but is dependent only on the number of caches $i = |\mathcal{T}|$ storing that subfile. The proof of (59) can be found in Section VII-C.

In the spirit of [2], defining

$$x_i \triangleq \sum_{n \in [N]} \sum_{\mathcal{T} \subseteq [\Lambda]: |\mathcal{T}| = i} |W_{\mathcal{T}}^n| \tag{60}$$

to be the total amount of data stored in exactly i helper nodes, we see that

$$N = \sum_{i=0}^{\Lambda} x_i = \sum_{i=0}^{\Lambda} \sum_{n \in [N]} \sum_{\mathcal{T} \subseteq [\Lambda]: |\mathcal{T}| = i} |W_{\mathcal{T}}^n| \tag{61}$$

and we see that combining (57), (58) and (59), gives

$$T(\boldsymbol{\mathcal{L}}, \chi) \geq \frac{1}{N_0} \sum_{i=0}^{\Lambda} \frac{Q_i}{P(N,K)\Lambda!} x_i. \tag{62}$$

Now substituting (59) into (62), after some algebraic manipulations, we get that

$$T(\boldsymbol{\mathcal{L}}, \chi) \geq \frac{1}{N_0} \sum_{i=0}^{\Lambda} \frac{\sum_{r=1}^{\Lambda-i} \mathcal{L}_r \binom{\Lambda-r}{i}}{N \binom{\Lambda}{i}} x_i \tag{63}$$

$$= \frac{1}{N_0} \sum_{i=0}^{\Lambda} \frac{x_i}{N} c_i \tag{64}$$

where $c_i \triangleq \frac{\sum_{r=1}^{\Lambda-i} \mathcal{L}_r \binom{\Lambda-r}{i}}{\binom{\Lambda}{i}}$ decreases with $i \in \{0, 1, \ldots, \Lambda\}$. The proof of the transition from (62) to (63), as well as the monotonicity proof for the sequence $\{c_i\}_{i \in [\Lambda] \cup \{0\}}$, are given in Appendix Sections VII-D and VII-E respectively.

Under the file-size constraint given in (61), and given the following cache-size constraint

$$\sum_{i=0}^{\Lambda} i \cdot x_i \leq \Lambda M \tag{65}$$

the expression in (63) serves as a lower bound on the delay of any caching-and-delivery scheme χ whose caching policy implies a set of $\{x_i\}$.

We then employ the Jensen's-inequality based technique of [3, Proof of Lemma 2] to minimize the expression in (63), over all admissible $\{x_i\}$. Hence for any integer $\Lambda\gamma$, we have

$$T(\boldsymbol{\mathcal{L}}, \chi) \geq \frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \mathcal{L}_r \binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} \tag{66}$$

whereas for all other values of $\Lambda\gamma$, this is extended to its convex lower envelop. The detailed derivation of (66) can again be found in Appendix Section VII-F.

This concludes lower bounding $\max_{(\mathcal{U}, \boldsymbol{d}) \in (\mathcal{U}_{\boldsymbol{\mathcal{L}}}, [N]^K)} T(\mathcal{U}, \boldsymbol{d}, \chi)$, and thus — given that the right hand side of (66) is independent of χ — lower bounds the performance for any scheme χ, which hence concludes the proof of the converse for Theorem 2 (and consequently for Theorem 1 after setting $N_0 = 1$). □

A. Proof of the Converse for Corollary 2

For the uniform case of $\boldsymbol{\mathcal{L}} = [\frac{K}{\Lambda}, \frac{K}{\Lambda}, \ldots, \frac{K}{\Lambda}]$, the lower bound in (66) becomes

$$\frac{1}{N_0} \frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \mathcal{L}_r \binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} = \frac{1}{N_0} \frac{K}{\Lambda} \frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} \tag{67}$$

$$\stackrel{(a)}{=} \frac{1}{N_0} \frac{K}{\Lambda} \frac{\binom{\Lambda}{\Lambda\gamma+1}}{\binom{\Lambda}{\Lambda\gamma}} \tag{68}$$

$$= \frac{K}{N_0} \frac{(1-\gamma)}{(\Lambda\gamma + 1)} \tag{69}$$

where the equality in step (a) is due to Pascal's triangle which says that $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. □

B. Example for $N = K = 9$, $N_0 = 2$ and $\boldsymbol{\mathcal{L}} = (4, 3, 2)$

We here give an example of deriving the converse for Theorem 2, emphasizing on how to convert the caching problem to the index-coding problem, and how to choose acyclic subgraphs. We consider the case of having $K = 9$

receiving users, and a transmitter with $N_0 = 2$ transmit antennas having access to a library of $N = 9$ files of unit size. We also assume that there are $\Lambda = 3$ caching nodes, of average normalized cache capacity γ. We will focus on deriving the bound for user-to-cache association profile $\boldsymbol{\mathcal{L}} = (4, 3, 2)$, meaning that we are interested in the setting where one cache is associated to 4 users, one cache to 3 users and one cache associated to 2 users.

Each file W^n is split into $2^\Lambda = 8$ disjoint subfiles $W^i_{\mathcal{T}}, \mathcal{T} \in 2^{[3]}$ where each $\mathcal{T}$ describes the set of helper nodes in which $W^i_{\mathcal{T}}$ is cached. For instance, W^1_{13} refers to the part of file W^1 that is stored in the first and third caching nodes.

As a first step, we present the construction of the set $\mathcal{D}_{\boldsymbol{\mathcal{L}}}$. To this end, let us start by considering the demand $\boldsymbol{d} = (1, 2, 3, 4, 5, 6, 7, 8, 9)$ and one of the 6 permutations $\pi \in S_3$; for example, let us start by considering $\pi(1) = 2, \pi(2) = 3, \pi(3) = 1$. Toward reordering $\boldsymbol{d}$ to reflect $\boldsymbol{\mathcal{L}}$, we construct

$$\boldsymbol{d}'_{\boldsymbol{1}} = (1, 2, 3, 4), \quad \boldsymbol{d}'_{\boldsymbol{2}} = (5, 6, 7), \quad \boldsymbol{d}'_{\boldsymbol{3}} = (8, 9)$$

to obtain the reordered demand vector

$$\begin{aligned} \boldsymbol{d}(\mathcal{U}) &= (\boldsymbol{d}'_{\boldsymbol{\pi^{-1}(1)}}, \boldsymbol{d}'_{\boldsymbol{\pi^{-1}(2)}}, \boldsymbol{d}'_{\boldsymbol{\pi^{-1}(3)}}) \\ &= (\boldsymbol{d}'_{\boldsymbol{3}}, \boldsymbol{d}'_{\boldsymbol{1}}, \boldsymbol{d}'_{\boldsymbol{2}}) \end{aligned}$$

which in turn yields $\boldsymbol{d_1} = (8, 9), \boldsymbol{d_2} = (1, 2, 3, 4), \boldsymbol{d_3} = (5, 6, 7)$. Similarly, we can construct the remaining 5 demands $\boldsymbol{d}(\mathcal{U})$ associated to the other 5 permutations $\pi \in S_3$. Finally, the procedure is repeated for all other worst-case demand vectors. These vectors are part of set $\mathcal{D}_{\boldsymbol{\mathcal{L}}}$.

