Sorbonne University

EURECOM

The high Dimensionality factor of Coded Caching:
Resolving Bottlenecks one Antenna at a time

Eleftherios LAMPIRIS

Dissertation submitted in partial satisfaction
of the requirements for the degree of
Doctor of Philosophy
in
Information and Communication Engineering

Thesis Supervisor: Petros ELIA

Defense date: 30 August 2019

before a committee composed of:

Prof. Giuseppe CAIRE	Technical Univ. of Berlin	Reviewer
Prof. Daniela TUNINETTI	Univ. of Illinois at Chicago	Reviewer
Prof. Mohammad Ali MADDAH-ALI	Sharif Univ. of Technology	Examiner
Prof. Antti TÖLLI	Univ. of Oulu	Examiner
Prof. Thrasyvoulos SPYROPOULOS	EURECOM	Examiner
Prof. Petros ELIA	EURECOM	Supervisor

Eurecom

The high Dimensionality Factor of Coded Caching: Resolving Bottlenecks one Antenna at a Time

Eleftherios LAMPIRIS

Sophia Antipolis, June 2019

Abstract

The main theme of this thesis is the combination of two opposing resources, namely the multiplexing gain, corresponding to multiple antennas, and the multicasting gain achieved in cache-aided communications. Multiple antennas, or multiple transmitters with caches, provide increased gains by separating messages, while coded caching merges messages together by exploiting cached and unwanted content to remove interference. Thus, at a first look, multiplexing and coded caching gains seem to be opposing to one another. Efforts to combine the two have shown that the gains appear to be additive. For example, in the Multiple-Input-Multiple-Output (MISO) Broadcast Channel (BC) where a Base Station, equipped with L transmit antennas, serves the demands of K receiving, single antenna users, asking files from a library of popular files, when users are equipped with caches, amounting to a total sum cache-size of t times the whole library, this network can provide the order optimal Degrees-of-Freedom (DoF) performance of $D_L = L + t$.

What we will show in this thesis is that, in many scenarios, pairing the multiplexing gains with coded caching can be a much more powerful combination, which can dramatically improve major fundamental limitations of both coded caching and multiple antenna precoding. A notable example, that will be proved in this thesis, is the role of multiple antennas on dramatically ameliorating the infamous subpacketization constraint i.e., the limitation that a coded caching gain equal to t requires each file to be segmented to an exponential number of packets. This, in practical systems can easily surpass the number of bits of a file by many orders of magnitude, thus imposing hard limits on caching gains. Specifically, we will show that in practical scenarios, where the number of subpackets needs to be finite, then L antennas can provide L times higher actual DoF compared to any single antenna Coded Caching scheme. As an example, in the single-antenna setting with $K = 50$ users and sum cache redundancy $t = 10$ in order to achieve the DoF performance of $D_1 = t + 1 = 11$ the required subpacketization is $S_1 = \binom{50}{10} \approx 10^{10}$ subfiles while on the other hand, we will show that the same system equipped with $L = 2$ antennas can achieve the DoF $D_L = t + L = 12$ with subpacketization of $S_L = 5 \cdot 10^4$. This DoF is approximately 2 times higher compared to the DoF that could be achieved under reasonable subpacketization constraints.

Another interesting result that we will present is the role of multiple antennas in resolving the bottleneck of having uneven cache sizes. For example, in the extreme case where cache-aided users coexist with cacheless users, we will first prove, via the use of Index Coding, that in the single

antenna case the two types of users cannot be served simultaneously showing that cache-size unevenness is a fundamental bottleneck of the single-antenna case. Nevertheless, we further show how a multiple antenna system can serve both user types at the same time, a result that either translates to the full DoF of $D_L = L + t$ for **all users involved**, i.e. the maximally known DoF of the system without the cache-less users or translates to a DoF improvement that is multiplicative to the number of antennas i.e., $L - 1$ added antennas can provide up to a $\times L$ DoF improvement compared to the optimal single antenna performance.

Some further contributions that will be presented are the reduction of the required Channel State Information at the transmitter (CSIT) and the receivers (CSIR) in the multiple-antenna coded caching, where we will show that the CSIT/CSIR costs can be untangled from the Coded Caching gains. Further, we will show that the proposed multi-antenna algorithm can provide huge CSIT/CSIR savings by reusing the already acquired feedback to transmit exponentially more information compared to the state-of-art. A further contribution is presented in the context of combining the reduced feedback costs with reduced subpacketization requirements, making progress towards combining subpacketization reductions with low feedback costs.

Moreover, we will explore how the DoF gains from partial connectivity can be combined with the Coded Caching gains, by studying two models, where in the first a transmitter's message can only be heard by a subset of the receivers, while in the second a receiver can hear all messages, some with full rate and some with lower rate.

Furthermore, we explore another bottleneck of wireless channels, where users are experiencing different channel capacities, which eventually leads to low-capacity users "slowing-down" the high-capacity ones. Specifically, we will study the fundamental limits of wireless Coded Caching with a single-antenna transmitter and users who experience different channel qualities. For this setting, we prove a performance bound and then continue to design an order optimal solution, with a gap to optimal of at most 4.

Finally, we will show how some of the above results can be applied to Coded MapReduce, a variant of the MapReduce framework which uses increased job assignment at the nodes during the mapping phase in order to decrease the shuffling (communication) phase's time. Specifically, we will show that known limitations such as the subpacketization constraint and heterogeneous nodes during the mapping phase can naturally be addressed by exploiting tools developed in multiple-antenna coded caching.

An important conclusion of this work is that for some of the presented settings we will show that, contrary to existing algorithms, adding antennas can provide an L-fold performance boost. This marks an unprecedented performance improvement that solidifies the importance of complementing Coded Caching with multiple antennas, and it is often attributed to the new design architectures that were invented here.

Thanks

This thesis has been the effort of approximately three and a half years and has flourished only because of the many people that have contributed directly, by being part of these works, or indirectly, through their support during this process.

My co-authors, Jingjing, Emanuele, Aly, Daniel, Berksan, Akis and Petros, have contributed to many of these works but were also always available to discuss ideas and to point out mistakes that I have made along the way (and some times I have been insisting on these stubbornly), and have helped me learn many things during these years.

I would like to thank my advisor, Petros, who believed in me enough to hire me and further stuck with me during the first couple of months of my PhD when I was going through a rough patch. His constant strive for excellence, his vision and his patience have helped me improve my skills and taught me to always look for the underlying meaning of things.

Furthermore, many people have indirectly helped me through this journey. First and foremost, my family and friends who were there to listen to me complaining when I was tired and when things were getting tough, but also my parents who have supported this decision of mine and have continued supporting me throughout this process.

Finally, I would like to thank Christos and Aris and Panagiotis who, not only, gave me directions in my early academic steps, during which I was completely at a loss, but they also instilled in me a passion and excitement for research that inspired me to pursue a PhD and an academic carreer.

Notation & System Model

Notation

Vectors are denoted by bold symbols.

Set notation

- Sets $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{C}$ denote the sets of naturals, integers, reals and complex numbers, respectively.
- $[N]$ denotes set $\{1, 2, ..., N\}$,
- For sets A, B we denote $A \backslash B$ the difference set i.e., $A \backslash B = \{x \in A \ \& \ x \notin B\}$.
- We note that, when relevant, sets are considered to be ordered, thus set ρ, $|\rho| = m$ can be written in the form $\rho = \{\rho(1), \rho(2), ..., \rho(m)\}$.

Operators

- $\oplus$ will denote the bit-wise XOR operation
- diag($\mathbf{x}$) transforms vector $\mathbf{x}$ to a diagonal square matrix with its entries comprized of the vector entries.
- $\binom{n}{k}$ denotes the n-choose-k operation for some $n, k \in \mathbb{N}$, $n \geq k$.
- $(x)^+ \triangleq \max\{x, 0\}$, $x \in \mathbb{R}$.
- For $a, b \in \mathbb{Z}$ then $a|b \Rightarrow 0 \equiv b \mod a$ and $a \nmid b \Rightarrow 0 \not\equiv b \mod a$

System Model

The main assumptions, that are common among the presented schemes, are the library size of N files, i.e. $\{W^n\}_{n=1}^N$, where each file is of size f bits. Further, the number of users is assumed to be K, and it is further assumed that users are equipped with caches able to store $M \cdot f$ bits, where $M < N$. We will be using the more convenient metric of the normalized cache size defined as $\gamma \triangleq \frac{M}{N}$ and we denote the cached content at user $k \in [K]$ by $\mathcal{Z}_k$.

User requests are denoted by d_k i.e., user $k \in [K]$ will be requesting file W^{d_k}. Our focus will be the calculation of the worst-case delivery time, corresponding to each user requesting a different file. As is common, when describing examples we will be denoting user requests by capital letters, as follows

$$A \triangleq W^{d_1}$$
$$B \triangleq W^{d_2}$$
$$\vdots$$

The two settings that we will mostly use are the single-antenna ($L = 1$) Broadcast Channel (BC) and the Multiple-Input-Single-Output (MISO) BC with L transmit antennas. For these settings, the metric of interest is the normalized, worst-case delivery time $T_L(K, \gamma)$.

Further, we will also use the cache-aided Degrees-of-Freedom (DoF) metric $D_L(K, \gamma)$, defined as

$$D_L(K, \gamma) = \frac{K(1-\gamma)}{T_L(K, \gamma)}. \tag{1}$$

In other words, the DoF[1] reveal how many users are treated in any single communication slot.

We will use quantity G to describe the theoretical caching gain as

$$G = D_L(K, \gamma) - D_L(K, \gamma = 0) \tag{2}$$

representing the number of extra users that could be served at a time, additionally, as a consequence of introducing caching.

MISO BC In many of the presented schemes we will be studying the DoF performance of the L-antenna MISO BC or its DoF-equivalent K_T cache-aided transmitter Interference Channel with K single antenna users. Each user simultaneously asks for a single and different file, from the library. In order to satisfy the users' demands, the base station transmits the $L \times 1$ vector $\mathbf{x}$. The signal at each receiver $k \in [K]$ takes the form

$$y_k = \mathbf{h}_k^T \mathbf{x} + w_k,$$

where $\mathbf{h}_k \in \mathbb{C}^{L \times 1}$ denotes the channel from the transmitter to receiver k, $\mathbf{x}$ is the $L \times 1$ vector message transmitted from the L antenna array, satisfying the power constraint $\mathbb{E}\{\|\mathbf{x}\|^2\} = P$ and $w_k \sim \mathbb{C}\mathcal{N}(0, 1)$ corresponds to the noise observed at user k. We assume that each node has all necessary channel-state information, and that for a given signal-to-noise ratio (SNR), each link has capacity of the form $\log(SNR) + o(log(SNR))$.

[1] In wireless communications with high signal-to-noise ratio (SNR), an achieved DoF D implies the ability to deliver approximately $D \log(SNR)$ bits of content, per second per Hz. Equivalently, it implies the ability to simultaneously serve D independent users (i.e., D users per complex dimension, i.e., per second per Hz).

Further, we will denote with $\mathcal{H}_\lambda^{-1}$ the $L \times L$ matrix formed as the normalized inverse of the channel between the L-antenna transmitter and the L users in set λ where, in the case of the Interference Channel setting this matrix is formed in a distributed manner.

In particular, matrix $\mathcal{H}_\lambda^{-1}$ can be written in the form

$$\mathcal{H}_\lambda^{-1} \triangleq [\mathbf{h}^{\perp}_{\lambda\setminus\{1\}}, ...\mathbf{h}^{\perp}_{\lambda\setminus\{L\}}] \tag{3}$$

where

$$\mathbf{h}_k^T \mathbf{h}^{\perp}_{\lambda\setminus\{i\}} = \begin{cases} 0, & k \in \lambda \setminus \{i\} \\ \neq 0, & k \in [K] \setminus \lambda \cup \{i\}. \end{cases} \tag{4}$$

Feedback and Precoding We assume that feedback is perfect and instantaneous, while the feedback training process is divided into 2 phases; the uplink phase where the transmitter estimates the channels of some users, and the downlink phase where receiver k estimates products $\mathbf{h}_k^T \mathbf{h}_\lambda^{\perp}$, for some set $\lambda \subset [K], |\lambda| = L$. The feedback training process follows that of [2], hence feedback for C users will require C training slots in the uplink training phase (for CSIT) and C training slots in the downlink training phase (for local and global CSIR). This process is described in detail in Chapter 4.

For simplicity, we will assume that the selected precoder is a Zero-Forcing (ZF) precoder, which is adequate for a DoF analysis.

Contents

Chapter 1

Introduction

The role of caching has been brought to the forefront of academic research due to its potential to reduce the load of networks. The main idea is to capitalize on the fact that a significant part of the traffic corresponds to cacheable content, thus by bringing content – mostly, popular content – and storing it closer to – or at – the users, can allow savings either in the backhaul or the fronthaul.

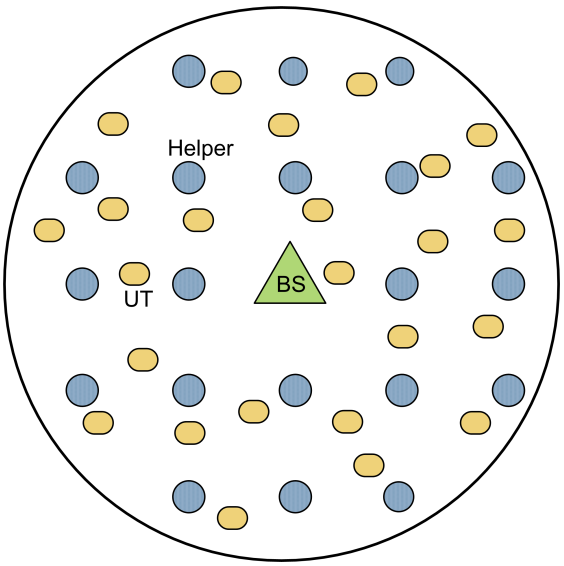

Figure 1.1: The femtocaching model of [3]. Helper nodes are equipped with caches so as to offload the backhaul. Users (UT) are requesting files from a library of popular files (source [3]).

The work in [3] proposed the use of smaller nodes (Helper nodes) to serve the demands of users (user terminals) (cf. Figure 1.1). Helper nodes have a limited radius in which they are able to satisfy the demands of some of the users and at the same time they have the ability to store content, which can help reduce the backhaul load. The helper nodes are connected to the base station and can request a file if this requested file is not found in their cache.

The main idea is to cache files at the helper nodes in such a manner that can minimize the demands passed on the base station. The above work has

sparked significant interest and many variants (see for example [4–7]).

1.1 Coded Caching - A paradigm shift

A very different approach in caching was introduced in [1] by Maddah-Ali and Niesen. Compared to the previously discussed caching efforts, where the focus was concentrated into predicting the content popularity and exploiting this knowledge by storing the most popular files first, in the Coded Caching approach each cached content is used to reduce the received interference.

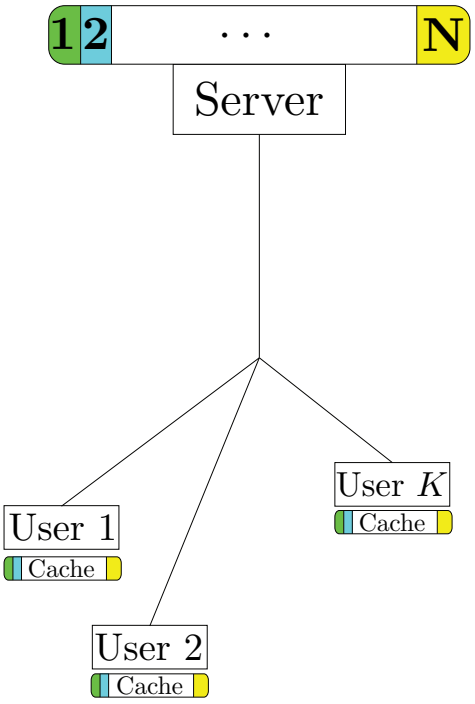

Figure 1.2: The bottleneck channel, with K users and N files.

Specifically, the model in the work of Maddah-Ali and Niesen [1] considered a server with access to a library of N files, which server was connected via a wired, noise-less channel with link capacity 1 file to K users, each equipped with a cache able to store the equivalent capacity of M files (cf. Figure 1.2). The system works in two distinct phases, where during the first (placement phase) content is pushed to the users in a manner oblivious to future requests. Then, during the second phase (delivery phase) each user requests one file from the server and the server transmits a sequence of messages in order to satisfy these demands.

1.1.1 Performance of Coded Caching

For the above described setting, the algorithm of [1] showed that it can achieve the normalized performance (delivery time) of

$$T_1(K,\gamma) = \frac{K(1-\gamma)}{1+K\gamma} \tag{1.1}$$

which also translates to a cache-aided Degrees-of-Freedom (DoF) performance of

$$D_1(K,\gamma) \triangleq \frac{K(1-\gamma)}{T_1(K,\gamma)} = 1+K\gamma \tag{1.2}$$

implying an ability to serve $1+K\gamma$ users at a time (all with a different requested content), over the same link that would – in the absence of Coded Caching – have allowed the serving of only one user at a time.

Toy Example

Before formally describing the placement and delivery algorithms in [1], we will first illustrate these algorithms through the use of a toy example. Let us assume the single transmitter, wired bottleneck channel with $K=2$ users (see Figure 1.3), where the server has access to a library of $N=2$ files, namely $\{W^1, W^2\}$. Further, let us assume that each user has the capacity to store content of size equal to the size of $M=1$ file.

Placement Phase The algorithm starts by splitting each of the two files into $S=2$ parts (subfiles) as follows

$$W^1 \rightarrow \left\{W_1^1, W_2^1\right\} \tag{1.3}$$
$$W^2 \rightarrow \left\{W_1^2, W_2^2\right\} \tag{1.4}$$

and caching at the two users as

$$\mathcal{Z}_1 = \left\{W_1^1, W_1^2\right\} \tag{1.5}$$
$$\mathcal{Z}_1 = \left\{W_2^1, W_2^2\right\} \tag{1.6}$$

where $\mathcal{Z}_k$ denotes the contents of user k's cache.

Delivery Phase Starting with denoting the two requested files using letters A, B i.e., A will represent the preference of user 1 and B the preference of user 2, the algorithm transmits the message

$$x_{12} = A_2 \oplus B_1 \tag{1.7}$$

where $\oplus$ is the bit-wise XOR operator. We can see that this naming abstraction can represent any possible combination of the files W^1, W^2, thus for

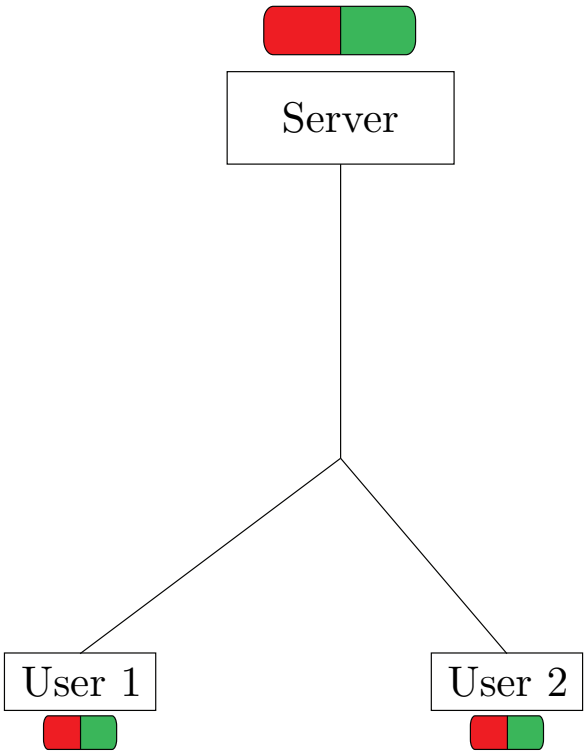

Figure 1.3: The bottleneck channel, with $K = 2$ users and $N = 2$ files.

any possible request pattern the above message will satisfy users' demands, as we will show in the following paragraph. Moreover, we can see that the above transmitted message contains both remaining desired subfiles, where the users can decode their desired subfile by "XORing" out the interfering message.

Specifically, user 1 will use subfile B_1, found in its cache, while user 2 will use subfile A_2 to recover their respective subfiles.

As evident from Eq. (1.7), only one transmission is needed to satisfy both users' demands. On the other hand, having not performed the Coded Caching algorithm, then the subfiles would need to be transmitted in separate transmissions and the reduction that could be achieved would amount to the size of the already cached parts, i.e. the local caching gain. Taking these comments into consideration we can see that the required time to complete all user requests is reduced in half.

Note Throughout this document we use the convention, unless otherwise stated, that each user selects a different file, thus the calculated delivery time amounts to the worst case delivery time. As a result, it follows that the number of files needs to always be higher than the number of users. For the scenario where the number of files is less than the number of users as well as the scenario where more than one users select the same file the reader is referred to the works in [8, 9].

The Coded Caching algorithm

We proceed with a detailed description of the coded caching algorithm of [1]. It is important to notice that some elements presented in this section will form the basis for the algorithms presented in the following next chapters.

As discussed above, the server has access to a library of N files, $\{W^n\}_{n=1}^N$, where each file is of size f bits. The server is connected to a set of K users, and each user can store in its cache $M \cdot f$ bits, corresponding to a cache of normalized size $\gamma \triangleq \frac{M}{N}$. The process starts with the placement phase.

Placement Phase During the placement phase the caches of the users are populated with content, where this phase takes place before any demand has been made.

Initially, each file is subpacketized into

$$S_1 = \binom{K}{K\gamma} \tag{1.8}$$

subfiles as

$$W^n \to \{W^n_\tau,\ \tau \subset [K],\ |\tau| = K\gamma\} \tag{1.9}$$

where index τ tells us that the subfile W^n_τ is stored at the $|\tau| = K\gamma$ caches indicated by set index τ.

Then, the cache of user $k \in [K]$, i.e. $\mathcal{Z}_k$, is populated with content as follows

$$\mathcal{Z}_k = \{W^n_\tau : k \in \tau, \quad \forall n \in [N]\} \tag{1.10}$$

where we can see that this selection of content respects the cache-size constraint since

$$\frac{|\mathcal{Z}_k|}{S_1} = \frac{\binom{K-1}{K\gamma-1}}{\binom{K}{K\gamma}} = \gamma. \tag{1.11}$$

From the placement algorithm (cf. Eq. (1.10)) we can make the following observations:

- First, as demonstrated, each user is storing a fraction of each file equal to $\gamma \in (0, 1)$, which migrates from traditional caching policies that either entirely store a file in a user's cache or would not store it and
- further, any file from the library can be found in its entirety in the collective cache of the users, which also constitutes a fundamentally different approach compared with the traditional caching algorithms where unpopular files may not be stored at the users,

- thus, we can deduce that the difference between the traditional caching algorithms and the Coded Caching algorithm is that in the first case users are treated individually and content is stored with the intention to optimize the delivery time regardless of the other users. On the other hand, the coded caching approach is designed in such a way to use the collective cache of the users.
- Moreover, each subfile of every file is cached by a subset of $K\gamma$ users and
- finally, the file index τ helps identify which $K\gamma$ users have stored this subfile.

Delivery Phase The delivery phase commences with the request of one file by each of the K users, where we denote the demand of user $k \in [K]$ by W^{d_k}.

for *all* $\sigma \subseteq [K], |\sigma| = K\gamma + 1$ *(Select* $K\gamma + 1$ *users)* **do**
  Transmit:

$$x_\sigma = \bigoplus_{k \in \sigma} W^{d_k}_{\sigma \setminus \{k\}} \tag{1.12}$$

end

Algorithm 1.1: Delivery Phase of the MN algorithm [1]

The delivery process is described in Algorithm 1.1, where we can see that in each iteration a new subset of $K\gamma + 1$ users is selected. For this user subset, a XOR is formed in such a way so as to contain, for any of the users in σ, one unknown and desired subfile and $K\gamma$ subfiles that can be found in that user's cache.

Decoding From Eq. (1.12) we have that if user k belongs in set σ of the selected users then the received message x_σ will contain a subfile that is desired by this user i.e., $W^{d_k}_{\sigma \setminus \{k\}}$. In order for user k to decode this subfile it is required to remove the interfering subfiles. It is easy to see that in each XOR the $K\gamma$ unwanted subfiles (from user k's perspective) are all cached, thus this user can decode its desired message.

1.2 Extension to multiple-transmitters

An extension of the work of [1] aimed to combine the gains from caching with the traditional multiplexing gains of feedback-aided multi-antenna systems. This was an interesting direction that sought to merge two seemingly opposing approaches, where traditional feedback-based multi-antenna systems work by creating parallel channels that separate users' signals, while coded caching fuses users' signals and counts on each user receiving maximum

interference. In this context, the work in [10] showed that — in a wired

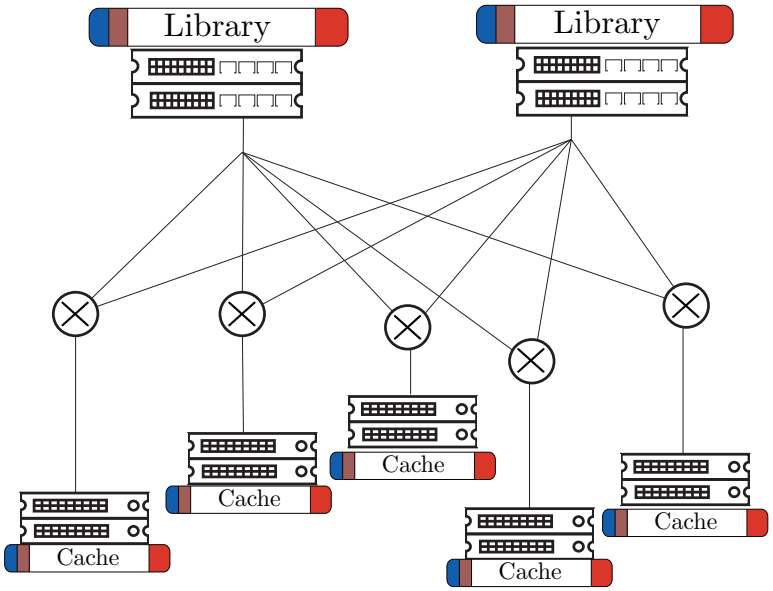

Figure 1.4: The multi-server setting of [10], where $L = 2$ servers, having access to the library of N files are serving the demands of K users. The transmitted messages are linearly combined (symbol $\times$ denotes a linear operation between a set of messages), while the receivers are equipped with caches, partially storing files from the library.

multi-server (L servers) setting (see Figure 1.4), which can easily be seen to correspond to the high-SNR cache-aided MISO BC setting with L transmit antennas (see Figure 1.5) — the two gains (multiplexing and caching gains) could be combined additively, yielding the normalized delay of

$$T_L(K, \gamma) = \frac{K(1-\gamma)}{L + K\gamma} \tag{1.13}$$

which corresponds to the DoF performance of

$$D_L(K, \gamma) = L + K\gamma. \tag{1.14}$$

Soon after, the work in [11] explored the Interference Channel (IC) comprized of K_T cache-aided, single-antenna transmitters with individual caches of normalized size γ_T, and K cache-aided, single-antenna receivers with individual caches of normalized size γ. In this setting, the achieved Degrees-of-Freedom performance is

$$D_{K_T\gamma_T}(K, \gamma) = K_T\gamma_T + K\gamma \tag{1.15}$$

and which performance was proved in [11] to be optimal up to a multiplicative factor of 2 under the assumptions of linear and one-shot schemes. We can see that the two systems, namely the K_T-transmitter IC with transmitter caches of normalized size γ_T, and the L-antenna MISO BC ($L = K_T\gamma_T$) are DoF-equivalent.

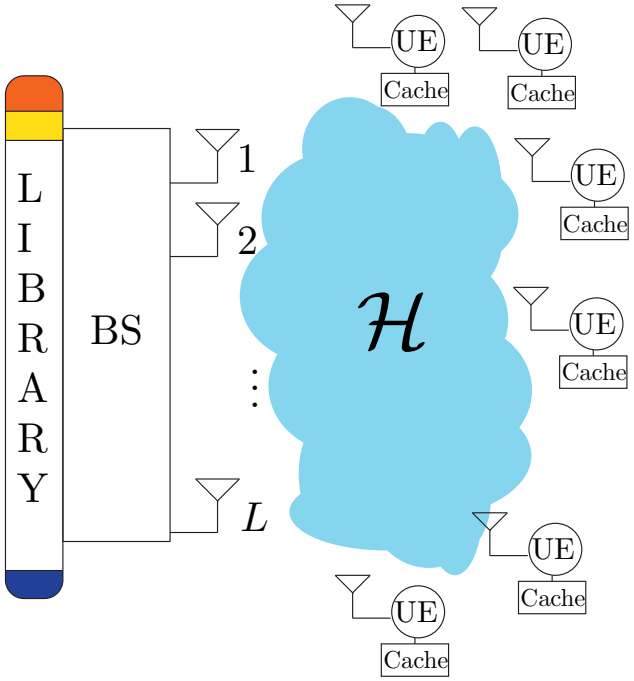

Figure 1.5: The base station serves the demands of K single-antenna wireless nodes and is equipped with $L < K$ antennas. Users request one out of N files and have caches of equivalent capacity equal to the size of M files. It is assumed that channel matrix $\mathcal{H}$ is perfectly known to both the base station and the users.

1.2.1 The Multi-server algorithm

The main idea governing the schemes of [10, 11] is based on the fact that a transmitted message can be "cached-out" by $K\gamma$ users and at the same time, using spatial multiplexing, can be "nulled-out" at $L-1$ users, thus allowing any transmitted message to simultaneously serve $L + K\gamma$ users.

We proceed with the description the multi-server (MS) algorithm of [10], since the two algorithms (cf. [10, 11]) share many commonalities.

Placement phase

Initially, each file W^n is subpacketized into

$$S_{L,MS} = \binom{K - K\gamma - 1}{L-1} \cdot \binom{K}{K\gamma} \tag{1.16}$$

subfiles, where a subfile is denoted by two indices i.e.,

$$W^n \to \left\{ W^n_{\phi,\tau},\ \phi \in \left[\binom{K - K\gamma - 1}{L-1} \right],\ \tau \subset [K],\ |\tau| = K\gamma \right\}. \tag{1.17}$$

User caches are filled similarly to the algorithm of [1] i.e.,

$$\mathcal{Z}_{k\in[K]} = \left\{ W^n_{\phi,\tau} :\ k \in \tau,\ \forall \phi,\ \forall n \in [N] \right\}. \tag{1.18}$$

Delivery phase

The delivery algorithm of the multi-server scheme, as discussed above, relies on the idea of a message being cache-able by $K\gamma$ users while at the same time it can be hidden from $L-1$ users. To capitalize on this, the algorithm creates XORs, in the same way as the original Coded Caching algorithm, and uses the multiple antennas to "steer-away" a XOR from the $L-1$ unintended recipients. Thus, this XOR only "appears" to those users that are requesting one of the subfiles contained in the XOR and which, at the same time, can remove (cache-out) all other elements. The delivery procedure is described in the form of pseudo-code in Algorithm 1.2.

for *all* $\rho \subseteq [K], |\rho| = K\gamma + L$ *(Select* $K\gamma + L$ *active users)* **do**
  Initialize transmit vector:

$$\mathbf{x}_\rho = \mathbf{0} \tag{1.19}$$

  for *all* $\sigma \subset \rho, |\sigma| = K\gamma + 1$ *(Select* $K\gamma + 1$ *subset)* **do**

$$\mathbf{x}_\rho = \mathbf{x}_\rho + \mathbf{h}^{\perp}_{\rho\setminus\sigma} \bigoplus_{k\in\sigma} W^{d_k}_{\mathrm{NEW}(\phi),\sigma\setminus\{k\}} \tag{1.20}$$

  end
  Transmit $\mathbf{x}_\rho$.
end

Algorithm 1.2: Delivery Phase of the multi-server algorithm [10].

By analysing the algorithm of [10], we can see that

1. Initially, set ρ comprized of $K\gamma+L$ users is selected out of the K users, and
2. the transmit vector $\mathbf{x}_\rho$ is initialized.
3. Then, one-by-one all subsets $\sigma \subset \rho$ comprized of $K\gamma + 1$ users are selected, and for such selected subset σ
   - A precoding vector is created $\mathbf{h}^{\perp}_{\rho\setminus\sigma}$, such that it is orthogonal to the channels of the users in set $\rho \setminus \sigma$,
   - XOR $\oplus_{k\in\sigma} W^{d_k}_{\mathrm{NEW}(\phi),\sigma\setminus\{k\}}$ is formed, that aims to convey information towards the users in set σ and
   - the transmit vector $\mathbf{x}_\rho$ is updated by adding the product of the precoder vector with the XOR.

We can see that each XOR created in the above algorithm (cf. Algorithm 1.2) is the same as those generated by the algorithm of [1], with the only difference that of index ϕ, which is used here to ensure that in every transmission a new subfile (mini-file) is transmitted.

Decoding process of the Multi-server algorithm

From Algorithm 1.2 we can see that a received message at some user $k \in \rho$ takes the form

$$y_\rho(k) = \mathbf{h}_k^T \sum_{\sigma \subset \rho} \mathbf{h}_{\rho\setminus\sigma}^{\perp} \bigoplus_{k\in\sigma} W_{\phi,\sigma\setminus\{k\}}^{d_k} \tag{1.21}$$

where noise has been ignored for convenience. The first observation that we need to make is that all XORs that don't include user k will not appear at the received message, since the precoders are designed in such a way that

$$\mathbf{h}_k^T \cdot \mathbf{h}_{\rho\setminus\sigma}^{\perp} = 0, \quad \text{if } k \in \rho \setminus \sigma. \tag{1.22}$$

Thus, the message received contains a linear combination with XORs that contain a subfile that user k needs. In order for this user to decode these subfiles, this transmission needs to be repeated $\binom{K\gamma+L}{K\gamma+1}$ times under different channel realizations, such that user k can have a set of $\binom{K\gamma+L}{K\gamma+1}$ different linear combinations of messages and proceed to decode them.

1.3 Extensions of Coded Caching

The introduction of Coded Caching in [1], has sparked significant interest and has subsequently been applied to many interesting scenarios.

In the Device-to-Device (D2D) setting of [12] (see also [13]), nodes are directly connected to each other via a wireless or a wired link. Nodes possess caches storing part of the library, from which they will eventually request one file, and need to satisfy each other's request.

The D2D model provides the basis for the analysis of other interesting settings involving distributed computing, such as those in Coded MapReduce [14–16] and distributed data shuffling [17, 18].