With the users demands $\boldsymbol{d}(\mathcal{U})$ known to the server, the delivery problem is translated into an index coding problem with a side information graph of $K2^{\Lambda-1} = 9 \cdot 2^2$ nodes. For each requested file $W^{\boldsymbol{d_\lambda}(j)}$, we write down the 4 subfiles that the requesting user does not have in its assigned cache. Hence, a given user of the caching problem requiring 4 subfiles from the main server, is replaced by 4 different new users in the index coding problem. Each of these users request a different subfile and are connected to the same cache λ as the original user. The nodes of the 6 side-information graphs corresponding to the aforementioned vectors $\boldsymbol{d}(\mathcal{U})$ (one for each permutation $\pi \in S_3$) for demand $\boldsymbol{d} = (1, 2, 3, 4, 5, 6, 7, 8, 9)$, are depicted in Figure 3.

For each side-information graph, we develop a lower bound as in Lemma 1. We recall that the lemma applies to acyclic subgraphs, which we create as follows; for each permutation[17] $\sigma \in S_3$, a set of nodes forming an acyclic subgraph is

$$\begin{aligned} &\{W^{\boldsymbol{d_{\sigma(1)}}(j)}_{\mathcal{T}_1}\}_{j=1}^{|\mathcal{U}_{\sigma(1)}|} \text{ for all } \mathcal{T}_1 \subseteq \{1,2,3\} \backslash \{\sigma(1)\}, \\ &\{W^{\boldsymbol{d_{\sigma(2)}}(j)}_{\mathcal{T}_2}\}_{j=1}^{|\mathcal{U}_{\sigma(2)}|} \text{ for all } \mathcal{T}_2 \subseteq \{1,2,3\} \backslash \{\sigma(1), \sigma(2)\}, \\ &\{W^{\boldsymbol{d_{\sigma(3)}}(j)}_{\mathcal{T}_3}\}_{j=1}^{|\mathcal{U}_{\sigma(3)}|} \text{ for all } \mathcal{T}_3 \subseteq \{1,2,3\} \backslash \{\sigma(1), \sigma(2), \sigma(3)\}. \end{aligned}$$

Based on this construction of acyclic graphs, our task now is to choose a permutation $\sigma_s \in S_3$ that forms the maximum-sized acyclic subgraph. For the case where $\boldsymbol{d_1} = (8, 9), \boldsymbol{d_2} = (1, 2, 3, 4)$ and $\boldsymbol{d_3} = (5, 6, 7)$, it can be easily verified that such a permutation σ_s is the one with $\sigma_s(1) = 2, \sigma_s(2) = 3$ and $\sigma_s(3) = 1$. In Figure 3, for each of the six graphs, we underline the nodes corresponding to the acyclic subgraph that is formed by such permutation σ_s. The outer bound now involves adding the sizes of these chosen (underlined) nodes. For example, for the demand $\boldsymbol{d}(\mathcal{U}) = ((8, 9), (1, 2, 3, 4), (5, 6, 7))$ (this corresponds to the lower center graph), the lower bound in (49) becomes

$$\begin{aligned} T(\boldsymbol{d}(\mathcal{U})) \geq &\frac{1}{2}(|W^1_\emptyset| + |W^1_1| + |W^1_3| + |W^1_{13}| + |W^2_\emptyset| \\ &+ |W^2_1| + |W^2_3| + |W^2_{13}| + |W^3_\emptyset| + |W^3_1| \\ &+ |W^3_3| + |W^3_{13}| + |W^4_\emptyset| + |W^4_1| + |W^4_3| \\ &+ |W^4_{13}| + |W^5_\emptyset| + |W^5_1| + |W^6_\emptyset| + |W^6_1| \\ &+ |W^7_\emptyset| + |W^7_1| + |W^8_\emptyset| + |W^9_0|). \end{aligned} \tag{70}$$

The lower bounds for the remaining 5 vectors $\boldsymbol{d}(\mathcal{U})$ for the same $\boldsymbol{d} = (1, 2, 3, 4, 5, 6, 7, 8, 9)$, are given in a similar way, again by adding the (underlined) nodes of the corresponding acyclic subgraphs (again see Figure 3).

Subsequently, the procedure is repeated for all $P(N, K) = K! = 9!$ worst-case demand vectors $\boldsymbol{d} \in \mathcal{D}_{wc}$. Finally, all the $P(N, K) \cdot \Lambda! = 9! \cdot 3!$ bounds are averaged to get

$$\begin{aligned} &T(\boldsymbol{\mathcal{L}}, \chi) \geq \\ &\frac{1}{2}\frac{1}{9! \cdot 3!} \sum_{\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}} \sum_{\lambda \in [3]} \sum_{j=1}^{\mathcal{L}_\lambda} \sum_{\mathcal{T}_\lambda \subseteq [3] \backslash \{\sigma_s(1), \ldots, \sigma_s(\lambda)\}} |W^{\boldsymbol{d_{\sigma_s(\lambda)}}(j)}_{\mathcal{T}_\lambda}| \end{aligned} \tag{71}$$

which is rewritten as

$$\begin{aligned} &T(\boldsymbol{\mathcal{L}}, \chi) \geq \\ &\frac{1}{2}\frac{1}{9! \cdot 3!} \sum_{i=0}^{3} \sum_{n \in [9]} \sum_{\mathcal{T} \subseteq [3] : |\mathcal{T}| = i} |W^n_{\mathcal{T}}| \cdot \underbrace{\sum_{\boldsymbol{d}(\mathcal{U}) \in \mathcal{D}_{\boldsymbol{\mathcal{L}}}} \mathbb{1}_{\mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}(\mathcal{U})}}}(W^n_{\mathcal{T}})}_{Q_i(W^n_{\mathcal{T}})} . \end{aligned} \tag{72}$$

After the evaluation of the term $Q_i(W^n_{\mathcal{T}})$, the bound in (72) can be written in a more compact form as

$$T(\boldsymbol{\mathcal{L}}, \chi) \geq \frac{1}{2} \sum_{i=0}^{3} \frac{\sum_{r=1}^{3-i} \mathcal{L}_r \binom{3-r}{i}}{9\binom{3}{i}} x_i \tag{73}$$

$$\geq Conv\left(\frac{1}{2} \frac{\sum_{r=1}^{3-i} \mathcal{L}_r \binom{3-r}{i}}{\binom{3}{i}}\right) \tag{74}$$

where the proof of the transition from (72) to (73) and from (73) to (74) can be found in the general proof (Section V).