Further, interesting connections have been made between Coded Caching and settings, such as combination networks [19–22], fog networks [23–29] and hierarchical networks [30–32].

1.3.1 Discussion

It is, thus, evident that the introduction of Coded Caching can significantly increase the performance of networks, where in many scenarios equipping each user with a cache of normalized size γ would yield an increase of the DoF performance by an additive gain of $G = K\gamma$.

In the next chapter we will identify some fundamental bottlenecks of coded caching that severely reduce these promised gains, in a variety of ways. We will finish this next chapter with a description of our solutions to these bottlenecks, as well as with a description of some other related results in this thesis.

Chapter 2

Fundamental Limitations of Coded Caching

2.1 Theoretical Performance

So far we have reviewed how the main idea behind Coded Caching is the use of receiver-side caches to massively reduce the required time to deliver cache-able content. This is achieved not by trying to predict what each individual user will ask, but rather to use the cached file segments at a user that have been requested by other users, so as to reduce the amount of received interference.

As we have seen in the previous section, this allows to transmit to multiple users simultaneously even when using a single antenna, which results in a delivery time (for the original Coded Caching setting of [1]) of

$$T_{L=1}(K,\gamma) = \frac{K(1-\gamma)}{1+K\gamma} \to \frac{1}{\gamma}. \tag{2.1}$$

By analyzing Eq. (2.1) we can see that even if the number of users is increasing, the achievable delivery time of the scheme remains approximately constant i.e., for some integer m the delay difference between two systems, one with $m \cdot K$ users and another with K users becomes

$$\lim_{K\to\infty} \big(T_1(m\cdot K,\gamma) - T_1(K,\gamma)\big) = 0. \tag{2.2}$$

The above observation sheds light on a very important reality of coded caching, namely that the system performance is less dependent on the number of users and more dependent on the size of the cache that each user can hold.

Subsequently, the results of [10, 11] that studied, respectively, the multi-antenna and multi-transmitter settings, have showed that the system delay can further decrease by increasing the number of transmit antennas, to yield the delay of

$$T_L(K,\gamma) = \frac{K(1-\gamma)}{L+K\gamma} \to \frac{1}{\frac{L}{K}+\gamma}. \tag{2.3}$$

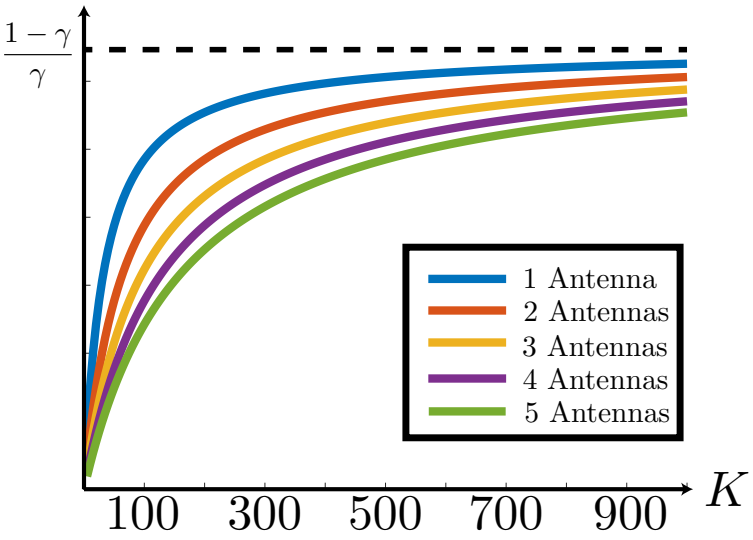

Figure 2.1: The delivery time as a function of the number of users K for various values of transmit antennas, as is theoretically achieved by the Coded Caching scheme of [1] and the multi-antenna schemes [10, 11] for a fixed normalized cache per user of fraction γ. We can see that the effect of introducing antennas to the system can only marginally improve the delay, and as users keep growing then this delay approaches the lower bound delay of $T_b = \frac{1-\gamma}{\gamma}$.

As we can see in Figure 2.1, the single antenna delay approaches the value $\frac{1-\gamma}{\gamma}$ as we increase the number of users, while the benefit of adding extra antennas can be seen to reduce the delay but in a rather modest manner in settings involving many users.

2.1.1 Optimality of Coded Caching

In the work of [1] it was proven that the achievable delay of the coded caching scheme is optimal up to a multiplicative gap of 12.

Subsequently, many works have sought to tighten this gap. Here we make an attempt to list some of these.

- Work [33] reduced the gap to a multiplicative factor of 8.
- Work [34] improved the bound to a multiplicative factor of 4.
- Work [35] showed that, under the assumption of uncoded placement, the achievable performance of Eq. (2.1) is *exactly* optimal.
- Subsequently work [9] extended the result of [35] to account for the case where $N < K$.
- The work in [36] reduced the gap for the worst-case demand to 2.35.

- Finally, work [37] showed that for any possible placement of files, i.e. not restricting to uncoded placement, the performance of Eq. (2.1) is optimal up to at most a multiplicative factor of 2.

2.2 Fundamental limitations and practical considerations of coded caching

As we have seen, the impact of Coded Caching can be substantial, especially as the number of users increases, where most modern communication systems fail to scale. Furthermore, we can observe that the above presented results are close to optimal, and under some assumptions optimal.

In this section, we will explore the performance of coded caching in the presence of some realistic fundamental limitations that appear once one takes a closer look. Specifically, we will focus on four main aspects related to Coded Caching that constrain the performance of the above algorithms in realistic settings namely, i) the subpacketization constraint, ii) the feedback bottleneck, iii) the worst-user effect and finally iv) the uneven cache sizes.

2.2.1 Subpacketization bottleneck of coded caching

We have seen that the gain $G \triangleq D_L(K,\gamma) - D_L(K,\gamma=0) = K\gamma$ corresponding to coded caching would, in theory, increase indefinitely with an increasing number of users K. Nevertheless, in practice this gain remained — under most realistic assumptions — hard-bounded by small constants, due to the fact that the underlying coded caching algorithms required the splitting of finite-length files into an exponential number of subpackets. For the algorithm in [1] in the original single-stream scenario, the DoF $D_1(K,\gamma) = 1 + K\gamma$ is achieved only if each file is segmented at least into

$$S_1 = \binom{K}{K\gamma} \tag{2.4}$$

subpackets (see [38]). As a result, having a certain maximum-allowable subpacketization of $S_{\max}$, one solution[1] that can reduce the required subpacketization to become less than $S_{\max}$ is to encode over a maximum of

$$\bar{K} = \arg\max_{K^o \le K} \left\{ \binom{K^o}{K^o\gamma} \le S_{\max} \right\} \tag{2.5}$$

users, which in turn implies a substantially reduced *effective caching gain* $\bar{G}_1$ of the form

$$\bar{G}_1 = \bar{K}\gamma. \tag{2.6}$$

[1]We note here that much more sophisticated algorithms have been proposed in the literature, which can reduce the required, single-stream subpacketization, while achieving a higher effective caching gain than that of Eq. (2.8). We discuss those in Chapter 3.

Given such a 'user-grouping' reduction of having to encode over groups of only $\bar{K}$ users at a time, and given that

$$\binom{\bar{K}}{\bar{K}\gamma} \in \left[\left(\frac{1}{\gamma}\right)^{\bar{K}\gamma}, \left(\frac{e}{\gamma}\right)^{\bar{K}\gamma}\right] = \left[\left(\frac{1}{\gamma}\right)^{\bar{G}_1}, \left(\frac{e}{\gamma}\right)^{\bar{G}_1}\right] \tag{2.7}$$

this effective gain $\bar{G}_1$ is bounded as

$$\frac{\log S_{\max}}{1+\log\frac{1}{\gamma}} \leq \bar{G}_1 \leq \frac{\log S_{\max}}{\log\frac{1}{\gamma}}, \qquad \bar{G}_1 \leq G \tag{2.8}$$

(log is the natural logarithm) which succinctly reveals that the effective caching gain $\bar{G}_1$ (and the corresponding *effective DoF* $\bar{D}_1 \triangleq 1+\bar{G}$) is placed under constant pressure from the generally small values of γ and $S_{\max}$. This is reflected in Figure 2.2 and Figure 2.3. Interestingly, as we know from [39], under some basic assumptions, in the context of single-antenna decentralized coded caching, this "user-grouping" approach is close to optimal.

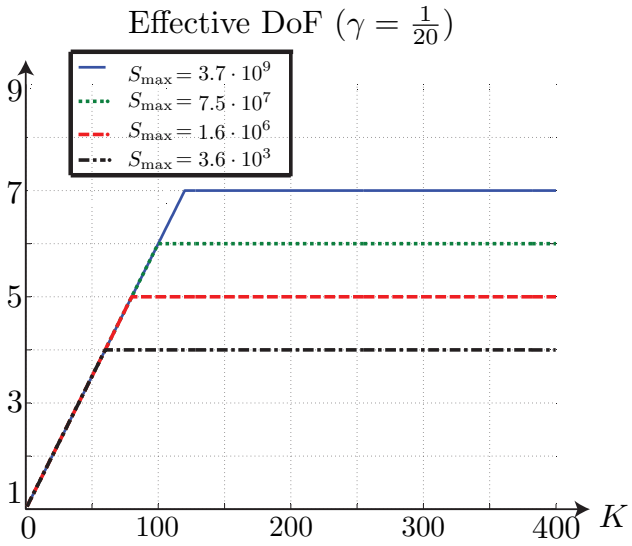

Figure 2.2: Maximum effective DoF $\bar{D}_1$ achieved by the original centralized algorithm (single antenna, $\gamma = \frac{1}{20}$) in the presence of different subpacketization constraints $S_{\max}$. The gain is hard-bounded irrespective of the number of users K (x-axis).

Remark 2.1. *It is worth noting here that, as argued in [40], in wireless cellular settings, the storage capacity at the end users is expected to induce caches of normalized size γ that can be less than 10^{-2}, which — for a given target caching gain — implies the need to code over many users, which in turn increases subpacketization. Compounding on this problem, there is a variety of factors that restrict the maximum allowable subpacketization level $S_{\max}$. One such parameter is the file size; for example, movies are expected to have size that is close to or less than 1 Gigabyte. Additionally, in applications like video*

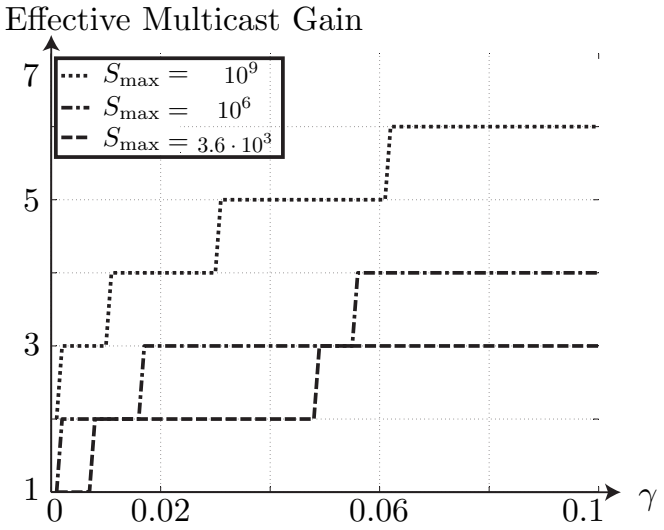

Figure 2.3: Effective caching gain $\bar{G}_1 = \bar{D}_1 - 1$ (maximized over K) of the original algorithm for different $S_{\max}$. Without subpacketization constraints, the theoretical gain is $G = K\gamma$ (unbounded as K increases).

streaming, a video file it self may be broken down into smaller independent parts (on which subpacketization will take place separately), in order to avoid the delay that comes from the asynchronous nature of decoding XORs in coded caching. Such restricted file sizes may be in the order of just a few tens of Megabytes. Another parameter that restricts $S_{\max}$ is the minimum packet size; the atomic unit of storage is not a bit but a sector (newer 'Advanced Format' hard drives use 4096-byte sectors and force zero-padding on the remaining unused sector), and similarly the atomic communication block is the packet, which must maintain a certain minimum size in order to avoid communication delay overheads.

Example 2.1. *Looking at Figure 2.3, we see that if the library files (e.g. movies) are each of size 1 Gigabyte, and under a constraint that each packet cannot be less than 1 Kilobyte (KB) long (which jointly imply a subpacketization limit of $S_{\max} \approx 10^6$), then having $\gamma < 1/20$ would hard-bound the effective caching gain $\bar{G}_1$ to be less than 4 (we add one extra in comparison to the plot, in order to account for any possible improvements from memory-sharing between operating points that yield neighboring integer-valued gains). This gain reduction is because we are forced to encode over less than $\bar{K} = 80$ users, to avoid a subpacketization $\binom{80}{4} > 10^6$ that exceeds $S_{\max}$. Having $\gamma < 1/100$ would limit this gain $\bar{G}_1$ to be less than 3 (since $\bar{K} = 300$ implies subpacketization $\binom{300}{3} > 10^6$). When $S_{\max} = 10^9$, where each packet consists of a single byte (without taking into consideration the overhead from using byte-sized packets), then having $\gamma < 1/20$ would limit the effective gain to less than 6, while having $\gamma < 1/100$ would limit the number $\bar{G}_1$ of additional users that could be served due to caching, to less than 4. When $S_{\max} \approx 36K$, reflecting perhaps low-latency video streaming applications, for $\gamma \leq 1/20$ then $\bar{G}_1 \approx 3$ ($\bar{D}_1 \approx 4$ users at a time), while for $\gamma \leq 1/100$ then $\bar{G}_1 \approx 2$ ($\bar{D}_1 \approx 3$).*

2.2.2 Scaling Feedback Costs in Multi-Antenna CC

Starting from the single antenna case [1] where one could achieve the caching gain $K\gamma$ without requiring any channel state information (CSI) at the transmitter (CSIT), a significant feedback problem arises in the presence of multiple antennas. Specifically, all known multi-antenna coded caching methods [10, 11, 41–43] that achieve the full DoF $L + K\gamma$, incur scaling feedback costs as they require each of the $L + K\gamma$ benefiting receivers to send feedback to the transmitter. A similar *global feedback* cost appears with respect to the required CSI at the receivers (CSIR) which was required to include information on the CSIT-based precoders of all the $L + K\gamma$ benefiting users.

The following example aims to demonstrate the aforementioned CSI costs, and it focuses on a simple instance of the original multiserver method in [10] which serves as a proxy to other methods with similar feedback requirements.

Example 2.2. *Let us consider the cache-aided MISO BC setting with $K = 4$ users, normalized cache-size $\gamma = 1/2$ and $L = 2$ transmit antennas where, using the multiserver approach, one can treat $L + K\gamma = 4$ users at a time.*

Each of the three transmissions takes the form

$$\begin{aligned}\mathbf{x} =& \mathbf{h}_4^{\perp}(A_{23} \oplus B_{13} \oplus C_{12}) + \mathbf{h}_3^{\perp}(A_{24} \oplus B_{14} \oplus D_{12}) + \\ &+ \mathbf{h}_2^{\perp}(A_{34} \oplus C_{14} \oplus D_{13}) + \mathbf{h}_1^{\perp}(B_{34} \oplus C_{24} \oplus D_{23})\end{aligned} \tag{2.9}$$

We clearly see that the transmitter must know all users' channel vectors (in order to form the four precoders), and at the same time — in order to be able to decode the desired subfile — each receiver must know the composite channel-precoder product for each precoder (e.g. receiver 1 must know $\mathbf{h}_1^T\mathbf{h}_1^{\perp}$ as well as $\mathbf{h}_1^T\mathbf{h}_2^{\perp}$, $\mathbf{h}_1^T\mathbf{h}_3^{\perp}$ and $\mathbf{h}_1^T\mathbf{h}_4^{\perp}$). This implies $L + K\gamma = 4$ uplink training slots for CSIT acquisition, and $L + K\gamma = 4$ downlink training slots for global CSIR acquisition[2]*.*

In the context of frequency division duplexing (FDD), this feedback cost of existing methods, implies a CSIT cost of $L+K\gamma$ feedback vectors, while in the more interesting setting of Time Division Duplexing (TDD), this requires $L + K\gamma$ uplink training time slots for CSIT acquisition, and an extra cost of $L + K\gamma$ downlink training time slots for global CSIR acquisition, thus the required CSI increases as the number of users increases.

As we know, such scaling feedback costs can consume a significant portion of the coherence time, thus resulting in diminishing DoF gains, as this was shown in [44].

2.2.3 Channel Unevenness (Worst-User Effect)

The third bottleneck of Coded Caching is related to a reality of the wireless channel, where each user is experiencing a different channel capacity. This

[2]The process of feedback acquisition will be described in Algorithm 4.1 found in Section 4.1.1, where we will recall that global CSIR acquisition can be performed by broadcasting a set of training symbols to all users.

bottleneck – also referred to as "the worst-user bottleneck"– is exacerbated by the multicast nature of Coded Caching, since if a user experiences a "bad" channel then the rate of each XOR that includes this user needs to be low enough to allow decoding at this user.

Thus, the uneven channels bottleneck not only affects the performance of a single user, but instead impacts the overall system performance.

Example 2.3. *Let us consider the wireless Single-Input-Single-Output (SISO) BC with K users, each equipped with a cache of normalized size γ and let us further assume that the channel strength[3] of the first user is equal to $\alpha_1 = \frac{1}{K}+\gamma$, while the remaining $K-1$ users have the maximum channel strengths i.e., $\alpha_2 = ... = \alpha_K = \alpha = 1$.*

Assuming a naive implementation of the algorithm of [1] – where the designed XORs (which are designed for the equal-strength case) are sent sequentially one after the other – the achieved delay, T_1^{uc}, under uneven channel strengths would take the form

$$T_1^{uc} = (1-\gamma)\frac{1}{\alpha_1} + \frac{(K-1)(1-\gamma)}{1+K\gamma}\frac{1}{\alpha} \tag{2.12}$$

$$= \frac{1-\gamma}{\frac{1+K\gamma}{K}} + \frac{(K-1)(1-\gamma)}{1+K\gamma} \approx 2T_1. \tag{2.13}$$

Thus, a naive implementation of the algorithm of [1] when even a single user is experiencing a bad channel can almost double the experienced delivery time, independently from the cache-size and which has the equivalent effect of reducing by half the multicasting gain.

2.2.4 Cache size unevenness

The final bottleneck we will discuss relates to uneven cache sizes. To accentuate this bottleneck let us, first, consider the extreme case where cache-less users coexist with cache-aided users. In this setting, we will see that, when considering the single-stream setting (either wireless or wired) where both user types co-exist we can treat only one type per transmission. This fact

[3]Here we will make use of the Generalized Degrees-of-Freedom (GDoF) framework used in [45] (see also [46]), thus a message received at some user $k \in [K]$ takes the form

$$y_k = \sqrt{P^{\alpha_k}} h_k x + w_k, \tag{2.10}$$

with P representing the transmitting power, $x \in \mathbb{C}$ the output signal of the transmitter satisfying the power constraint $\mathbb{E}\{|x|^2\} \leq 1$ and $h_k \in \mathbb{C}$ corresponding to the channel coefficient of user k. Further, $\alpha_k \in (0,1]$ represents the normalized, by factor $\log P$, link strength of user k and finally, $w_k \sim \mathbb{CN}(0,1)$ represents the Gaussian noise at user k. From the above, the average signal-to-noise-ratio (SNR) at user $k \in [K]$ takes the form

$$\mathbb{E}\{|\sqrt{P^{\alpha_k}} h_k x|^2\} = P^{\alpha_k} \tag{2.11}$$

which amounts to a (normalized) user rate of $r_k = \alpha_k$. The GDoF model is presented in detail in Chapter 6.

can deeply impact the performance of the system, since the addition of a cache-less user to the system incurs a total delay increase of one unit of time.

This bottleneck is closely related to the multicasting nature of coded caching, where a transmitted message is designed to be useful to many users. As mentioned before, the decoding process requires the use of cached content to remove the interfering subfiles so that a user can retrieve its file, which has the effect of a cache-less user not being able to take advantage of these messages, which means that the two types of users will need to be treated separately.

As suggested, the above bottleneck translates to an increase of a whole unit of time for each cache-less user, where we will show in section 5.4.1 that in the single-stream case this performance is exactly optimal under the assumption of uncoded placement i.e., the delay of a system with K_1 cache-aided users (each equipped with a cache of normalized size γ_1) and K_2 cache-less users takes the form

$$T_1(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1} + K_2. \tag{2.14}$$

The scenario of cache-less users co-existing with cache-aided ones is of particular interest for various reasons. First, it is expected that during the transition from a system that does not exploit cached content to a fully cache-aided system, there will inevitably be a period when both user types coexist. Further, even in the fully cache-aided scenario there may well be some users that opt-out from dedicating part of their cache to store content. Finally, some users may possess legacy devices that may not be able to take advantage of coded caching.

2.3 Preview of the Results

In this chapter we have discussed some performance bottlenecks that can severely deteriorate the performance of Coded Caching. In the following chapters we will show how impactful multiple antennas can be in ameliorating some of these bottlenecks, such as the subpacketization constraint (see Chapter 3) and the uneven cache-size bottleneck (see Chapter 5).

We proceed with a preview of the contributions of the remaining chapters.

Chapter 3 - Subpacketization The first bottleneck that we will explore is the subpacketization constraint, which can severely deteriorate the Coded Caching gains in practical systems.

What we will show here for the first time is that pairing transmitters with Coded Caching not only does not increase the required subpacketization, but can severely reduce it. Our contribution lies in the realization that having this extra dimensionality on the transmitter side, in fact reduces rather than increases subpacketization, and does so in a very accelerated manner. This

property is based on the principle of the *virtual decomposition* of the cache-aided MISO BC into L parallel, single-stream coded caching channels with $\frac{K}{L}$ users each. This decomposition is made possible because, as we show here, the near optimal DoF $D_L(K,\gamma) = L\big(1+\frac{K}{L}\gamma\big) = L + K\gamma$ can be gained **without encoding across parallel channels**.

We continue with the two main results.

Theorem. *In the cache-aided MISO BC with L transmitting antennas and K receiving users, the delay of $T_L(K,\gamma) = \frac{K(1-\gamma)}{L+K\gamma}$ and the corresponding DoF $D_L(K,\gamma) = L + K\gamma$, can be achieved with subpacketization*

$$S_L = \binom{\frac{K}{L}}{\frac{K\gamma}{L}}. \tag{2.15}$$

Corollary. *Under a maximum allowable subpacketization $S_{\max}$, the multi-antenna effective caching gain and DoF take the form*

$$\bar{G}_L = \min\{L\cdot\bar{G}_1, K\gamma\} \tag{2.16}$$

$$\bar{D}_L(K,\gamma) = \min\{L\cdot\bar{D}_1(K,\gamma), L+K\gamma\} \tag{2.17}$$

which means that with extra antennas, the (single-antenna) effective DoF $\bar{D}_1(K,\gamma)$ is either increased by a multiplicative factor of L, or it reaches the theoretical (unconstrained) DoF $D_L(K,\gamma) = L + K\gamma$.

Chapter 4 - CSI Bottleneck The second bottleneck that we will explore is the scaling (with the number of users) feedback requirements of multi-antenna systems. The effects of this bottleneck were first revealed in our work in [47], which continued to show how a novel XOR design (comprized of $\frac{K\gamma}{L}+1$ subfiles instead of $K\gamma+1$) can reduce the per-transmission CSI requirements from $K\gamma + L$, i.e. CSI that scaled with the number of users, to CSI equal to L, i.e. completely untangled from the number of users. Moreover, the work also showed that this novel XOR design allows for a more efficient reuse of the acquired CSI, by transmitting exponentially more than in the previously known algorithms for the same amount of CSI, thus making the result relevant even for longer coherence periods.

This chapter is comprized of three delivery algorithms that show the progress we made in designing algorithms that require low CSI, beginning from a very high subpacketization requirement and managing to reduce it in subsequent works. Specifically,

- the first algorithm that we will discuss shows that it is possible to achieve the multi-antenna DoF of $D_L(K,\gamma) = L + K\gamma$, while requiring L CSIT/CSIR vectors per transmission. This allowed, for the first time, to completely untangle the feedback cost from the DoF gains.

- Subsequently, the second algorithm that we will present improves upon the previous result by maintaining the low CSI costs, while at the same time reducing the subpacketization to (approximately) match the subpacketization of the single antenna case.

- Finally, the third algorithm that we present has the slightly reduced DoF $D_L(K,\gamma) = (L+K\gamma)(1-\gamma)$, but requires L CSIT/CSIR feedback costs per transmission and further reduces the subpacketization of the previous scheme by an exponentially large factor thus, making progress into combining the subpacketization reductions of multiple-antenna systems with the desirable low CSI costs.

Chapter 5 - Uneven cache sizes In this chapter we will explore the fundamental performance of both the single and multiple antenna channels, where users are equipped with caches of uneven sizes. For the two type setting of interest (where K_1 cache-aided users are equipped with caches of normalized size γ_1 and K_2 users are equipped with normalized cache-size $\gamma_2 \in [0, \gamma_1)$), we first prove, with the help of Index Coding tools, that cache-less users present a fundamental bottleneck in the single antenna case, i.e.

Theorem. *In a single-stream BC with K_1 cache-aided users equipped with normalized cache of size γ_1 and with K_2 additional cache-less users, the optimal delay, under the assumption of uncoded placement, takes the form*

$$T_1(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1} + K_2. \tag{2.18}$$

Further, we show that adding $L-1$ transmitters allows either an L-boost on the DoF or **full cache-aided DoF for all users**. Specifically,

Theorem. *In the MISO BC with $L \geq 1$ antennas, K_1 cache-aided users equipped with cache of fractional size γ_1, and $K_2 \geq (L-1)T_1^{(1)}$ cache-less users, the delivery time*

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = T_1^{(1)} + \frac{K_1 - (L-1)T_1^{(1)}}{\min\{L, K_2\}} \tag{2.19}$$

is achievable and within a factor of 2 from optimal, while if $K_2 \leq (L-1)T_1^{(1)}$ then

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{K_2 + K_1(1-\gamma_1)}{K_1\gamma_1 + L} \tag{2.20}$$

is achievable and within a factor of 3 from optimal under the assumption of linear and one-shot schemes, where $T_1^{(1)} = \frac{K(1-\gamma)}{1+K\gamma}$.

Moreover, we will show that increasing the number of transmit antennas has the equivalent effect of smoothing-out the cache unevenness and more specifically, we will show that starting from the single-antenna two type, cache-aided and cache-less users, setting by adding cumulative cache Γ_2 to the cache-less users and $L-1$ transmit antennas, the DoF can be increased by a multiplicative factor of up to $\Gamma_2 + L$.

Chapter 6 - Uneven Channels This chapter is concerned with the uneven channel bottleneck that is experienced in wireless channels. For the single antenna setting, we prove a performance converse, and then proceed to design an algorithm that is based on superposition coding and has a multiplicative gap from the optimal of at most 4.

Theorem. *In the K-user SISO Broadcast Channel with single antenna receivers, receiver channel strengths $\{\alpha_k\}_{k=1}^{K}, \alpha_k \leq \alpha_{k+1}, \ \forall k \in [K]$ and each receiver equipped with a cache of normalized size γ, the achieved worst-case delivery time takes the form*

$$T_{sc}(K,\gamma,\boldsymbol{\alpha}) = \max_{w\in[K]} \left\{ \frac{1}{\alpha_w} \cdot \frac{\binom{K}{K\gamma+1} - \binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}} \right\} \tag{2.21}$$

and has a multiplicative gap from the optimal performance of at most 4.

Chapter 7 - Partially connected cache-aided networks In this chapter we will study partially connected models. The first is inspired by Wyner's network [48] that models interfering mobile cells and where any single transmitter is connected to two subsequent receivers. When transmitters have limited backhaul access and receivers are equipped with caches, we explore how caching can alleviate the backhaul load and how users that may not be connected to the same transmitter can take advantage of coded caching gains. We proceed with the main result of this work.

Theorem. *In the Wyner's network with per-transmitter maximum backhaul load $M_T \cdot f$ bits and a normalized cache size at each receiver of a fraction γ of the library, the per-user (interference-free) DoF of $d(M_T,\gamma) = 1$ can be achieved with the following pairs for any $x \in \mathbb{N}$:*

$$d\left(\frac{1-\gamma^2}{4\gamma}, \gamma = \frac{1}{2x+1}\right) = 1, \tag{2.22}$$

$$d\left(\frac{1}{4\gamma}, \gamma = \frac{1}{2x}\right) = 1. \tag{2.23}$$

For this setting we will show how introducing caches to the users can provide exponential savings in the the backhaul load.

Further, we will also study the K transmitter, K receiver network where it is assumed that a transmitter is connected via a high (unit) capacity link to its corresponding receiver, while to any other receiver by a weaker link, characterized by parameter $\alpha \leq 1$ (see [49]). Both transmitters and receivers are equipped with caches and it is also assumed that there is no CSIT.

In this setting, we will explore how the topological factor can be combined with coded caching in order to take advantage of both topological gains and coded caching gains in an effort to counter the lack of CSIT.

The results show, for example, that when each transmitter has access to the whole library then the performance of the system is favored by both the topological factor and the caching gains. More formally,

Theorem. *In the K-user topological MISO BC with parameter α and receiver-side caches of normalized size γ, the cache-aided GDoF take the form*

$$D_\alpha(K, \gamma_T = 1, \gamma_R) = K(1-\alpha) + (K\gamma_R + 1)\alpha. \tag{2.24}$$

Chapter 8 - Coded MapReduce This last chapter studies the bottlenecks of distributed computing under the MapReduce framework. While earlier works [14,15] have showed a trade-off between computation and communication that can boost MapReduce, nevertheless some notable bottlenecks arose. In the Coded MapReduce (CMR) framework the main idea is to increase the amount of computation at each node (Mapping phase), in exchange for a lower communication load (Shuffling Phase). CMR capitalizes on the core concept of Coded Caching i.e., the transmission of coded messages and the decoding of those messages through the use of side information. In the above framework, we will present two contributions that take advantage of the potentially high-dimensionality (high-rank) of Device-to-Device (D2D) systems. The first is inspired by the subpacketization reductions in multi-antenna environments where we will show that this reduction can be naturally applied in CMR. Further, we apply the results from the uneven cache-size chapter to show how a system with heterogeneous nodes (nodes have different computing capabilities) can perform as its equivalent homogeneous system.

The first bottleneck is the subpacketization constraint that, as in the caching case, is attributed to the need to encode across many users. Our contribution lies in completely resolving this bottleneck in the wireless case, while also showing that a wired case can take advantage of this solution. We summarize the main result in the following theorem.

Theorem. *In the K-node distributed computing setting where the dataset can only be split into at most $S_{\max}$ identically sized packets, the proposed Group-based Coded MapReduce algorithm with groups of L users, can achieve communication delay*

$$T_{shuf}^{GCMR} = \frac{1-\gamma}{\bar{t}_L} T_c \tag{2.25}$$

for

$$\bar{t}_L = \gamma \cdot \arg\max_{\bar{K} \le K} \left\{ \binom{\frac{\bar{K}}{L}}{\frac{\bar{K}\gamma}{L}} \le S_{\max} \right\}. \tag{2.26}$$

Note 2.1. *The subpacketization constraint is further accentuated in CMR compared to the caching case, by the fact that each initial dataset segment (subfile) will be further mapped to one out of a set of intermediate results. This mapping, for example, in the case of sorting [50] arranges each of the values of a segment, from the vector that needs to be sorted, into one out of the K (equal to the nodes) different intermediate families. If the file segment does not contain a large number of numbers, then the resulting intermediate families will contain uneven number of elements thus before the multicast transmission they*

will require zero-padding in order to be transmitted together, resulting in a further reduction of the multicasting gains.

Further, in the remaining part of this chapter we will discuss the heterogeneous computing bottleneck, where nodes that possess different computing capabilities are tasked to perform the same CMR problem. The main difficulty in this scenario is that faster nodes will finish the mapping phase before their slower counterparts, which will result in delays (faster nodes will wait for the slower nodes to complete the mapping phase so that the shuffling phase can commence). We resolve this bottleneck by forcing faster nodes to map more, and fully utilizing this increased redundancy at the shuffling phase.

It is notable that the system's performance is exactly equal to the corresponding homogeneous system's performance thus showing how computing systems are immune to mapping heterogeneity. The main result of this part follows.

Theorem. *The achievable time of a MapReduce algorithm in a heterogeneous distributed computing system with K_1 nodes each able to map fraction γ_1 of the total dataset, while K_2 nodes can map fraction γ_2 of the dataset each, takes the form*

$$T_{tot}^{hCMR}(\gamma_1,\gamma_2)=T_{map}^{(1)}(\gamma_1 f)+\frac{T_{shuf}}{K}\left(\frac{1-\gamma_1}{\gamma_1}+\frac{K_2\left(1-\frac{\gamma_2}{\gamma_1}\right)}{\min\{K_2,K_1\gamma_1+K_2\gamma_2\}}\right)+\frac{Q}{K}T_{red}$$

where the communication (shuffling) cost can be simplified, in the case of $K_2 \geq K_1\gamma_1+K_2\gamma_2$, to

$$T_{comm}^{hCMR}(\gamma_1,\gamma_2)=\frac{T_{shuf}}{K}\frac{K_1(1-\gamma_1)+K_2(1-\gamma_2)}{K_1\gamma_1+K_2\gamma_2}. \tag{2.27}$$

Intuition on the designs

The presented results mark an important element on the designing of Coded Caching algorithms, that of adapting the placement and delivery algorithms to the specific scenario. While the approach of [1] is almost optimal [37] (and under some assumptions optimal [35]) in the single antenna case, where each user has the same cache size, nevertheless by maintaining some fundamental elements of the algorithm (both in the case of placement and that of delivery) from [1], but changing others, we show that the emerged gains can be substantial and in many cases multiplicative.