VI. Conclusions

We have treated the multi-sender coded caching problem with shared caches which can be seen as an information-theoretically simplified representation of some instances of the so-called cache-aided heterogeneous networks, where one or more transmitters communicate to a set of users, with the assistance of smaller nodes that can serve as caches.

The work is among the first — after the work in [2] — to employ index coding as a means of providing (in this case, exact) outer bounds for more involved cache-aided

[17] We caution the reader not to confuse the current permutations (σ) that are used to construct large-sized acyclic graphs, with the aforementioned permutations π which are used to construct $\mathcal{D}_{\boldsymbol{\mathcal{L}}}$.

$\boldsymbol{d_1} = (1,2,3,4), \boldsymbol{d_2} = (5,6,7),$ $\boldsymbol{d_3} = (8,9)$

$$\begin{array}{cccc}
\underline{W_\emptyset^1} & \underline{W_2^1} & \underline{W_3^1} & \underline{W_{23}^1} \\
\underline{W_\emptyset^2} & \underline{W_2^2} & \underline{W_3^2} & \underline{W_{23}^2} \\
\underline{W_\emptyset^3} & \underline{W_2^3} & \underline{W_3^3} & \underline{W_{23}^3} \\
\underline{W_\emptyset^4} & \underline{W_2^4} & \underline{W_3^4} & \underline{W_{23}^4} \\
\underline{W_\emptyset^5} & W_1^5 & \underline{W_3^5} & W_{13}^5 \\
\underline{W_\emptyset^6} & W_1^6 & \underline{W_3^6} & W_{13}^6 \\
\underline{W_\emptyset^7} & W_1^7 & \underline{W_3^7} & W_{13}^7 \\
\underline{W_\emptyset^8} & W_1^8 & W_2^8 & W_{12}^8 \\
\underline{W_\emptyset^9} & W_1^9 & W_2^9 & W_{12}^9
\end{array}$$

$\boldsymbol{d_1} = (1,2,3,4), \boldsymbol{d_2} = (8,9),$ $\boldsymbol{d_3} = (5,6,7)$

$$\begin{array}{cccc}
\underline{W_\emptyset^1} & \underline{W_2^1} & \underline{W_3^1} & \underline{W_{23}^1} \\
\underline{W_\emptyset^2} & \underline{W_2^2} & \underline{W_3^2} & \underline{W_{23}^2} \\
\underline{W_\emptyset^3} & \underline{W_2^3} & \underline{W_3^3} & \underline{W_{23}^3} \\
\underline{W_\emptyset^4} & \underline{W_2^4} & \underline{W_3^4} & \underline{W_{23}^4} \\
\underline{W_\emptyset^5} & W_1^5 & \underline{W_2^5} & W_{12}^5 \\
\underline{W_\emptyset^6} & W_1^6 & \underline{W_2^6} & W_{12}^6 \\
\underline{W_\emptyset^7} & W_1^7 & \underline{W_2^7} & W_{12}^7 \\
\underline{W_\emptyset^8} & W_1^8 & W_3^8 & W_{13}^8 \\
\underline{W_\emptyset^9} & W_1^9 & W_3^9 & W_{13}^9
\end{array}$$

$\boldsymbol{d_1} = (5,6,7), \boldsymbol{d_2} = (1,2,3,4),$ $\boldsymbol{d_3} = (8,9)$

$$\begin{array}{cccc}
\underline{W_\emptyset^1} & \underline{W_1^1} & \underline{W_3^1} & \underline{W_{13}^1} \\
\underline{W_\emptyset^2} & \underline{W_1^2} & \underline{W_3^1} & \underline{W_{13}^2} \\
\underline{W_\emptyset^3} & \underline{W_1^3} & \underline{W_3^3} & \underline{W_{13}^3} \\
\underline{W_\emptyset^4} & \underline{W_1^4} & \underline{W_3^4} & \underline{W_{13}^4} \\
\underline{W_\emptyset^5} & W_2^5 & \underline{W_3^5} & W_{23}^5 \\
\underline{W_\emptyset^6} & W_2^6 & \underline{W_3^6} & W_{23}^6 \\
\underline{W_\emptyset^7} & W_2^7 & \underline{W_3^7} & W_{23}^7 \\
\underline{W_\emptyset^8} & W_1^8 & W_2^8 & W_{12}^8 \\
\underline{W_\emptyset^9} & W_1^9 & W_2^9 & W_{12}^9
\end{array}$$

$\boldsymbol{d_1} = (5,6,7), \boldsymbol{d_2} = (8,9),$ $\boldsymbol{d_3} = (1,2,3,4)$

$$\begin{array}{cccc}
\underline{W_\emptyset^1} & \underline{W_1^1} & \underline{W_2^1} & \underline{W_{12}^1} \\
\underline{W_\emptyset^2} & \underline{W_1^2} & \underline{W_2^2} & \underline{W_{12}^2} \\
\underline{W_\emptyset^3} & \underline{W_1^3} & \underline{W_2^3} & \underline{W_{12}^3} \\
\underline{W_\emptyset^4} & \underline{W_1^4} & \underline{W_2^4} & \underline{W_{12}^4} \\
\underline{W_\emptyset^5} & \underline{W_2^5} & W_3^5 & W_{23}^5 \\
\underline{W_\emptyset^6} & \underline{W_2^6} & W_3^6 & W_{23}^6 \\
\underline{W_\emptyset^7} & \underline{W_2^7} & W_3^7 & W_{23}^7 \\
\underline{W_\emptyset^8} & W_1^8 & W_3^8 & W_{13}^8 \\
\underline{W_\emptyset^9} & W_1^9 & W_3^9 & W_{13}^9
\end{array}$$

$\boldsymbol{d_1} = (8,9), \boldsymbol{d_2} = (1,2,3,4),$ $\boldsymbol{d_3} = (5,6,7)$

$$\begin{array}{cccc}
\underline{W_\emptyset^1} & \underline{W_1^1} & \underline{W_3^1} & \underline{W_{13}^1} \\
\underline{W_\emptyset^2} & \underline{W_1^2} & \underline{W_3^2} & \underline{W_{13}^2} \\
\underline{W_\emptyset^3} & \underline{W_1^3} & \underline{W_3^3} & \underline{W_{13}^3} \\
\underline{W_\emptyset^4} & \underline{W_1^4} & \underline{W_3^4} & \underline{W_{13}^4} \\
\underline{W_\emptyset^5} & \underline{W_1^5} & W_2^5 & W_{12}^5 \\
\underline{W_\emptyset^6} & \underline{W_1^6} & W_2^6 & W_{12}^6 \\
\underline{W_\emptyset^7} & \underline{W_1^7} & W_2^7 & W_{12}^7 \\
\underline{W_\emptyset^8} & W_2^8 & W_3^8 & W_{23}^8 \\
\underline{W_\emptyset^9} & W_2^9 & W_3^9 & W_{23}^9
\end{array}$$

$\boldsymbol{d_1} = (8,9), \boldsymbol{d_2} = (5,6,7),$ $\boldsymbol{d_3} = (1,2,3,4)$

$$\begin{array}{cccc}
\underline{W_\emptyset^1} & \underline{W_1^1} & \underline{W_2^1} & \underline{W_{12}^1} \\
\underline{W_\emptyset^2} & \underline{W_1^2} & \underline{W_2^2} & \underline{W_{12}^2} \\
\underline{W_\emptyset^3} & \underline{W_1^3} & \underline{W_2^3} & \underline{W_{12}^3} \\
\underline{W_\emptyset^4} & \underline{W_1^4} & \underline{W_2^4} & \underline{W_{12}^4} \\
\underline{W_\emptyset^5} & \underline{W_1^5} & W_3^5 & W_{13}^5 \\
\underline{W_\emptyset^6} & \underline{W_1^6} & W_3^6 & W_{13}^6 \\
\underline{W_\emptyset^7} & \underline{W_1^7} & W_3^7 & W_{13}^7 \\
\underline{W_\emptyset^8} & W_2^8 & W_3^8 & W_{23}^8 \\
\underline{W_\emptyset^9} & W_2^9 & W_3^9 & W_{23}^9
\end{array}$$

Fig. 3: Nodes of the side information graphs corresponding to demand vector $\boldsymbol{d} = (1,2,3,4,5,6,7,8,9)$ (profile $\boldsymbol{\mathcal{L}} = (4,3,2)$).

network topologies that better capture aspects of cache-aided wireless networks, such as having shared caches and a variety of user-to-cache association profiles. Dealing with such non uniform profiles, raises interesting challenges in redesigning converse bounds as well as redesigning coded caching which is known to generally thrive on uniformity. Our effort also applied to the related problem of coded caching with multiple file requests. In addition to crisply quantifying the (adverse) effects of user-to-cache association non-uniformity, the work also revealed a multiplicative relationship between multiplexing gain and cache redundancy, thus providing further evidence of the powerful impact of jointly introducing a modest number of antennas and a modest number of helper nodes that serve as caches. We believe that the result can also be useful in providing guiding principles on how to assign shared caches to different users, especially in the presence of multiple senders. Furthermore, as our results reveal, when considering multiple antennas or multiple senders, then having half the caches of twice the size, can have a much more powerful effect than having twice as many caches of half the size. As a testament to the importance of this shared caches setting, it is worth noting that after the preliminary conference version of this work appeared in [45], the shared caches problem for the single antenna setting studied in this paper, has been extended in a variety of ways. For example, in [46], the authors extended the single antenna setting of our work to the case when the delivery phase is error prone as well as to the case where some users can request the same files. Similarly, in [47], it was shown that our derived optimal performance can be improved if the server has full knowledge — during cache placement — of the number of users associated to each cache. This same knowledge was exploited in [48] which proposed a scheme that by optimizing the cache improves upon the scheme in [47], under the constraint of uncoded prefetching. Finally we believe that the current presented adaptation of the outer bound technique to non-uniform settings, may also be useful in analyzing different applications like distributed computing [49]–[52] or data shuffling [53]–[56] which can naturally entail such non uniformities.

VII. Appendix

A. *Proof of Lemma 1*

In the addressed problem, we consider a MISO broadcast channel with N_0 antennas at the transmitter serving K receivers with some side information due to caches. In the wired setting this (high-SNR setting) is equivalent to the distributed index coding problem with N_0 senders $J_1, \dots, J_{N_0}$, all having knowledge of the entire set of messages, and each being connected via an (independent) broadcast line link of capacity $C_{J_i} = 1, i \in [N_0]$ to the K receivers which hold side information. This multi-sender index coding problem is addressed in [43]. By adapting the achievable rate result in [43, Corollary1] to our problem, we get

$$\sum_{\mathcal{V}\in\mathcal{V}_{\mathcal{J}}} R_{\mathcal{V}} \leq \sum_{i\in[N_0]} C_{J_i} \tag{75}$$

($R_{\mathcal{V}} = \frac{|\mathcal{V}|}{T}$ is the rate for message $\mathcal{V}$), that yields

$$\sum_{\mathcal{V}\in\mathcal{V}_{\mathcal{J}}} \frac{|\mathcal{V}|}{T} \leq N_0 \tag{76}$$

which, when inverted, gives the bound in Lemma 1. □

B. Proof of Lemma 2

Consider a permutation σ where the subfiles $W_{\mathcal{T}_\lambda}^{\boldsymbol{d}_{\sigma(\lambda)}(j)}, \forall j \in \mathcal{U}_{\sigma(\lambda)}$ for all $\mathcal{T}_\lambda \subseteq [\Lambda]\setminus\{\sigma(1),\dots,\sigma(\lambda)\}$ are all placed in row λ of a matrix whose rows are labeled by $\lambda = 1, 2, \dots, \Lambda$. The index coding users corresponding to subfiles in row λ only know (as side information) subfiles $W_{\mathcal{T}}^{d_k}$, $\mathcal{T} \ni \sigma(\lambda)$. Consequently each user/node of row λ does not know any of the subfiles in the same row[18] nor in the previous rows. As a result, the proposed set of subfiles chosen according to permutation σ, forms a subgraph that does not contain any cycle.

A basic counting argument can tell us that the number of subfiles — in the acyclic subgraph formed by any permutation $\sigma \in S_\Lambda$ — that are stored in exactly i caches, is

$$\sum_{r=1}^{\Lambda-i} |\mathcal{U}_{\sigma(r)}| \binom{\Lambda - r}{i}. \tag{77}$$

This means that the total number of subfiles in the acyclic subgraph is simply

$$\sum_{i=0}^{\Lambda}\sum_{r=1}^{\Lambda-i} |\mathcal{U}_{\sigma(r)}| \binom{\Lambda - r}{i}. \tag{78}$$

This number is maximized when the permutation σ guarantees that the vector $(|\mathcal{U}_{\sigma(1)}|, |\mathcal{U}_{\sigma(2)}|, \dots, |\mathcal{U}_{\sigma(\Lambda)}|)$ is in descending order. This maximization is achieved with our choice of the ordering permutation σ_s (as this was defined in the notation part) when constructing the acyclic graphs. □

C. Proof of Equation (59)

Here, through a combinatorial argument, we derive $Q_i(W_{\mathcal{T}}^n)$, that is the number of times that a subfile $W_{\mathcal{T}}^n$ with index size $|\mathcal{T}| = i$ appears in all the acyclic subgraphs chosen to develop the lower bound.