This becomes evident, for example, in Chapter 4 where the reduction of the feedback costs was achieved by the modification of the XORs to be composed not of $K\gamma+1$ subfiles as in previous algorithms, but designed to carry $\frac{K\gamma}{L}+1$ subfiles, thus allowing us to have a class of users that can "cache-out" a (reduced number of) $\frac{K\gamma}{L}$ subfiles, and some users that could cache out a much increased $K\gamma+L-1$ subfiles, thus alleviating the role of beamforming for canceling interference, thus in turn alleviating the cost

of CSI. Further, in the same chapter, we can see that the joint reduction in subpacketization and CSI was achieved through a novel placement algorithm, which asymmetrically assigns content to users, by dividing a file into parts and where only a subset of the users can store from a specific part. This novel placement created a set of cache-aided and cache-less users that are requesting content simultaneously and, by taking advantage of the delivery algorithm from the uneven cache-size chapter we were able to maintain the DoF performance, and at the same time to reduce the problem dimensionality which then reduces the subpacketization

Another notable example is the major reductions achieved in the subpacketization bottleneck. In this scenario, going against the core philosophy of Coded Caching, i.e. the requirement for partially overlapping caches, we assigned caches that are identical, and then combined this with a decomposition principle which allowed us to separate the network into L non-interfering (single-antenna) coded caching networks. The challenge comes naturally from the fact that, of course, if (for example) you have two antennas and $K = 100$ users, you cannot simply use these two antennas to "separate" the first 50 users from the other 50 users (two antennas can protect one user, not 50). This was essentially achieved with the decomposition-based algorithm that we proposed which avoids XORs and brings forward massive reductions in the required subpacketization, and which have not been experienced before.

Chapter 3

The role of transmitters in resolving the Subpacketization Bottleneck

A main assumption in Coded Caching algorithms is that the file size can be potentially infinite. In reality, though, if one constraints this file size into the finite size region, then the theoretical gains will be significantly reduced.

The observation that subpacketization can severely restrict coded caching gains was first made in [39], in the context of decentralized Coded Caching[1]. Specifically, the authors of [39] showed that the algorithm of [51] can provide effective caching gains of approximately $\bar{G} = 1$ under the assumption of a subpacketization constraint

$$S_{\max} \leq \frac{e^{K\gamma}}{K\gamma}. \tag{3.1}$$

Further the authors prove an upper subpacketization bound for the decentralized case showing that the effective gain $\bar{G} = \frac{4}{3} \cdot g - 1$ would require at least subpacketization

$$S_{\max,\text{FLA}} \geq \mathcal{O}\left(\frac{g}{K}\left(\frac{1}{\gamma}\right)^{g-1}\right). \tag{3.2}$$

In other words, the above result states that any Coded Caching gains, achieved in the decentralized setting, will always be restricted by the logarithm of the file size.

3.1 The road to low-subpacketization

The subpacketization bottleneck sparked significant interest in designing Coded Caching algorithms which can provide further caching gains under

[1]Decentralized Coded Caching (cf. [51]) removes the assumption of a known number of users during the placement phase and proceeds to cache without this knowledge, usually by breaking a file into subfiles and storing each subfile randomly with probability γ.

reduced subpacketization costs. A first breakthrough came with the work in [52] (see also [53]) which reformulated the Coded Caching problem into a *placement-delivery array* (PDA) combinatorial design problem, and which exploited interesting connections between Coded Caching and distributed storage to design an algorithm that provided the maximum theoretical caching gain of

$$G_{1,\text{PDA}} = K\gamma - 1 \tag{3.3}$$

i.e, treating a total of $K\gamma$, rather than $K\gamma+1$, users at a time, at the reduced subpacketization of

$$S_{1,\text{PDA}} = \left(\frac{1}{\gamma}\right)^{K\gamma} = \left(\frac{1}{\gamma}\right)^{G_{1,\text{PDA}}+1} \tag{3.4}$$

thus allowing — under some constraints on the operating parameters — for an effective caching gain of

$$\bar{G}_{1,\text{PDA}} = \min\left\{\frac{\log S_{\max}}{\log\frac{1}{\gamma}} - 1,\ K\gamma - 1\right\}. \tag{3.5}$$

Similar conclusions were also drawn in [53] which used linear codes (LC) over high-order finite fields, to create set partitions that identify — under some constraints on the values of γ — how the subpackets are cached and delivered, thus allowing for a tradeoff between an adjustable theoretical gain $G_{1,\text{LC}} \leq K\gamma - 1$ and the corresponding subpacketization $S \approx \left(\frac{1}{\gamma}\right)^{G_{1,\text{LC}}}$, resulting in a similar effective gain of

$$\bar{G}_{1,\text{LC}} \approx \frac{\log S_{\max}}{\log\frac{1}{\gamma}} - 1 \tag{3.6}$$

(naturally again the effective gain $\bar{G}_{1,\text{LC}}$ cannot exceed the theoretical gain $G_{1,\text{LC}}$).

Another breakthrough was presented in [54] which took a hyper-graph theoretic approach to show that there do not exist caching algorithms that achieve a constant delivery time $T_1(K,\gamma)$, i.e. independent of K, with subpacketization that grows linearly[2] with K. This work also provided constructions which nicely tradeoff performance with subpacketization, which require though (see [54, Construction 6]) that $K > 4/\gamma^2$ (approximately) in order[3] to have gains bigger than 1.

Another milestone of a more theoretical nature was the very recent work in [55] which employed the Ruzsa-Szeméredi graphs to show for the first time that, under the assumption of (an unattainably) large K, one can get

[2] This assumes that γ is independent of K, that each file is divided into an identical number of subpackets, and also assumes uncoded cache placement.

[3] K must be large because the theoretical gain is reduced and is approximately $K\gamma^2/4$. K must also be (essentially) a square integer; square integers become rarer as K increases.

a (suboptimal) gain that scales with K, with a subpacketization that scales with $K^{1+\delta}$ for some arbitrarily small positive δ.

While indeed different new algorithms provide exponential reduction in subpacketization, the corresponding improvement on the actual gain $\bar{G}$ — over the original (MN) algorithm in [1], for realistic values of γ and $S_{\max}$ — remains hard bounded and small. For example, for $\gamma \leq 1/20$ and $S_{\max} \leq 10^5$, no known algorithm can improve over the MN algorithm's effective caching gain (and effective DoF) by more than two[4] (2 additional users served at a time) (see also Section 3.3.1).

3.2 Multiple Transmitters and Subpacketization

As we have seen, works [10, 11] showed the increased DoF gains that can be achieved through the combination of multiple antennas or multiple transmitters, with coded caching. These methods, though, required even higher subpacketization.

Specifically, in the the Multi-Server (MS) setting of the work in [10], the required subpacketization takes the form

$$S_{L,\text{MS}} = \binom{K - K\gamma - 1}{L - 1}\binom{K}{K\gamma} \approx \left(\frac{K}{L}\right)^{L} \cdot \left(\frac{1}{\gamma}\right)^{K\gamma} \tag{3.7}$$

while in the cache-aided Interference Channel (IC) setting of [11] the required subpacketization takes the form

$$S_{L,\text{IC}} = \binom{K_T}{K_T\gamma_T}\binom{K}{K\gamma}\binom{K - K\gamma - 1}{L - 1}(K\gamma)!(K_T\gamma_T - 1)! \tag{3.8}$$

$$\gg \left(\frac{1}{\gamma_T}\right)^{K_T\gamma_T} \cdot S_{L,\text{MS}}. \tag{3.9}$$

Chapter Overview

In this chapter we will present the result of [56], which constitutes the first work that considered the use of antennas or transmitters to reduce the subpacketization. The result shows that the order-optimal DoF, $D_L(K,\gamma) = L + K\gamma$ in the multi-antenna case and the order optimal DoF $D_{K_T,\gamma_T}(K,\gamma)$ can be achieved with respective subpacketizations

$$S_L = \binom{\frac{K}{L}}{\frac{K\gamma}{L}} \tag{3.10}$$

$$S_{K_T\gamma_T} = \binom{\frac{K}{K_T\gamma_T}}{\frac{K\gamma}{K_T\gamma_T}}. \tag{3.11}$$

[4] This best-known improvement is due to [54, Construction 6] ($a = b = 2, \lambda = 40$) which encodes over $\bar{K} = 3160$ users to give an effective DoF of 6, while the MN algorithm gives a DoF of 4 (with $\bar{K} = 60$).

In practise, the above results show that when considering the transmission of finite length files, having L antennas allows for up to $\times L$ coded caching gains, as we will show later on.

Another interesting point, is that the above result eliminates any subpacketization associated with the multiplexing gains.

3.3 The role of Transmitters in Subpacketization

We present the main results, first for the integer case where $L|K$ and $L|K\gamma$. The interpolation to all cases K, L is easily handled using memory sharing, and as we note later on, does not result in substantial performance degradation. The details for this are handled in Section 3.4.

Theorem 3.1. *In the cache-aided MISO BC with L transmitting antennas and K receiving users, the delay of $T_L(K,\gamma) = \frac{K(1-\gamma)}{L+K\gamma}$ and the corresponding DoF $D_L(K,\gamma) = L + K\gamma$, can be achieved with subpacketization*

$$S_L = \binom{\frac{K}{L}}{\frac{K\gamma}{L}}. \tag{3.12}$$

Proof. We begin by organizing the users into groups of size L i.e., a total of $\frac{K}{L}$ groups. The significance of a group is that users belonging in the same group will cache *exactly* the same content, a proposal that comes in contrast to Coded Caching's core idea, where cache overlap needs to be partial in order to allow for multicasting opportunities to emerge.

We denote the $\frac{K}{L}$ groups by $\mathcal{G}_\kappa,\ \kappa \in \left[\frac{K}{L}\right]$ while, for tractability, we assume the following placement of a user $U_k,\ k \in [K]$ into a group

$$\mathcal{G}_{\kappa\in\left[\frac{K}{L}\right]} = \left\{U_\kappa, U_{\kappa+\frac{K}{L}}, ..., U_{\kappa+(L-1)\cdot\frac{K}{L}}\right\}. \tag{3.13}$$

Group-based placement phase We begin the placement phase by segmenting files into subpackets according to

$$S_L = \binom{\frac{K}{L}}{\frac{K\gamma}{L}} \tag{3.14}$$

i.e., a file is divided into

$$W^n \to \left\{W^n_\tau,\ \tau \subset \left[\frac{K}{L}\right],\ |\tau| = \frac{K\gamma}{L}\right\}. \tag{3.15}$$

In this case, comparably to the Coded Caching algorithm of [1], index τ will denote which group of users have this subfile cached.

In the placement phase which, as mentioned before, is based on the notion that users belonging to the same group will have identically cached content, the caches are filled as follows

$$\mathcal{Z}_{k\in\mathcal{G}_\kappa} = \{W^n_\tau\ : \kappa \in \tau, \forall n \in [N]\}. \tag{3.16}$$

Note 3.1. *At this point we can notice that the above placement is based on the algorithm of [1], where instead of considering K users with cumulative cache $K\gamma$, the algorithm that we present segments the files as if only $\frac{K}{L}$ users where in the channel, with cumulative cache $\frac{K\gamma}{L}$.*

Furthermore, it is interesting to note that we could have instead used any single-antenna algorithm for the placement (e.g. [52–54]), again caching same content to the users of a group while the files would be subpacketized according to parameters $\frac{K}{L}$ and $\frac{K\gamma}{L}$. This connection is further explored in Section 3.3.1.

Group-based delivery of content The delivery process is described in the form of a pseudo-code in Algorithm 3.1.

for *all* $\sigma \subseteq \left[\frac{K}{L}\right]$, $|\sigma| = \frac{K\gamma}{L} + 1$ *(Select* $\frac{K\gamma}{L} + 1$ *groups)* **do**
  Let $\mathcal{G}_\mu = \{g_{\mu,1}, g_{\mu,2}, ..., g_{\mu,L}\}$.
  Transmit:

$$\mathbf{x}_\sigma = \sum_{\mu \in \sigma} \mathcal{H}_{\mathcal{G}_\mu}^{-1} \cdot \begin{bmatrix} W_{\sigma\setminus\{\mu\}}^{d_{g_{\mu,1}}} \\ \vdots \\ W_{\sigma\setminus\{\mu\}}^{d_{g_{\mu,L}}} \end{bmatrix}. \tag{3.17}$$

end

Algorithm 3.1: Group-based Delivery Process

Algorithm 3.1 selects, in every iteration, a new subset of $\frac{K\gamma}{L} + 1$ groups. Then, it creates the linear combination of $\frac{K\gamma}{L} + 1$ vectors, where each vector is of size L and contains information for the users of a specific group. For example one of these vectors, intended for users of group $\kappa \in \sigma$, is

$$\mathcal{H}_{\mathcal{G}_\kappa}^{-1} \cdot \begin{bmatrix} W_{\sigma\setminus\{\kappa\}}^{d_{g_{\kappa,1}}} \\ \vdots \\ W_{\sigma\setminus\{\kappa\}}^{d_{g_{\kappa,L}}} \end{bmatrix} \tag{3.18}$$

where $\mathcal{H}_{\mathcal{G}_\kappa}^{-1}$ is the ZF-precoding matrix that spatially separates the users of the group. We can see that every user of a group will receive, from its desired file, *exactly* the same index as the other same-grouped users, i.e. subfile indexed by $\sigma \setminus \{\kappa\}$.

Thus, in each iteration of the algorithm a total of $L \cdot \left(\frac{K\gamma}{L} + 1\right) = L + K\gamma$ users will be treated, were we proceed to show that each user will be able to successfully decode its file.

Decodability Assuming user $k \in \mathcal{G}_\kappa$, then the received message takes the form (noise is suppressed for simplicity)

$$y_k = \mathbf{h}_k^T \cdot \sum_{\lambda \in \sigma} \mathcal{H}_{\mathcal{G}_\lambda}^{-1} \cdot \begin{bmatrix} W_{\sigma\backslash\{\lambda\}}^{d_{g_{\lambda,1}}} \\ \vdots \\ W_{\sigma\backslash\{\lambda\}}^{d_{g_{\lambda,L}}} \end{bmatrix}. \tag{3.19}$$

First, user $k \in \mathcal{G}_\kappa$ can remove any subfile that has index $\tau \ni \kappa$, using its cached content and the acquired CSI. Then, the remaining part of the received signal is

$$y_k^{\text{rem}} = \mathbf{h}_k^T \cdot \mathcal{H}_{\mathcal{G}_\kappa}^{-1} \cdot \begin{bmatrix} W_{\sigma\backslash\{\kappa\}}^{d_{g_{\kappa,1}}} \\ \vdots \\ W_{\sigma\backslash\{\kappa\}}^{d_{g_{\kappa,L}}} \end{bmatrix} = W_{\sigma\backslash\{\kappa\}}^{d_k}. \tag{3.20}$$

The remaining signal (Eq. (3.20)) constitutes a ZF-precoded signal which, due to the design of the precoder, will separate the messages of the users so it will provide each user with its desired subfile, free from any inter-group interference. □

Note 3.2. *The above algorithm has been presented for the case of a single transmitter with L antennas. Here we will see how this algorithm can be extended to the case of multiple transmitters (K_T) each with cache of size equivalent to M_T files, where these transmitters are equipped with one or more antennas.*

We begin with the placement of content to the transmitters, where we impose two important rules for a successful placement policy.

- *To cache each file at $K_T \cdot \gamma_T$ different transmitters, and*
- *to avoid, as much as possible, any further, unnecessary subpacketization.*

The above two points will permit firstly, that a file is cached at a maximal amount of transmitters so as to succesfully perform ZF-precoding, and secondly to achieve the highest possible caching gains under finite file sizes, by constraining the amount of subpacketization due to transmitter side caching.

A placement algorithm that respects both above points, and more importantly that requires no additional subpacketization, is the following

$$\mathcal{Z}_{k_T \in [K_T]} = \Big\{ W^m : m \in \{1 + [(k_T - 1) \cdot M_T \mod N], ..., 1 + [(k_T \cdot M_T - 1) \mod N]\} \Big\}. \tag{3.21}$$

Then, using the above placement of files at the transmitters we can construct the transmitted message of Eq. (3.17) by creating each ZF precoded vector in a distributed manner and by further allowing these $\frac{K\gamma}{L} + 1$ vectors to combine linearly in the air.

3.3.1 Elevating different coded caching algorithms to the L antenna setting

The aforementioned subpacketization can be further reduced when considering alternative coded caching algorithms. We recall that the scheme that we have presented, involved 'elevating' the original MN algorithm in [1], from the single-stream scenario ($L = 1$) with $K' = K/L$ users, to the L-antenna case with K' groups of L-users per group. This same idea can apply *directly* to other centralized coded caching algorithms like those in [52, 53, 55], in which case the steps are almost identical:

- Choose the new coded caching algorithm for the single-stream K'-user scenario.
- Split the K users into K' groups of L users each, and employ the new algorithm to fill the caches as in the K'-user single-stream case, as if each group is a user, such that same-group users have caches that are identical.
- Using the coded caching algorithm for the single-stream K'-user scenario, generate the sequence of XORs. Each XOR consists of $D_1(K', \gamma)$ summands, where $D_1(K', \gamma)$ is the theoretical sum-DoF provided by the coded caching algorithm in the K'-user single-antenna (single stream) BC.
- Each element (summand) of the XOR, corresponds to a group of users, and each such XOR summand is replaced by a (precoded) L-length vector that carries the L-requests of the associated group. Add these $D_1(K', \gamma)$ vectors together, to form a composite transmitted vector that corresponds to the XOR.
- Each composite vector treats a total of $D_1(K', \gamma)$ groups at a time, i.e., treats $L \cdot D_1(K', \gamma)$ users at a time.
- Then continue with the rest of the XORs.

We recall that when[5] elevating the MN algorithm – which, for the single-stream K'-user case, treats $D_1(K', \gamma) = K'\gamma + 1$ users at a time – we treated $D_1(K', \gamma) = K'\gamma + 1$ groups at a time, thus treating a total of $D_L(K, \gamma) = L \cdot D_1(K', \gamma) = L + K\gamma$ users at a time.