There are $\binom{N-1}{K-1}$ subsets $\Upsilon_m, m \in [\binom{N-1}{K-1}]$ out of $\binom{N}{K}$ unordered subsets of K files from the set $\{W^j, j\in[N]\}$ that contain file W^n, and for each Υ_m there exists $K!$ different demand vectors $\boldsymbol{d}'$. For each Υ_m, among all possible demand vectors, a subfile $W_{\mathcal{T}}^n : |\mathcal{T}| = i$ appears in the side information graph an equal number of times. For a fixed Υ_m, file W^n is requested by a user connected to any helper node with a certain cardinality $\mathcal{L}_r$. By construction, $Q_i(W_{\mathcal{T}}^n)$ can be rewritten as

$$\begin{aligned}Q_i(W_{\mathcal{T}}^n) &= \sum_{\boldsymbol{d}\in\mathcal{D}_{wc}}\sum_{\pi\in S_\Lambda} \mathbb{1}_{\mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}_r(\mathcal{U})}}}(W_{\mathcal{T}}^n)\\ &= \binom{N-1}{K-1}\sum_{r=1}^{\Lambda}\sum_{\boldsymbol{d}'_r\in\mathcal{D}_{wc}}\sum_{\pi\in S_\Lambda} \mathbb{1}_{\mathcal{V}_{\mathcal{J}_s^{\boldsymbol{d}'_r(\mathcal{U})}}}(W_{\mathcal{T}}^n)\end{aligned} \tag{79}$$

where $\boldsymbol{d}'_r$ denotes the subset of all demand vectors from Υ_m such that $n \in \boldsymbol{d}_{\boldsymbol{\lambda}} : |\boldsymbol{d}_{\boldsymbol{\lambda}}| = \mathcal{L}_r$. The number of chosen maximum acyclic subgraphs containing $W_{\mathcal{T}}^n$ that arise from all the demand vectors $\boldsymbol{d}'_r(\mathcal{U})$ is evaluated as follows. After fixing the demands such that $n \in \boldsymbol{d}_{\boldsymbol{\lambda}} : |\boldsymbol{d}_{\boldsymbol{\lambda}}| = \mathcal{L}_r$, then $W_{\mathcal{T}}^n$ appears in the side information graph only if it is requested by a user connected to helper node λ such that $\lambda \notin \mathcal{T}$, which corresponds to $(\Lambda - i)$ different *available* positions in the demand vector $\boldsymbol{d'}_r(\mathcal{U})$, since $|\mathcal{T}| = i$. After fixing one of the $(\Lambda - i)$ positions occupied by $\boldsymbol{d}_{\boldsymbol{\lambda}} : |\boldsymbol{d}_{\boldsymbol{\lambda}}| = \mathcal{L}_r$, for the remaining demands $\boldsymbol{d}_{\boldsymbol{\lambda}} : |\boldsymbol{d}_{\boldsymbol{\lambda}}| = \mathcal{L}_j, \forall j \in [\Lambda]\setminus\{r\}$ there are $P(\Lambda - i - 1, r - 1)\cdot(\Lambda - r)!$ possible ways to be placed into $\boldsymbol{d}$. After fixing the order of $\boldsymbol{d}_{\boldsymbol{\lambda}}, \forall\lambda\in[\Lambda]$ in $\boldsymbol{d}$ and $n \in \boldsymbol{d}_{\boldsymbol{\lambda}} : |\boldsymbol{d}_{\boldsymbol{\lambda}}| = \mathcal{L}_r$, there are $\mathcal{L}_r$ different positions in which n can be placed in $\boldsymbol{d}_{\boldsymbol{\lambda}} : |\boldsymbol{d}_{\boldsymbol{\lambda}}| = \mathcal{L}_r$. This leaves out $\mathcal{L}_r - 1$ positions with $K - 1$ different numbers from the considered set $\Upsilon_m \setminus \{n\}$, and the remaining $K - \mathcal{L}_r$ positions in $\boldsymbol{d}'_r$ are filled with $K - \mathcal{L}_r$ numbers. Therefore, there exist $\mathcal{L}_r P(K-1, \mathcal{L}_r - 1)(K - \mathcal{L}_r)!$ different demand vectors where the subfile $W_{\mathcal{T}}^n$ will appear in the associated maximum acyclic subgraphs. Hence, the above jointly tell us that

$$\begin{aligned}Q_i(W_{\mathcal{T}}^n) &= \binom{N-1}{K-1}\sum_{r=1}^{\Lambda} P(\Lambda - i - 1, r - 1)\\ &\quad\times(\Lambda - r)!\mathcal{L}_r P(K-1, \mathcal{L}_r - 1)(K - \mathcal{L}_r)!(\Lambda - i)\end{aligned} \tag{80}$$

which concludes the proof. □

D. Transition from Equation (62) to (63)

The coefficient of x_i in equation (62), can be further simplified as follows

$$\begin{aligned}&\frac{Q_i}{\Lambda! P(N,K)} =\\ &= \frac{(N-1)!(N-K)!}{(K-1)!(N-K)!\Lambda! N!}\sum_{r=1}^{\Lambda}\mathcal{L}_r P(K-1,\mathcal{L}_r - 1)\\ &\quad\times(K-\mathcal{L}_r)!(\Lambda - i)P(\Lambda - i - 1, r - 1)(\Lambda - r)!\\ &= \frac{1}{(K-1)!\Lambda! N}\\ &\quad\times\sum_{r=1}^{\Lambda}\mathcal{L}_r\frac{(K-1)!(K-\mathcal{L}_r)!(\Lambda - i)(\Lambda - i - 1)!(\Lambda - r)!}{(K-\mathcal{L}_r)!(\Lambda - i - r)!}\\ &= \frac{1}{\Lambda! N}\sum_{r=1}^{\Lambda}\mathcal{L}_r\frac{(K-1)!(\Lambda - i)!(\Lambda - r)!}{(K-1)!(\Lambda - i - r)!}\\ &= \frac{1}{N}\sum_{r=1}^{\Lambda} L_{\pi_s(r)}\frac{(\Lambda - i)!(\Lambda - r)! i!}{\Lambda!(\Lambda - i - r)! i!}\\ &= \frac{1}{N}\sum_{r=1}^{\Lambda}\mathcal{L}_r\frac{\binom{\Lambda - r}{i}}{\binom{\Lambda}{i}}\end{aligned} \tag{81}$$

[18] Notice that the index coding users/nodes who are associated to the same cache, are not linked by any edge in the corresponding graph.

which concludes the proof. $\square$

E. Monotonicity of $\{c_i\}$

Let us define the following sequences

$$(a_n)_{n\in[\Lambda-i]} \triangleq \left\{\frac{\binom{\Lambda-n}{i}}{\binom{\Lambda}{i}}, n\in[\Lambda-i]\right\} \quad (82)$$

$$(b_n)_{n\in[\Lambda-i-1]} \triangleq \left\{\frac{\binom{\Lambda-n}{i+1}}{\binom{\Lambda}{i+1}}, n\in[\Lambda-i-1]\right\}. \quad (83)$$

It is easy to verify that $a_n \geq b_n,\ \forall n\in[\Lambda-i]$. Consider now the set of scalar numbers $\{V_j, j\in[\Lambda-i], V_j\in\mathbb{N}\}$. The inequality $a_n^*\geq b_n^*,\ \forall n\in[\Lambda-i]$ holds for

$$(a_n^*)_{n\in[\Lambda-i]} \triangleq \{V_n\cdot a_n, n\in[\Lambda-i]\} \quad (84)$$

$$(b_n^*)_{n\in[\Lambda-i-1]} \triangleq \{V_n\cdot b_n, n\in[\Lambda-i-1]\}. \quad (85)$$

As a result, we have

$$\sum_{n\in[\Lambda-i]} V_n\cdot a_n \geq \sum_{n\in[\Lambda-i]} V_n\cdot b_n \quad (86)$$

which proves that $c_i\geq c_{i+1}$. $\square$

F. Proof of (66)

Through the respective change of variables $t\triangleq\Lambda\frac{M}{N}$, $x_i'\triangleq\frac{x_i}{N}$ and $c_i'\triangleq\frac{c_i}{N_0}$, in equations (63), (61) and (65), we obtain

$$T(\boldsymbol{\mathcal{L}},\chi) \geq \sum_{i=0}^{\Lambda} x_i'c_i' \quad (87)$$

$$\sum_{i=0}^{\Lambda} x_i' = 1 \quad (88)$$

$$\sum_{i=0}^{\Lambda} ix_i' \leq t. \quad (89)$$

Let X denote a discrete integer-valued random variable with probability mass function $f_X(x)=\{x_i' \text{ if } x=i,\forall i\in\{0,1,\dots,\Lambda\}\}$, where the x_i' are those that satisfy equation (88). The value c_i' can also be seen as the realization of a random variable $Y\triangleq g(X)$, where $g(x)=\frac{\sum_{r=1}^{\Lambda-x}\mathcal{L}_r\binom{\Lambda-r}{x}}{N_0\binom{\Lambda}{x}}$, having the same probability mass function as X, i.e. $f_Y(y)=\{x_i' \text{ if } y=c_i',\forall i\in\{0,1,\dots,\Lambda\}\}$. Due to the equation in (89), the expectation of X is bounded as $\mathbb{E}[X]\leq t$. Similarly, (87) is equivalent to $T(\boldsymbol{\mathcal{L}},\chi)\geq\mathbb{E}[Y]$. From Jensen's inequality, we have $T(\boldsymbol{\mathcal{L}},\chi)\geq\mathbb{E}[Y]\geq g(\mathbb{E}[X])$. Next, we notice that function $g(x)$ is monotonically decreasing because it is obtained from the decreasing sequence $\{c_i'\}$ by replacing integer i with a continuous real variable x. This allows for the lower bound

$$T(\boldsymbol{\mathcal{L}},\chi)\geq g(\mathbb{E}[X])\geq g(t)\overset{(a)}{=}\frac{\sum_{r=1}^{\Lambda-t}\mathcal{L}_r\binom{\Lambda-r}{t}}{N_0\binom{\Lambda}{t}} \quad (90)$$

where (a) is due the decreasing monotonicity of $g(x)$ and the fact that $\mathbb{E}[X]\leq t$. This concludes the proof. $\square$

G. Proof of Equation (19)

We remind the reader that (for brevity of exposition, and without loss of generality) this part assumes that the $|\mathcal{U}_\lambda|$ are in decreasing order.

We define the following quantity

$$b_\lambda\triangleq|\mathcal{U}_1|-|\mathcal{U}_\lambda|$$

and rewrite the total number of transmissions using the above definition as

$$\begin{aligned}
&\sum_{j=1}^{|\mathcal{U}_1|}\binom{\Lambda}{\Lambda\gamma+1}-\binom{a_j}{\Lambda\gamma+1}\\
&=|\mathcal{U}_1|\binom{\Lambda}{\Lambda\gamma+1}-\sum_{j=1}^{|\mathcal{U}_1|}\binom{a_j}{\Lambda\gamma+1}\\
&=\sum_{i=1}^{\Lambda-\Lambda\gamma}(|\mathcal{U}_i|+b_i)\binom{\Lambda-i}{\Lambda\gamma}-\sum_{j=1}^{|\mathcal{U}_1|}\sum_{i=\Lambda\gamma}^{a_j-1}\binom{i}{\Lambda\gamma}\\
&=\sum_{i=1}^{\Lambda-\Lambda\gamma}|\mathcal{U}_i|\binom{\Lambda-i}{\Lambda\gamma}+\sum_{i=\Lambda\gamma}^{\Lambda-1}b_{\Lambda-i}\binom{i}{\Lambda\gamma}-\sum_{j=1}^{|\mathcal{U}_1|}\sum_{i=\Lambda\gamma}^{a_j-1}\binom{i}{\Lambda\gamma}\\
&\overset{(a)}{=}\sum_{i=1}^{\Lambda-\Lambda\gamma}|\mathcal{U}_i|\binom{\Lambda-i}{\Lambda\gamma}+\sum_{i=\Lambda\gamma}^{\Lambda-1}\sum_{j:a_j\geq i+1}^{|\mathcal{U}_1|}\binom{i}{\Lambda\gamma}\\
&\quad-\sum_{j:a_j-1\geq\Lambda\gamma}^{|\mathcal{U}_1|}\sum_{i=\Lambda\gamma}^{a_j-1}\binom{i}{\Lambda\gamma}\\
&\overset{(b)}{=}\sum_{i=1}^{\Lambda-\Lambda\gamma}|\mathcal{U}_i|\binom{\Lambda-i}{\Lambda\gamma}+\sum_{j:a_j\geq\Lambda\gamma+1}^{|\mathcal{U}_1|}\sum_{i=\Lambda\gamma}^{a_j-1}\binom{i}{\Lambda\gamma}\\
&\quad-\sum_{j:a_j-1\geq\Lambda\gamma}^{|\mathcal{U}_1|}\sum_{i=\Lambda\gamma}^{a_j-1}\binom{i}{\Lambda\gamma}\\
&=\sum_{i=1}^{\Lambda-\Lambda\gamma}|\mathcal{U}_i|\binom{\Lambda-i}{\Lambda\gamma}
\end{aligned} \quad (91)$$

where step (a) uses the equality $b_{\Lambda-i}=\sum_{j:a_j\geq i+1}^{|\mathcal{U}_1|}1$, and where step (b) is true as $\sum_{i=a}^{b}\sum_{j:a_j>i}^{c}$ is equivalent to $\sum_{j:a_j>a}^{c}\sum_{i=a}^{a_j}$. Substituting (91) into the numerator of (18) yields the overall delivery time given in (19).

The same performance holds for any $\mathcal{U}$ with the same profile $\boldsymbol{\mathcal{L}}$.

$\square$

H. Transition to the Multiple File Request Problem

We here briefly describe how the converse and the scheme presented in the shared cache problem, can fit the multiple file request problem.