On the other hand, when elevating for example the algorithms in [52, 53], we would naturally have to change the cache placement and the sequence of XORs, and we would have to account for the fact that – for the single stream K'-user case – the algorithm treats $D_{1,\text{PDA}}(K', \gamma) = K'\gamma$ users at a time (not $K'\gamma + 1$), and thus for $L \geq 1$, we would treat $D_{1,\text{PDA}}(K', \gamma) = \frac{K}{L}\gamma$ groups at a time ($L \leq K\gamma$), thus treating a total of $D_{L,\text{PDA}}(K, \gamma) = L \cdot D'_{1,\text{PDA}} = K\gamma$ users at a time (not $K\gamma + L$).

[5] We will use the term 'elevate' to correspond to when we apply a single-stream coded caching algorithm to the multi-antenna case, via the above sequence of steps.

The following corollary describes the effective caching gain provided by the scheme that elevates to the L antenna case, the placement-delivery array (PDA) and linear code (LC) algorithms in [52] and [53]. These algorithms exist under some constraints on γ.

Corollary 3.1. *Given a maximum allowable subpacketization $S_{\max}$, the effective caching gain of the here-elevated PDA and LC algorithms, takes the form*

$$\bar{G}_{L,PDA} = \bar{G}_{L,LC} = \min\left\{ L \cdot \frac{\log S_{\max}}{\log(\frac{1}{\gamma})} \, , K\gamma - L \right\}. \tag{3.22}$$

Proof. With a theoretical gain $G_{L,\text{PDA}} = D_{L,\text{PDA}}(K,\gamma) - D_{L,\text{PDA}}(\gamma = 0) = K\gamma - L$, the underlying subpacketization

$$S_{L,\text{PDA}} = \left(\frac{1}{\gamma}\right)^{K'\gamma - 1} \tag{3.23}$$

can be written as

$$S_{L,\text{PDA}} = \left(\frac{1}{\gamma}\right)^{\frac{G_{L,\text{PDA}}}{L}} \tag{3.24}$$

and thus the effective gain is $\bar{G}_{L,\text{PDA}} = L \cdot \frac{\log S_{\max}}{\log(\frac{1}{\gamma})}$, which is bounded by the theoretical caching gain $K\gamma - L$ offered by the scheme in the absence of subpacketization constraints. □

The L-fold impact of antennas to alternative single-antenna coded caching algorithms

The fact that the underlying coded caching algorithm is used in the design at the level of groups of users, implies that any difference in the effective caching gain between two underlying algorithms in the single-stream case, will be magnified — once each algorithm is elevated to the L-antenna case as was shown here — by a factor of up to L. For example, if we were to compare the elevated MN scheme to, say, the aforementioned elevated PDA and LC schemes, we would see (cf. Corollary 3.1 and Corollary 3.3) that

$$\bar{G}_{L,\text{PDA}} = \min\left\{ L \cdot \frac{\log S_{\max}}{\log(\frac{1}{\gamma})}, K\gamma - L \right\} \tag{3.25}$$

$$\bar{G}_{L} \geq \min\left\{ L \cdot \frac{\log S_{\max}}{1 + \log(\frac{1}{\gamma})}, \; K\gamma \right\} \tag{3.26}$$

which would tell us that (when $K\gamma$ is an integer) the improvement in effective gains is bounded as

$$\bar{G}_{L,\text{PDA}} - \bar{G}_{L} \leq L \cdot \frac{\log S_{\max}}{\log(\frac{1}{\gamma})(1 + \log(\frac{1}{\gamma}))}. \tag{3.27}$$

When $L = 1$, this improvement — under realistic assumptions on γ and $S_{\max}$ — can be small, but when the algorithm is elevated to the multi-antenna setting, this improvement increases as a multiple of L.

Remark 3.1. *This implies that the method proposed here, rather than bypassing the need for novel single-stream coded caching algorithms of reduced subpacketization, it in fact accentuates the importance of searching for such algorithms.*

3.3.2 Example

Let us assume a MISO BC with $L = 5$ transmit antennas, $K = 50$ users and user caches of normalized size $\gamma = \frac{3}{10}$. We will show how the DoF of

$$D_L(K, \gamma) = L + G = L + K\gamma = 5 + 15 = 20 \tag{3.28}$$

can be achieved with subpacketization $S_L = 120$.

Placement First, we assign the $K = 50$ users into $K' = 10$ groups of $L = 5$ users per group i.e.,

$$\mathcal{G}_1 = \{1, 11, 21, 31, 41\}, \ldots, \mathcal{G}_{10} = \{10, 20, 30, 40, 50\}.$$

Since $K'\gamma = 3$, we split each file W^n into $|\tau| = \binom{K'}{K'\gamma} = 120$ parts

$$W^n = \{W^n_{(1,2,3)}, W^n_{(1,2,4)}, \ldots, W^n_{(1,3,4)}, \ldots, W^n_{(8,9,10)}\}$$

and then fill the caches

$$\begin{aligned} \mathcal{Z}_{\mathcal{G}_1} &= \{W^n_{(1,2,3)}, W^n_{(1,2,4)}, \ldots W^n_{(1,3,4)}, \ldots W^n_{(1,9,10)}\}_{n=1}^N \\ &\vdots \\ \mathcal{Z}_{\mathcal{G}_{10}} &= \{W^n_{(1,2,10)}, W^n_{(1,3,10)}, \ldots, W^n_{(2,3,10)}, \ldots W^n_{(8,9,10)}\}_{n=1}^N \end{aligned}$$

as described. We will serve $K'\gamma + 1 = 4$ groups at a time.

Delivery We start by treating the group clique $\sigma = (1, 2, 3, 4)$ first. Let

$$\boldsymbol{w}^{\kappa}_{\tau} = \left[W_{\tau}^{d_{\mathcal{G}_\kappa(1)}}, W_{\tau}^{d_{\mathcal{G}_\kappa(2)}}, W_{\tau}^{d_{\mathcal{G}_\kappa(3)}}, W_{\tau}^{d_{\mathcal{G}_\kappa(4)}}, W_{\tau}^{d_{\mathcal{G}_\kappa(5)}}\right]^T \tag{3.29}$$

be the information vector comprized of $L = 5$ subfiles with index τ, that are meant for the users of group $\mathcal{G}_\kappa$.

For these 4 groups, i.e. 20 users, the transmission takes the from

$$\boldsymbol{x}_{(1,2,3,4)} = \mathcal{H}^{-1}_{\mathcal{G}_1}\boldsymbol{w}^1_{(2,3,4)} + \mathcal{H}^{-1}_{\mathcal{G}_2}\boldsymbol{w}^2_{(1,3,4)} + \mathcal{H}^{-1}_{\mathcal{G}_3}\boldsymbol{w}^3_{(1,2,4)} + \mathcal{H}^{-1}_{\mathcal{G}_4}\boldsymbol{w}^4_{(1,2,3)}. \tag{3.30}$$

Decoding Receiver 1 can immediately remove — using its cache — the last three summands in (3.30), while ZF-precoding can remove the unwanted $L-1=4$ elements from $\boldsymbol{w}^1_{(2,3,4)}$.

3.3.3 Effective gains and multiplicative boost of effective DoF

We recall that in the absence of subpacketization constraints, adding extra transmitting antennas, takes us from a theoretical DoF $D_1(K,\gamma)=1+K\gamma$ to $D_L(K,\gamma)=L+K\gamma$ (cf. [10]), leaving the theoretical caching gain unaffected, and adding $D_L(K,\gamma)-D_1(K,\gamma)=L-1$ DoF. For example, adding one extra antenna (going from $L=1$ to $L=2$) simply allows us to add one extra served user per second per hertz.

What the result of Theorem 3.1 shows is that, when subpacketization is taken into consideration, adding extra transmitting antennas (or adding extra transmitter-side caching) can have a much more powerful, multiplicative impact on the effective gains.

Recall from Eq. (2.5) that for $L=1$, the subpacketization takes the form $S_1=\binom{K}{K\gamma}$, which – as we briefly argued before – means that having a maximum allowable subpacketization S_{max}, limits the number of users we can encode over, from K to a smaller

$$\bar{K}_1 \triangleq \arg\max_{K^o\leq K}\left\{\binom{K^o}{K^o\gamma}\leq S_{max}\right\}. \tag{3.31}$$

On the other hand, in the L antenna case, the reduced subpacketization cost $S_L=\binom{\frac{K}{L}}{\frac{K\gamma}{L}}$ allows us, for the same constraint S_{max}, to encode over

$$\bar{K}_L \triangleq \arg\max_{K^o\leq K}\left\{\binom{\frac{K^o}{L}}{\frac{K^o\gamma}{L}}\leq S_{\text{max}}\right\}=\min\{L\cdot\bar{K}_1,K\} \tag{3.32}$$

users, just because the transition from S_1 to S_L reflects a simple substitution of K by K/L. Going from 1 to L antennas, allows us to encode over L times as many users (up to K), which in turn offers L times more caching gain

$$\bar{G}_L=\min\{L\cdot\bar{G}_1,G\}$$

up to the theoretical $G=K\gamma$. Specifically if $\binom{\frac{K}{L}}{\frac{K\gamma}{L}}\leq S_{\text{max}}$ then $\bar{G}_L=G$ (corresponding to a multiplicative boost of $\frac{\bar{G}_L}{\bar{G}_1}=\frac{G}{\bar{G}_1}$), else the effective gain and the effective sum-DoF both experience a multiplicative increase by a factor of exactly L. For completeness this is represented in the following corollary, which ignores for now integer rounding-off effects. The corollary follows directly from the above.

Corollary 3.2. *Under a maximum allowable subpacketization $S_{\max}$, the multi-antenna effective caching gain and DoF take the form*

$$\bar{G}_L = \min\{L \cdot \bar{G}_1, G = K\gamma\} \tag{3.33}$$
$$D_L(\bar{K}, \gamma) = \min\{L \cdot D_1(\bar{K}, \gamma), D_L(K, \gamma) = L + K\gamma\} \tag{3.34}$$

which means that with extra antennas, the (single-antenna) effective DoF $D_1(\bar{K}, \gamma)$ is either increased by a multiplicative factor of L, or it reaches the theoretical (unconstrained) DoF $D_L(K, \gamma) = L + K\gamma$.

Remark 3.2. *What we saw is that this L-fold multiplicative DoF boost stays into effect as long as $\binom{\frac{K}{L}}{\frac{K\gamma}{L}} \geq S_{\max}$, so in essence it stays into effect as long as subpacketization remains an issue.*

The following corollary bounds the derived effective caching gain $\bar{G}_L$.

Corollary 3.3. *Given a maximum allowable subpacketization $S_{\max}$, the effective caching gain of the presented scheme is bounded as*

$$\bar{G}_L \geq \min\left\{ L \cdot \frac{\log S_{\max}}{1 + \log(\frac{1}{\gamma})}, K\gamma \right\}. \tag{3.35}$$

Proof. This follows directly from Sterling's approximation which bounds subpacketization as $S_L = \binom{K'}{K'\gamma} \leq \left(\frac{e}{\gamma}\right)^{K'\gamma} = \left(\frac{e}{\gamma}\right)^{\frac{G}{L}}$ which directly implies that $\bar{G}_L \geq L \cdot \frac{\log S_{\max}}{1+\log(\frac{1}{\gamma})}$ (up to the theoretical gain $G = K\gamma$). □

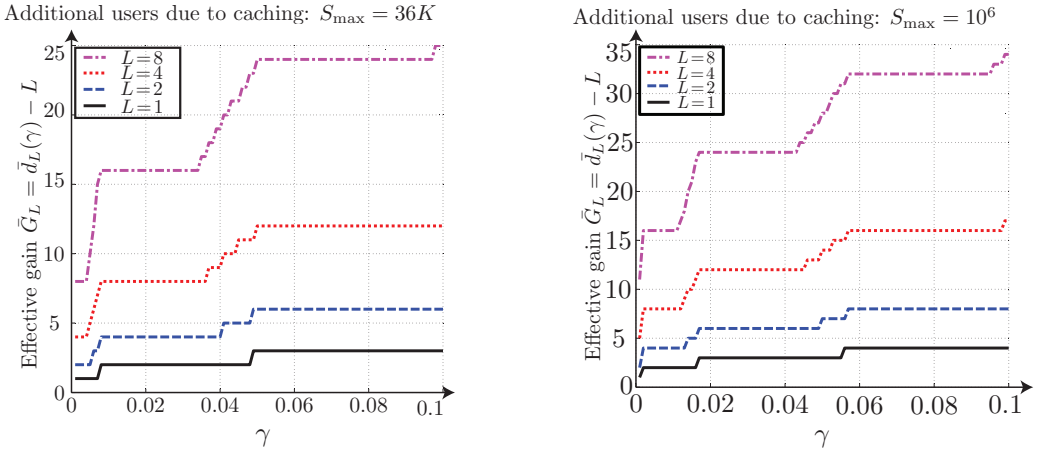

Figure 3.1: Maximum achievable effective caching gain $\bar{G}_L = D_L(K, \gamma) - L$ (maximized over all possible K), achieved by the new scheme for different L, under subpacketization constraint $S_{\max} = 3.6 \cdot 10^4$ (left) and $S_{\max} = 10^6$ (right).

Practical implication - Making small caches relevant Another benefit of the reduced subpacketization, is the resulting exponential increase in the range of cache sizes that can achieve a given target gain. While in theory, a small γ does not necessarily preclude higher caching gains because we could conceivably compensate by increasing the number of users we encode over, such an increase would increase subpacketization thus again precluding high gains (subpacketization limits would not allow for such an increase in the number of users we encode over). Specifically we recall (cf. (2.7)) that for $L = 1$ the subpacketization is bounded as $S_1 \geq \left(\frac{1}{\gamma}\right)^G$, which means that to meet subpacketization constraint S_{max} and target caching gain G, we need

$$\gamma \geq (S_{\text{max}})^{-1/G}. \tag{3.36}$$

On the other hand, the reduced subpacketization $S_L \geq \left(\frac{1}{\gamma}\right)^{\frac{1}{L}G}$ in the L antenna case (cf. (2.7), after substituting K by K/L), can allow for the same caching gain G (given sufficiently many users to encode over) with only

$$\gamma \geq \left((S_{\text{max}})^{-1/G}\right)^L. \tag{3.37}$$

This exponential reduction in the minimum applicable γ, matches well with the spirit of exploiting caches at the very periphery of the network, where we are expected to find relatively small but abundantly many caches.

3.3.4 Subpacketization cost of complementing the multiplexing gains

The following corollary highlights that, in an L-antenna MISO BC system, the subpacketization cost is not determined by K or $L = \lambda K$, nor by the number of extra users G we wish to add due to caching, but rather by the ratio $x = \frac{D_L(K,\gamma)}{D_L(K,\gamma=0)}$ between the DoF and the multiplexing gain.

Corollary 3.4. *In the L-antenna MISO BC setting, a subpacketization of*

$$S_L = \binom{1/\lambda}{x-1} = \binom{\frac{1}{\lambda}}{\frac{\gamma}{\lambda}} \tag{3.38}$$

can yield a DoF that is x times the multiplexing gain.

Proof. The DoF increase from $D_L(K,\gamma=0) = L$ to $D_L(K,\gamma) = L + K\gamma = x \cdot L, \quad x \in \mathbb{N}$, implies that $K\gamma = L(x-1)$ and that $\gamma = \lambda(x-1)$, which means that the corresponding subpacketization $S_L = \binom{K/L}{K\gamma/L}$ now takes the form

$$S_L = \binom{\frac{1}{\lambda}}{\frac{\gamma}{\lambda}} = \binom{1/\lambda}{x-1}. \tag{3.39}$$

□

Remark 3.3. *This generalization here, from the known single-antenna case where $\lambda = \frac{1}{K}$, $D_1(K,\gamma) = x = K\gamma + 1$, to the L antenna case, is nicely captured by Sterling's approximation which — for $D_L(K,\gamma) = x \cdot L$ — remains fixed at*

$$S_L \in \left[\left(\frac{1}{\gamma}\right)^{x-1}, \left(\frac{e}{\gamma}\right)^{x-1} \right]. \tag{3.40}$$

The result is simply a reflection of the fact that the same subpacketization cost of treating $K'\gamma + 1$ users at a time ($K' = K/L$) in the single-antenna case, now guarantees the treatment of $K'\gamma + 1$ groups at a time.

Example 3.1. *From the above we see that normalized cache sizes $\gamma = \lambda(x-1) = \lambda$ and a subpacketization $S_L = 1/\lambda = K/L$, suffice to double the total cache-free DoF ($x = 2$), while $\gamma = 2\lambda$ and*

$$S_L = \binom{1/\lambda}{2} < \frac{1}{2\lambda^2} \tag{3.41}$$

can triple the number of users served at a time, from L to $3L$. Hence, for example, in a cell of K users served by a multi-antenna base-station that provides $D_L(K, \gamma = 0)/K = L/K = \lambda = 1/30$ cache-free DoF per user, having $\gamma = \frac{xL-L}{K} = \lambda(x-1) = 2\lambda = 2/30$ and $S_{\max} = \binom{1/\lambda}{x-1} = \binom{30}{2} = 435$ would allow caching to triple the number of users served at a time ($x = 3$).

3.3.5 Effects of cache-aided transmitter-cooperation on coded caching

Given Eq. 3.11, it is not difficult to conclude that all the previous corollaries apply directly to the $K_T \times K$ cache-aided interference scenario, after substituting L with $K_T\gamma_T$. In particular, drawing from the previous corollaries, we can summarize the following results that apply to cache-aided transmitter cooperation.