Converse: In Remark 2 we described the equivalence between the two problems. Based on this equivalence, we will describe how the proof of the converse in Section V holds in the multiple file request problem with $N_0=1$, where now simply some terms carry a different meaning. Firstly, each entry $\boldsymbol{d_\lambda}$ of the vector defined in equation (48) now denotes the vector of file indices requested by user λ. Then we see that Lemma 2 (proved in Section VII-B) directly applies to the equivalent index coding problem of the multiple file requests problem, where now, for a given permutation σ (see Section VII-B), all the subfiles placed in row λ — i.e., subfiles $W_{\mathcal{T}_\lambda}^{\boldsymbol{d}_{\sigma(\lambda)}(j)}, \forall j \in \mathcal{U}_{\sigma(\lambda)}$ for all $\mathcal{T}_\lambda \subseteq [\Lambda] \setminus \{\sigma(1), \ldots, \sigma(\lambda)\}$ — are obtained from different files requested by the same user, and therefore any two of these subfiles/nodes are not connected by any edge in the side information graph. After these two considerations, the rest of the proof of Lemma 2 is exactly the same. The remaining of the converse consists only of mathematical manipulations which remain unchanged and which yield the same lower bound expression.

Scheme: The cache placement phase is identical to the one described in Section IV-A1, where now each cache λ is associated to the single user λ. In the delivery phase, the scheme now follows directly the steps in Section IV-A2 applied to the shared-link (single antenna) setting, where now $\boldsymbol{s}_{\boldsymbol{\lambda}} = \mathcal{U}_\lambda$ (cf. (10)). As in the case with shared caches, the scheme consists of $\mathcal{L}_1$ rounds, each serving users

$$\mathcal{R}_j = \bigcup_{\lambda \in [\Lambda]} \big(\mathcal{U}_\lambda(j) : \mathcal{L}_\lambda \geq j\big) \tag{92}$$

where $\mathcal{U}_\lambda(j)$ is the j-th user in set $\mathcal{U}_\lambda$. The expression in (92) now means that the multiple files requested by each user are transmitted in a *time-sharing* manner, and at each round the transmitter serves at most one file per user. Next, equation (12) is replaced by

$$\chi_{\mathcal{Q}} = \bigcup_{\lambda \in \mathcal{Q}} \big(\mathcal{U}_\lambda(j) : \mathcal{L}_\lambda \geq j\big) \tag{93}$$

and then each transmitted vector described in equation (13), is substituted by the scalar

$$x_{\chi_{\mathcal{Q}}} = \bigoplus_{\lambda \in \mathcal{Q} : \mathcal{L}_\lambda \geq j} W_{\mathcal{Q} \setminus \{\lambda\}, 1}^{d_{\mathcal{U}_\lambda(j)}}. \tag{94}$$

Finally decoding remains the same, and the calculation of delay follows directly.

References

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

[2] K. Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *IEEE Information Theory Workshop, (ITW)*, 2016, pp. 161–165.

[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] E. Parrinello, A. Ünsal, and P. Elia, "Fundamental limits of coded caching with shared caches and uncoded prefetching," 2018. [Online]. Available: https://arxiv.org/abs/1809.09422

[5] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "On the average performance of caching and coded multicasting with random demands," in *11th International Symposium on Wireless Communications Systems, (ISWCS)*, Aug 2014, pp. 922–926.

[6] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146–1158, Feb 2017.

[7] 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.

[8] S. S. Bidokhti, M. Wigger, and R. Timo, "Erasure broadcast networks with receiver caching," in *IEEE International Symposium on Information Theory, (ISIT)*, July 2016, pp. 1819–1823.

[9] J. Zhang and P. Elia, "Wireless coded caching: A topological perspective," in *IEEE International Symposium on Information Theory, (ISIT)*, June 2017, pp. 401–405.

[10] ——, "Fundamental limits of cache-aided wireless BC: Interplay of coded-caching and CSIT feedback," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 3142–3160, May 2017.

[11] A. Ghorbel, M. Kobayashi, and S. Yang, "Content delivery in erasure broadcast channels with cache and feedback," *IEEE Transactions on Information Theory*, vol. 62, no. 11, pp. 6407–6422, Nov 2016.

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

[13] V. Ravindrakumar, P. Panda, N. Karamchandani, and V. Prabhakaran, "Fundamental limits of secretive coded caching," in *IEEE International Symposium on Information Theory, (ISIT)*, 2016, pp. 425–429.

[14] A. Sengupta, R. Tandon, and O. Simeone, "Cache aided wireless networks: Tradeoffs between storage and latency," in *2016 Annual Conference on Information Science and Systems, (CISS)*, Mar 2016, pp. 320–325.

[15] Y. Cao, M. Tao, F. Xu, and K. Liu, "Fundamental storage-latency tradeoff in cache-aided MIMO interference networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 5061–5076, Aug 2017.

[16] J. S. P. Roig, D. Gündüz, and F. Tosato, "Interference networks with caches at both ends," in *IEEE International Conference on Communications, (ICC)*, 2017, pp. 1–6.

[17] Y. Cao and M. Tao, "Treating content delivery in multi-antenna coded caching as general message sets transmission: A DoF region perspective," 2018. [Online]. Available: https://arxiv.org/abs/1807.01432

[18] E. Piovano, H. Joudeh, and B. Clerckx, "Robust cache-aided interference management under full transmitter cooperation," in *IEEE International Symposium on Information Theory, (ISIT)*, June 2018, pp. 1540–1544.

[19] M. Bayat, R. K. Mungara, and G. Caire, "Achieving spatial scalability for coded caching over wireless networks," 2018. [Online]. Available: http://arxiv.org/abs/1803.05702

[20] E. Lampiris and P. Elia, "Achieving full multiplexing and unbounded caching gains with bounded feedback resources," in *IEEE International Symposium on Information Theory, (ISIT)*, June 2018, pp. 1440–1444.

[21] E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," in *IEEE International Symposium on Information Theory, (ISIT)*, June 2017, pp. 2960–2964.

[22] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite-length analysis of caching-aided coded multicasting," *IEEE Transactions on Information Theory*, vol. 62, no. 10, pp. 5524–5537, Oct 2016.

[23] M. Ji, K. Shanmugam, G. Vettigli, J. Llorca, A. M. Tulino, and G. Caire, "An efficient multiple-groupcast coded multicasting scheme for finite fractional caching," in *IEEE International Conference on Communications, (ICC)*, June 2015, pp. 3801–3806.

[24] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Transactions on Information Theory*, vol. 63, no. 9, pp. 5821–5833, Sept 2017.

[25] L. Tang and A. Ramamoorthy, "Low subpacketization schemes for coded caching," in *IEEE International Symposium on Information Theory, (ISIT)*, June 2017, pp. 2790–2794.

[26] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5755–5766, Aug 2018.

[27] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using Ruzsa-Szemeredi graphs," in *IEEE International Symposium on Information Theory, (ISIT)*, June 2017, pp. 1237–1241.