- As the transmitter-side cache redundancy $K_T\gamma_T$ increases, the effective DoF will either be increased by a multiplicative factor of $K_T\gamma_T$, or it will reach the theoretical (unconstrained) DoF $K_T\gamma_T + K\gamma$ (cf. Corollary 3.2).
- In the presence of transmitter-side cache redundancy $K_T\gamma_T$, the effective caching gain is bounded below by $(K_T\gamma_T) \cdot \frac{\log S_{\max}}{1+\log(\frac{1}{\gamma})}$ (cf. Corollary 3.3).
- Increasing the transmitter-side cache redundancy $K_T\gamma_T$, allows for an exponentially reduced minimum applicable $\gamma \geq \left((S_{\max})^{-1/G}\right)^{K_T\gamma_T}$ that can offer a (receiver-side) caching gain of $G = K\gamma$ (cf. Section 3.3.3).
- Subpacketization $S_{K_T\gamma_T} = \binom{K}{\frac{K}{K_T\gamma_T}}{x-1}$ can yield a sum DoF that is x times the cooperative multiplexing gain $K_T\gamma_T$ (cf. Corollary 3.4).

- When the transmitter-side and receiver-side cache redundancies match (i.e., when $K_T\gamma_T = K\gamma$), the DoF $K_T\gamma_T + K\gamma$ can be achieved with subpacketization $S_{K_T\gamma_T} = \frac{K}{K_T\gamma_T}$ (cf. Corollary 3.6).

3.3.6 Near-optimality of schemes

The schemes that we have employed here have the 'one-shot, linear' property which means that each data element is manipulated linearly, and only once (a data bit is not transmitted more than once). This lends all the above results amenable to the analysis in [11] whose outer bound then allows us to directly conclude that the schemes are near optimal. This is described below, for purposes of completeness, in the form of a corollary.

Corollary 3.5. *The described subpacketization*

$$S_L = \begin{pmatrix} \frac{K}{L} \\ \frac{K\gamma}{L} \end{pmatrix} \tag{3.42}$$

$$S_{K_T\gamma_T} = \begin{pmatrix} \frac{K}{K_T\gamma_T} \\ \frac{K\gamma}{K_T\gamma_T} \end{pmatrix} \tag{3.43}$$

guarantees DoF performance that is at most a factor of 2 from the theoretical optimal linear-DoF.

Proof. As stated, the proof is direct from the bound in [11], from the performance achieved by the schemes here, and from the fact that the schemes have the 'one-shot linear' property. □

3.4 Removing the integer constraint

We proceed to remove the constraints $L|K$ and $L|K\gamma$, by applying, as in [1], memory sharing. The results, after removing the integer constraints, will remain approximately the same except for a marginal increase in subpacketization[6] to at most

$$S_L \leq K \cdot \max\left\{ \begin{pmatrix} \lceil K/L \rceil \\ \lceil K\gamma/L + 1 \rceil \end{pmatrix}, \begin{pmatrix} \lceil K/L \rceil \\ \lfloor K\gamma/L + 1 \rfloor \end{pmatrix} \right\} \tag{3.44}$$

and a relatively small reduction in the achieved DoF ($D_L(K,\gamma) = L + K\gamma$) by a multiplicative factor (gap) that is bounded above by 2 when $L > K\gamma$ and by $\frac{3}{2}$ when $L < K\gamma$, while the gap vanishes as $\frac{K\gamma}{L}$ increases.

To remove the constraint $L|K$ we will add to the system "phantom" users such that the new (hypothetical) number of users is $\hat{K} = L\left\lceil \frac{K}{L} \right\rceil$. Moreover, if $L \nmid \hat{K}\gamma$ we will perform memory sharing (cf. [1]) by splitting each file W^n

[6] Note that for the settings in [1,10,11], the aforementioned subpacketization costs do not account for the extra subpacketization costs due to memory sharing.

into two parts, $(W^n)', (W^n)''$ of different sizes $|(W^n)'| = p|W_n|$ and $|(W^n)''| = (1-p)|W^n|$, and cache each part with normalized cache sizes

$$\gamma' = \frac{|\mathcal{Z}_k \cap (W^n)'|}{|(W^n)'|} = \frac{L}{\hat{K}} \left\lfloor \frac{\hat{K}\gamma}{L} \right\rfloor \tag{3.45}$$

$$\gamma'' = \frac{|\mathcal{Z}_k \cap (W^n)''|}{|(W^n)''|} = \frac{L}{\hat{K}} \left\lceil \frac{\hat{K}\gamma}{L} \right\rceil \tag{3.46}$$

which guarantees that $L|\hat{K}\gamma'$ and $L|\hat{K}\gamma''$. This also gives that $p = \frac{\gamma''-\gamma}{\gamma''-\gamma'}$.

Then, as the original scheme describes, we divide $(W^n)'$ into $\binom{\hat{K}/L}{\hat{K}\gamma'/L}$ parts, $(W^n)''$ into $\binom{\hat{K}/L}{\hat{K}\gamma''/L}$ parts, and cache from $(W^n)', (W^n)''$ according to Eq. (3.16). The corresponding subpacketization cost is thus bounded as

$$\begin{aligned} S_L &\leq K \cdot \max\left\{ \binom{\hat{K}/L}{\hat{K}\gamma'/L}, \binom{\hat{K}/L}{\hat{K}\gamma''/L} \right\} \\ &\leq K \cdot \max\left\{ \binom{\lceil K/L \rceil}{\lceil K\gamma/L \rceil + 1}, \binom{\lceil K/L \rceil}{\lfloor K\gamma/L \rfloor + 1} \right\} \end{aligned} \tag{3.47}$$

where the multiplicative factor of K is the one that upper bounds the subpacketization effect of splitting the file in two parts before subpacketizing each part. This effect is bounded by K because $p \geq 1/K$ by virtue of the fact that $K\gamma$ is an integer[7].

Then, in order to derive a multiplicative gap on DoF, $D_L^{nc}(K,\gamma)$, that accounts for removing the two constraints, we will consider two separate cases. First, we will look at the case of $\hat{K}\gamma \leq L$. By applying memory sharing, we can see that each part will be cached with redundancy 0 and L respectively. This means that the completion time will be

$$T = \frac{m'}{0+L} + \frac{m''}{L+L}, \tag{3.48}$$

where $m' = Kp(1-\gamma')$ and $m'' = K(1-p)(1-\gamma'')$. Then, we can see that the completion time is upper-bounded $T \leq \frac{K(1-\gamma)}{L}$ and lower-bounded $T \geq \frac{K(1-\gamma)}{2L}$, which incorporates the facts that the performance cannot be worse than if there were any caching gains, but it cannot be better than if the caching gain was L. Using that, we can calculate the bounds of the DoF as follows

$$\begin{aligned} \frac{K(1-\gamma)}{L} \geq T \geq \frac{K(1-\gamma)}{2L} \\ \frac{K(1-\gamma)}{\frac{K(1-\gamma)}{2L}} \geq D_L^{nc} \geq \frac{K(1-\gamma)}{\frac{K(1-\gamma)}{L}} \\ 2L \geq D_L^{nc}(K,\gamma) \geq L \end{aligned}$$

[7] To see this, we rewrite γ as $\gamma = a/K$ where a is an integer, and then we see that $p = \frac{\gamma''-\gamma}{\gamma''-\gamma'} = \frac{\left\lceil \frac{\hat{K}a}{KL} \right\rceil - \frac{a\hat{K}}{KL}}{\left\lceil \frac{\hat{K}a}{KL} \right\rceil - \left\lfloor \frac{\hat{K}a}{KL} \right\rfloor} > \frac{1}{K}$ where, in the last step we used the fact that the denominator is 1 (unless it is zero, in which case there is no additional subpacketization cost), while for the numerator we have that $\left\lceil \frac{\hat{K}a}{KL} \right\rceil - \frac{a\hat{K}}{KL} > \frac{1}{K}$ because $L|\hat{K}a$.

which implies a gap of 2.

Similarly, for $K\gamma \in (qL, qL+1), \quad q = \{1, 2, ...\}$ we can see that the above gap becomes $\frac{q+1}{q}$ thus, if $L < K\gamma$ then the gap is at most $\frac{3}{2}$.

3.5 Conclusion and Final Remarks

Intuition on design

The design was based on the simple observation that multi-node (transmitter-side) precoding, reduces the need for content overlap. The subpacketization reduction from $\binom{K}{K\gamma}$ to $\binom{K/L}{K\gamma/L}$ was here related to the fact that the receivers of each group have identical caches. Subpacketization can generally increase because there needs to be a large set of pairings between the different caches. Here the number of different distinct caches is reduced, and thus the number of such pairings remains smaller.

Practicality and timeliness of result

The scheme consists of the basic implementable ingredients of ZF and low-dimensional coded caching, and it works for all values of $K, L, \gamma, K_T, \gamma_T$. Its simplicity and effectiveness suggest that having extra transmitting antennas (servers) can play an important role in making coded caching even more applicable in practice, especially at a time when subpacketization complexity is the clear major bottleneck of coded caching, and also at a time when multiple antennas and transmitter cooperation are standard ingredients in wireless communications.

Subpacketization scaling and algorithmic simplicity

We can make the following observation

Corollary 3.6. *For $L = K\gamma$, the aforementioned DoF $L + K\gamma$ can be achieved with subpacketization*

$$S_L = \frac{1}{\gamma} = \frac{K}{L}.$$

Proof. The proof is direct from the above. □

Chapter 4

The Real Cost of Adding Antennas, a.k.a. The CSI Bottleneck

One of the biggest advantages of the single-antenna Coded Caching algorithm is its ability to serve a scaling number of users at a time, without any CSIT thus, offering an alternative to the high feedback-costs required by multi-antenna systems. As is known (cf. [44], [57]), such feedback costs are the reason most multi-antenna solutions fail to scale[1].

On the other hand, as we previously saw, trying to complement the coded caching gains with multiplexing gains by adding even one more antenna, would have required, with the state-of-art algorithms, a feedback cost that would scale with the number of users.

Let us recall Example 2.2 (see also Eq. 4.1), where any transmission of the algorithm in [10] (see also [11, 73, 74] requires CSIT from all $K\gamma + L$ benefiting users so as to form the precoders and, subsequently, requires the CSIR cost of $L+K\gamma$, so as to inform all benefiting users of the channel-vector precoder products that the users will use to cache-out interfering messages. In Example 2.2, corresponding to the setting with L antennas, $K = 4$ users equipped with cache of normalized size $\gamma = \frac{1}{2}$ one such vector takes the form

$$\begin{aligned}\mathbf{x} =& \mathbf{h}_4^{\perp}(A_{23} \oplus B_{13} \oplus C_{12}) + \mathbf{h}_3^{\perp}(A_{24} \oplus B_{14} \oplus D_{12}) + \\ &+ \mathbf{h}_2^{\perp}(A_{34} \oplus C_{14} \oplus D_{13}) + \mathbf{h}_1^{\perp}(B_{34} \oplus C_{24} \oplus D_{23})\end{aligned} \tag{4.1}$$

where we can see that the channels of all 4 users need to be known to form the precoding vectors and further all 4 composite channel - precoder products need to be fed-back.

[1] For a detailed view on the feedback requirements of the MISO (multiple input single output) BC the reader is directed in the work of [58] for a Degrees-of-Freedom characterization under perfect CSIT, in [59–62] for a no CSIT analysis, in [63–65] for a treatment of the compound CSIT scenario and in [66] for the exploitation of delayed CSIT knowledge. Further, [67] considers only the patterns of the channel coherence period to be known, the works in [68–71] allowed mixed CSI at the transmitters to be known, while the work in [72] considered alternating CSIT.

Related Work

Motivated by this feedback bottleneck, different works on multi-antenna (multi-transmitter) coded caching have sought to reduce CSI costs, but in all known cases, any subsequent CSI reductions come at the direct cost of substantially reduced DoF. For example, the works in [75,76] consider reduced quality CSIT, but yield a maximum DoF that is bounded close to $K\gamma + 1$, while the works in [77, 78] consider only statistical CSI, but again achieve much lower DoF. Similarly, the work in [79] uses ACK/NACK type CSIT to ameliorate the issue of unequal channel strengths, but again, achieves no multiplexing gains.

Preview of the Results In this chapter we will describe three multiple-antenna algorithms that achieve[2] the maximally known DoF of $D_L(K,\gamma) = L+K\gamma$, requiring a per-transmission CSI cost of L training slots in the uplink and L training slots in the downlink.

These three algorithms constitute a progressive effort that aimed to complement the reduces feedback costs, with a low subpacketization. We will start from Algorithm 4.2 (see [47]), which constitutes the first work to show that the aforementioned feedback costs are achievable, and does this with the subpacketization

$$S_{L,\text{CSI}} = (K\gamma + L)\binom{K}{K\gamma}\binom{K - K\gamma - 1}{L-1}. \tag{4.2}$$

Further, in Algorithm 4.3 we introduce a new multi-antenna Coded Caching scheme which retains the CSI savings of the previous scheme and does so with a subpacketization that approximately matches the subpacketization of the single antenna case i.e.,

$$S_{L,\text{CSAS}} = (L + K\gamma)\binom{K}{K\gamma}. \tag{4.3}$$

Finally, the third algorithm (see Section 4.3 and the work in [80]) are making progress towards combining the low feedback cost of [47] with the low subpacketization that the multiple antennas provide (cf. [56]) and specifically we show how an exponentially smaller subpacketization, compared to the single stream case, can be achieved while retaining the low feedback costs of previous works. Specifically, the required subpacketization takes the form

$$S_{L,\text{JCS}} = L_c\binom{\frac{K}{L_c}}{K\gamma}, \quad \text{where } L_c = \frac{L + K\gamma}{1 + K\gamma}. \tag{4.4}$$

Note 4.1. *We can see that all the above algorithms can readily be applied in the fully connected multiple transmitter Interference Channel setting (K_T transmitters, each equipped with a cache of normalized size γ_T). In this case, instead of files being combined into XORs locally at the transmitter, they are instead linearly combined in the air (see also Note 3.2 and Section 3.2).*

[2] We need to note that the third algorithm achieves the marginally smaller DoF of $(L + K\gamma)(1 - \gamma)$.

4.1 Coded Caching Gains with Low CSIT

To address the CSI bottleneck, the algorithm designed in [47] is able to achieve the maximally known DoF of $D_L(K,\gamma) = L + K\gamma$ by requiring the much lower CSI cost of L training slots for the uplink and L training slots for the downlink.

Further, as we will demonstrate in Section 9.2, the state-of-art scheme [10] not only requires much more CSI in each transmission slot, but at the same time reuses less of this CSI, thus showing how these feedback costs aggregate with every time-slot, thus consuming a big part of the coherence time, even when this coherence time is long.

We proceed with the main result of the work in [47].

Theorem 4.1. *In the L-antenna MISO-BC with K single-antenna users each equipped with cache of normalized size γ, the DoF $D_L(K,\gamma) = L + K\gamma$ can be achieved with CSIT from only L users at a time, and thus with CSIT cost of L uplink training time-slots, and global CSIR cost of L downlink training time-slots and required subpacketization of*

$$S_{L,CSI} = (K\gamma + L)\binom{K}{K\gamma}\binom{K - K\gamma - 1}{L-1}. \tag{4.5}$$

Intuition and small example Before fully describing the scheme, we proceed with some intuition on the design.

We first note that for the cache placement, the partition of files into subfiles and the storing of subfiles in the users' caches, will draw directly from the placement algorithm of [1].

On the other hand, the XOR generation method will be fundamentally different. The first step is to construct XORs composed of $\frac{K\gamma}{L} + 1$ subfiles, and to then have each transmission communicate L such XORs, thus allowing each transmission to communicate $L + K\gamma$ different subfiles aimed at simultaneously serving a set of $L + K\gamma$ users. Each such set of $L + K\gamma$ served ("active") users will be divided into two sets; the first set λ will consist of the L users that will be assisted by precoding, while the second set π will have $K\gamma$ users who will not be assisted by precoding and who must thus compensate with their caches. Finally, the information vector that was described above (comprized of the L XORs) will be multiplied by the normalized inverse $\mathcal{H}_\lambda^{-1}$ of the channel matrix between the base station and the users in set λ.

What we will see is that the design will guarantee that, during the decoding process, each of the users in λ will only receive one of the XORs (the rest will be nulled-out by the precoder), while the remaining $K\gamma$ users (i.e., those in set π) will receive a linear combination of all L XORs. Hence this will mean that the users in λ will have to each cache out $\frac{K\gamma}{L}$ subfiles[3] in

[3]Here we need to point out that the number of subfiles that each user in λ needs to have cached in order to decode its desired subfile is much smaller than in the original scheme of [1], a fact that has been exploited in [81] to show how users without caches can have the full cache-aided DoF in a multiple-antenna environment.

order to decode their desired subfile, while the users in π will have to cache out $K\gamma + L - 1$ subfiles i.e., all but one subfiles.

Next, we will demonstrate a single transmission of our algorithm using the setting of Example 2.2. The goal is to achieve the same performance as before (all 4 users being able to decode their subfiles in each transmission), while using CSIT from only two users at a time, thus requiring no more than $L = 2$ training slots in the uplink and $L = 2$ training slots in the downlink (this example in its entirety can be found in Example 4.3 located in Section 4.1.2).