[28] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE Journal on Selected Areas in Communications (Special Issue on Caching)*, vol. 36, no. 6, pp. 1176–1188, June 2018.

[29] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *IEEE Conference on Computer Communications, (INFOCOM)*, March 2012, pp. 1107–1115.

[30] J. Hachem, N. Karamchandani, and S. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Transactions on Information Theory*, vol. 63, pp. 3108–3141, May 2017.

[31] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, Aug 2015.

[32] H. Xu, C. Gong, and X. Wang, "Efficient file delivery for coded prefetching in shared cache networks with multiple requests per user," 2018. [Online]. Available: http://arxiv.org/abs/1803.09408

[33] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Caching-aided coded multicasting with multiple random requests," in *IEEE Information Theory Workshop, (ITW)*, May 2015, pp. 1–5.

[34] A. Sengupta and R. Tandon, “Improved approximation of storage-rate tradeoff for caching with multiple demands,” *IEEE Transactions on Communications*, vol. 65, no. 5, pp. 1940–1955, May 2017.

[35] L. Zhang, Z. Wang, M. Xiao, G. Wu, and S. Li, “Centralized caching in two-layer networks: Algorithms and limits,” in *IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications, (WiMob)*, Oct 2016, pp. 1–5.

[36] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, “Hierarchical coded caching,” *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3212–3229, June 2016.

[37] Y. Wei and S. Ulukus, “Coded caching with multiple file requests,” in *55th Annual Allerton Conference on Communication, Control, and Computing*, Oct 2017, pp. 437–442.

[38] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, “Multi-server coded caching,” *IEEE Transactions on Information Theory*, vol. 62, pp. 7253–7271, Dec 2016.

[39] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “Characterizing the rate-memory tradeoff in cache networks within a factor of 2,” *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 647–663, Jan 2019.

[40] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, “Fundamental limits of cache-aided interference management,” *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 3092–3107, May 2017.

[41] S. Jin, Y. Cui, H. Liu, and G. Caire, “Order-optimal decentralized coded caching schemes with good performance in finite file size regime,” in *IEEE Global Communications Conference, (GLOBECOM)*, Dec 2016, pp. 1–7.

[42] S. Jin, Y. Cui, H. Liu, and G. Caire, “A new order-optimal decentralized coded caching scheme with good performance in the finite file size regime,” *IEEE Transactions on Communications*, 2019.

[43] M. Li, L. Ong, and S. J. Johnson, “Cooperative multi-sender index coding,” 2018. [Online]. Available: https://arxiv.org/abs/1701.03877v4

[44] P. Sadeghi, F. Arbabjolfaei, and Y. H. Kim, “Distributed index coding,” in *IEEE Information Theory Workshop, (ITW)*, Sept 2016, pp. 330–334.

[45] E. Parrinello, A. Unsal, and P. Elia, “Optimal coded caching in heterogeneous networks with uncoded prefetching,” in *2018 IEEE Information Theory Workshop (ITW)*, Nov 2018, pp. 1–5.

[46] N. S. Karat, S. Dey, A. Thomas, and B. S. Rajan, “An optimal linear error correcting delivery scheme for coded caching with shared caches,” in *2019 IEEE International Symposium on Information Theory (ISIT)*, July 2019, pp. 1217–1221.

[47] K. Wan, D. Tuninetti, M. Ji, and G. Caire, “A novel cache-aided fog-ran architecture,” in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 2977–2981.

[48] E. Parrinello and P. Elia, “Coded caching with optimized shared caches,” in *2019 IEEE Information Theory Workshop (ITW)*, Aug 2019.

[49] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, Jan 2018.

[50] E. Parrinello, E. Lampiris, and P. Elia, “Coded distributed computing with node cooperation substantially increases speedup factors,” in *IEEE International Symposium on Information Theory, (ISIT)*, June 2018, pp. 1291–1295.

[51] K. Konstantinidis and A. Ramamoorthy, “Leveraging coding techniques for speeding up distributed computing,” 2018. [Online]. Available: http://arxiv.org/abs/1802.03049

[52] Q. Yan, S. Yang, and M. A. Wigger, “A storage-computation-communication tradeoff for distributed computing,” 2018. [Online]. Available: http://arxiv.org/abs/1805.10462

[53] M. A. Attia and R. Tandon, “Information theoretic limits of data shuffling for distributed learning,” in *IEEE Global Communications Conference, (GLOBECOM)*, Dec 2016, pp. 1–6.

[54] ——, “Approximately optimal distributed data shuffling,” in *IEEE International Symposium on Information Theory, (ISIT)*, June 2018, pp. 721–725.

[55] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, “Fundamental limits of distributed data shuffling,” 2018. [Online]. Available: https://arxiv.org/abs/1807.00056

[56] A. Elmahdy and S. Mohajer, “On the fundamental limits of coded data shuffling,” in *IEEE International Symposium on Information Theory, (ISIT)*, June 2018, pp. 716–720.

Emanuele Parrinello received his B.Sc. degree in "Telecommunication Engineering" from Politecnico di Torino in 2015. He received the M.Sc. degree with honors in 2018 in "Communications and Computer Networks Engineering" and "Mobile Communications" in the context of a Double Degree program between Politecnico di Torino and EURECOM (Telecom ParisTech), respectively. He is currently pursuing his Ph.D with Sorbonne University, at the Communication systems department of EURECOM. His interests lie in caching networks, network information theory and wireless communication.

Ayşe Ünsal received the Bachelor of Science degree in Statistics from Hacettepe University, Ankara, Turkey in 2007. She obtained her Master of Science degree in the same subject from Dokuz Eylül University, Izmir, Turkey in 2010. In May 2011, she joined the Advanced Wireless Technology Group of the Mobile Communications Department of EURECOM to conduct her Ph.D. studies and obtained the Ph.D. degree in Electronics and Communication from Telecom ParisTECH, Paris, France in November 2014. She worked in the Signal and System Theory Group of Dept. of Electrical Engineering and Information Theory at the Paderborn University, Germany as a research and teaching fellow. After that she joined the Telecommunications Dept. of INSA Lyon and in 2017 she returned to the Communication Systems Dept. of EURECOM as a researcher. Her research interests lie in the area of Information Theory with an emphasis on detection and estimation.

Petros Elia received the B.Sc. degree from the Illinois Institute of Technology, Chicago, IL, USA, in 1997, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2001 and 2006, respectively. He is currently a Professor with the Department of Communication Systems, EURECOM. His latest research deals with information theoretic aspects of caching, as well with different problems in the areas of complexity-constrained communications, multi in multi out, coding theory, and surveillance networks. He is a Fulbright Scholar. He was a co-recipient of the NEWCOM++ Distinguished Achievement Award from 2008 to 2011, for a sequence of publications on the topic of complexity in wireless communications. He is a recipient of the European Research Council Consolidator Grant on cache-aided wireless communications from 2017 to 2022.