Example 4.1. *In the same MISO BC setting of Example 2.2, with $L = 2$ transmit antennas, $K = 4$ users, and a user cache of fractional size $\gamma = 1/2$, a transmitted vector from the proposed algorithm, takes the form*[4]

$$\mathbf{x} = \mathbf{h}_2^{\perp}(A_{34} \oplus C_{14}) + \mathbf{h}_1^{\perp}(B_{34} \oplus D_{23}) \tag{4.6}$$

where, as before, files A, B, C and D are requested by users $1, 2, 3$ and 4, respectively, and where A_{ij} represents the part of A that can be found in the caches of users i and j (similarly for B_{ij}, C_{ij} and D_{ij}).

Assuming that user k receives y_k, $k \in \{1, 2, 3, 4\}$, then the message at each user takes the form:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}^T = \begin{bmatrix} \mathbf{h}_1^T(\mathbf{h}_2^{\perp} \cdot A_{34} \oplus C_{14} + \mathbf{h}_1^{\perp} \cdot B_{34} \oplus D_{23}) \\ \mathbf{h}_2^T(\mathbf{h}_2^{\perp} \cdot A_{34} \oplus C_{14} + \mathbf{h}_1^{\perp} \cdot B_{34} \oplus D_{23}) \\ \mathbf{h}_3^T(\mathbf{h}_2^{\perp} \cdot A_{34} \oplus C_{14} + \mathbf{h}_1^{\perp} \cdot B_{34} \oplus D_{23}) \\ \mathbf{h}_4^T(\mathbf{h}_2^{\perp} \cdot A_{34} \oplus C_{14} + \mathbf{h}_1^{\perp} \cdot B_{34} \oplus D_{23}) \end{bmatrix} \tag{4.7}$$

$$= \begin{bmatrix} A_{34} \oplus C_{14} \\ B_{34} \oplus D_{23} \\ \mathbf{h}_3^T(\mathbf{h}_2^{\perp} \cdot A_{34} \oplus C_{14} + \mathbf{h}_1^{\perp} \cdot B_{34} \oplus D_{23}) \\ \mathbf{h}_4^T(\mathbf{h}_2^{\perp} \cdot A_{34} \oplus C_{14} + \mathbf{h}_1^{\perp} \cdot B_{34} \oplus D_{23}) \end{bmatrix} \tag{4.8}$$

where we have suppressed noise for simplicity.

Hence we see that user 1 and user 2 only receive the first and second XOR respectively (due to the design of the precoders), which means that each of these two users can decode their desired subfiles, i.e. A_{34} and B_{34} respectively, by "caching-out" the unwanted subfiles C_{14} and D_{23}, respectively.

On the other hand, looking at the decoding process for users 3 and 4, we see that user 3 must cache-out subfiles A_{34}, B_{34} and D_{23} (which, by design of the placement are already cached at user 3) in order to decode the desired C_{14}, while user 4 must cache-out subfiles A_{34}, B_{34} and C_{14} (which, by design of the placement, are already cached at user 4) to decode the desired subfile D_{23}. In order to achieve this, users 3 and 4 need to employ their cached content and, also, need some CSI knowledge; user 3 needs products $\mathbf{h}_3^T\mathbf{h}_2^{\perp}$ and $\mathbf{h}_3^T\mathbf{h}_1^{\perp}$, while user 4 needs $\mathbf{h}_4^T\mathbf{h}_2^{\perp}$ and $\mathbf{h}_4^T\mathbf{h}_1^{\perp}$.

[4] Here the reader is warned that there is a notational discrepancy between the subfile indices of this example and the formal notation. In this example we have kept the notation as simple as possible in order to more easily provide a basic intuition on the structure of the scheme.

4.1.1 Scheme Description

We proceed to present the scheme's cache-placement, feedback-acquisition and content-delivery phases.

Placement Phase

The placement phase happens without knowledge of the number of transmit antennas, nor assumes any CSI knowledge. It follows the scheme in [1] where each file $W^n, n \in [N]$, is initially split into $\binom{K}{K\gamma}$ subfiles i.e.,

$$W^n \to \{W^n_\tau,\ \tau \subset [K],\ |\tau| = K\gamma\} \tag{4.9}$$

and the placement of content to the users takes the form

$$\mathcal{Z}_k = \{W^n_\tau : \forall \tau \ni k, |\tau| = K\gamma, \forall n \in [N]\}\,. \tag{4.10}$$

CSI Acquisition

This part takes place at the beginning of the coherence period and it involves first an *uplink training phase* and then a *downlink training phase*. In the uplink phase, a user set λ is selected, comprized of L users, who will transmit pilot signals so that the transmitter can estimate their channel coefficients $\mathbf{h}_k,\ \forall k \in \lambda$, and thus construct precoders $\mathbf{h}^\perp_{\lambda\setminus\{k\}},\ \forall k \in \lambda$. Then, during the downlink training phase, the transmitter will broadcast, for each of the L precoders, P vectors[5] as follows

$$\text{diag}\left(\mathbf{h}^\perp_{\lambda\setminus\{k\}}\right)\mathbf{S}_{L\times P},\quad \forall k \in \lambda$$

which, when multiplied by a user's individual channel, will allow for estimation at any user $q \in \pi \subset [K]\setminus\lambda$ (recall $|\pi| = K\gamma$), the needed global-CSIR products $\mathbf{h}_q^T \cdot \mathbf{h}^\perp_{\lambda\setminus\{k\}},\ \forall k \in \lambda$ and where the operation diag($\mathbf{h}$) creates a square diagonal matrix whose elements are the entries of vector $\mathbf{h}$.

In summary: there are L slots for CSIT because only the L users in set λ need to send CSIT, and there are L slots for global CSIR because, for each fixed precoder, one (training symbol) shot suffices to communicate the composite channel-precoder product to any number of users. The above process in the form of a pseudo-algorithm can be found in Algorithm 4.1.

An extra required Subpacketization Upon notification of the requests $\{W^{d_k},\ k \in [K]\}$ and after the number of antennas is revealed to be L, each requested subfile $W^{d_k}_\tau$ is further split twice as follows[6]

$$W^{d_k}_\tau \to \{W^{d_k}_{\sigma,\tau},\ \sigma \subseteq [K]\setminus(\tau\cup\{k\}), |\sigma| = L-1\} \tag{4.11}$$

$$W^{d_k}_{\sigma,\tau} \to \{W^{\phi,d_k}_{\sigma,\tau},\ \phi \in [L + K\gamma]\}. \tag{4.12}$$

[5] P is simply the minimum number of vectors that are required for a perfect estimation of the intended CSI.

[6] We note that, for clarity of exposition and to avoid many indices, the index ϕ of Equation 4.12 will henceforth be suppressed, thus any $W^{\phi,d_k}_{\sigma,\tau}$ will be denoted as $W^{d_k}_{\sigma,\tau}$ unless ϕ is explicitly needed.

Uplink Training
Transmitter selects set $\lambda \subset [K]$ of L users.
Users in λ sequentially transmit from set $\mathbf{s}_{up}$ of predetermined pilot symbols.
Transmitter receives pilots and forms $\mathbf{h}_k, \ \forall k \in \lambda$.
Transmitter constructs precoders $\mathbf{h}^{\perp}_{\lambda\backslash\{k\}}, \ \forall k \in \lambda$.
Downlink Training
for $k \in \lambda$ *(Select Precoder)* **do**
  Select $\mathbf{S}_{P\times L}$ (Set of P downlink pilot vectors)
  for $p \in P$ **do**
    Transmit:

$$\mathbf{x}_k(p) = \begin{bmatrix} \mathbf{h}^{\perp}_{\lambda\backslash\{k\}}(1)\mathbf{S}(1,p) \\ \vdots \\ \mathbf{h}^{\perp}_{\lambda\backslash\{k\}}(L)\mathbf{S}(L,p) \end{bmatrix}.$$

  end
end

Algorithm 4.1: Uplink and downlink feedback training phases

Delivery phase details

In the following we describe how, for every transmission, the transmitter will first create a vector of L XORs, and will then precode these with the appropriate set of precoders.

Individual XOR design For any two disjoint sets

$$\mu \subset [K], \nu \subset [K], \ \mu \cap \nu = \emptyset$$

where $|\mu| = \frac{K\gamma}{L}+1, \ |\nu| = K\gamma\frac{L-1}{L}$, and for an arbitrary $\sigma \subseteq \Big([K]\backslash(\mu\cup\nu)\Big), \ |\sigma| = L-1$, we construct the XOR

$$X^{\nu,\sigma}_{\mu} = \bigoplus_{k\in\mu} W^{d_k}_{\sigma,(\nu\cup\mu)\backslash\{k\}} \tag{4.13}$$

which consists of $\frac{K\gamma}{L}+1$ subfiles, where

- each subfile is requested by one user that belongs to the user set μ, and where
- all subfiles are known by all users in the user set ν.

The set $(\nu\cup\mu)\setminus\{k\}, \ k\in\mu$ plays the role of τ from the placement phase, as it describes the set of users that have this subfile in their cache, while set σ is a selected subset of $L-1$ users from the set λ.

Example 4.2. *Let us consider the $L = 2$ MISO BC with cumulative cache of size $K\gamma = 4$. Let $\mu = \{1,2,3\}, \nu = \{4,5\}$ and consider some arbitrary $\sigma \in [K] \setminus \{1,2,3,4,5\}$, $|\sigma| = 1$. Then, the designed XOR takes the form*

$$X^{45,\sigma}_{\{123\}} = W^{d_1}_{\{\sigma,\underbrace{2345}_{\tau}\}} \oplus W^{d_2}_{\{\sigma,1345\}} \oplus W^{d_3}_{\{\sigma,1245\}} \tag{4.14}$$

and it delivers the subfiles requested by the users in set μ, while each element of the XOR is cached at all users of set ν. Users $1,2$ and 3 work in the traditional way to cache out each others' subfiles in order to get their own (e.g. user 1 caches out $W^{d_2}_{\{\sigma,1345\}} \oplus W^{d_3}_{\{\sigma,1245\}}$ to get its own $W^{d_1}_{\{\sigma,2345\}}$), while users 4 and 5 are fully protected (since they have cached all 3 subfiles) against this entire undesired XOR. As a quick verification, we see that each index τ (which indicates the set of users that have cached the specific subfile) has size $|\tau| = K\gamma = 4$ which adheres to the available cache-size constraint, which tells us that each file can be stored with redundancy $K\gamma = 4$.

Design of vector of XORs

for *all* $\lambda \subset [K], |\lambda| = L$ *(Select precoded users)* **do**
  Create $\mathcal{H}_\lambda^{-1}$
  for *all* $\pi \subset ([K] \setminus \lambda),\ |\pi| = K\gamma$ *(Select non-Precoded-aided users)* **do**
    Break π into some $\mathcal{F}_i\ i \in [L]:\ |\mathcal{F}_i| = \frac{K\gamma}{L}$, $\bigcup_{i\in[L]} \mathcal{F}_i = \pi,\ \mathcal{F}_i \cap \mathcal{F}_j = \emptyset, \forall i,j \in [L]$
    for $s \in \{0,1,...,L-1\}$ **do**
      $r_i = ((s+i-1) \mod L) + 1, i \in [L]$
      Transmit

$$\mathbf{x}^s_{\lambda,\pi} = \mathcal{H}_\lambda^{-1} \cdot \begin{bmatrix} X^{\pi\setminus\mathcal{F}_{r_1},\lambda\setminus\lambda(1)}_{\lambda(1)\cup\mathcal{F}_{r_1}} \\ X^{\pi\setminus\mathcal{F}_{r_2},\lambda\setminus\lambda(2)}_{\lambda(2)\cup\mathcal{F}_{r_2}} \\ \vdots \\ X^{\pi\setminus\mathcal{F}_{r_L},\lambda\setminus\lambda(L)}_{\lambda(L)\cup\mathcal{F}_{r_L}} \end{bmatrix}. \tag{4.15}$$

    end
  end
end

Algorithm 4.2: Delivery Phase of Low CSI Scheme

At this point we describe how the L XORs of a transmitted vector are chosen. As explained above, the aim of every transmission is to serve different subfiles to $L + K\gamma$ users (there is no data repetition), while requiring CSIT

from only L users. This is described by the following sequence of steps in Algorithm 4.2. Up to now, we have seen that:

- In Step 1, a set λ of L users is chosen, which set signifies the precoding-assisted users.
- In Step 2, a (ZF-type) precoder $\mathcal{H}_{\lambda}^{-1}$ is designed to spatially separate the L users in set λ.
- In Step 3, another set $\pi \subseteq [K] \setminus \lambda$ of $K\gamma$ users is selected from the remaining users.

To construct the L XORs and to properly place them in the vector, the following steps take place.

- In Step 4, the set π of $K\gamma$ users is partitioned, arbitrarily, into L non-overlapping sets $\mathcal{F}_i,\ i \in [L]$, each having $\frac{K\gamma}{L}$ users.
- In Step 5 all users from set λ are associated to a distinct set $\mathcal{F}_i$, as a function of parameter s that takes values from $\{0, 1, ..., L-1\}$. For example, when $s = 0$, the first XOR of the vector will be intended for users in set $\lambda(1) \cup \mathcal{F}_1$ (while completely known by all users in $\pi \setminus \mathcal{F}_1$), the second XOR will be intended for the users in the set $\lambda(2) \cup \mathcal{F}_2$ (while completely known by all users in $\pi \setminus \mathcal{F}_2$) and so on.

  Further, when $s = 1$ the first XOR is intended for users in $\lambda(1) \cup \mathcal{F}_2$ (while completely known by all users in $\pi \setminus \mathcal{F}_2$), the second XOR is for users in $\lambda(2) \cup \mathcal{F}_3$ (while completely known by all users in $\pi \setminus \mathcal{F}_3$) and so on. In particular, Step 5 (and the operation in Step 6, as shown in Algorithm 4.2), allow us to iterate over all sets $\mathcal{F}_i$, associating every time a distinct set $\mathcal{F}_i$ to a distinct user from group λ, until all users from set λ have been associated with all sets $\mathcal{F}_i$.
- Then in the last step (Step 7), the vector of the L XORs is transmitted after being multiplied by precoder matrix $\mathcal{H}_{\lambda}^{-1}$.

Decoding at the users

By design of the XORs (cf. (4.13)), the constructed vector guarantees (together with the precoder) that the users in λ can decode the single XOR (due to ZF) that they receive, and from there (due to caching) proceed to decode their own subfile.

Moreover, this XOR design also guarantees that each user in π has cached all subfiles that are found in the entire vector, apart from its desired subfile. Further, the training phase of Section 4.1.1 has provided the users of set π with all the necessary CSI estimates (specifically, it has provided the receivers with all the necessary precoder-channel composite scalars) to perform the decoding of the linear combination of the transmitted vector.

To see the above more clearly, let us look at the signal received and the decoding process at some of the users.

For some user k belonging in set λ, the decoding process is simple. The received message takes the form

$$y_k = \mathbf{h}_k^T \cdot \mathcal{H}_\lambda^{-1} \begin{bmatrix} X_{\lambda(1)\cup\mathcal{F}_{r_1}}^{\pi\setminus\mathcal{F}_{r_1},\lambda\setminus\lambda(1)} \\ X_{\lambda(2)\cup\mathcal{F}_{r_2}}^{\pi\setminus\mathcal{F}_{r_2},\lambda\setminus\lambda(2)} \\ \vdots \\ X_{\lambda(L)\cup\mathcal{F}_{r_L}}^{\pi\setminus\mathcal{F}_{r_L},\lambda\setminus\lambda(L)} \end{bmatrix} = X_{\{k\}\cup\mathcal{F}_{r_k}}^{\pi\setminus\mathcal{F}_{r_k},\lambda\setminus\{k\}}. \tag{4.16}$$

Due to the design of the remaining XOR (see eq. (4.13)), all but one subfiles have been cached by user k, thus the user can decode the desired subfile.

On the other hand, the decoding process at some user in set π also requires access to feedback information. The received message at user $m \in \pi$ takes the form

$$y_m = \mathbf{h}_m^T \cdot \mathcal{H}_\lambda^{-1} \begin{bmatrix} X_{\lambda(1)\cup\mathcal{F}_{r_1}}^{\pi\setminus\mathcal{F}_{r_1},\lambda\setminus\lambda(1)} \\ X_{\lambda(2)\cup\mathcal{F}_{r_2}}^{\pi\setminus\mathcal{F}_{r_2},\lambda\setminus\lambda(2)} \\ \vdots \\ X_{\lambda(L)\cup\mathcal{F}_{r_L}}^{\pi\setminus\mathcal{F}_{r_L},\lambda\setminus\lambda(L)} \end{bmatrix} = \sum_{j=1}^{L} \mathbf{h}_m^T \mathbf{h}_{\lambda\setminus\lambda(j)}^{\perp} X_{\lambda(j)\cup\mathcal{F}_{r_j}}^{\pi\setminus\mathcal{F}_{r_j},\lambda\setminus\lambda(j)}. \tag{4.17}$$

First, we can observe that due to the process described in Algorithm 4.1, user m has estimated all products $\mathbf{h}_m^T \mathbf{h}_{\lambda\setminus\lambda(j)}^{\perp}$, $\forall j \in [L]$ that appear in Eq. (4.17). Then, by taking account of the fact that $\mathcal{F}_{r_i} \cap \mathcal{F}_{r_j} = \emptyset$, $i \neq j$ we can see that user m belongs in one of the $\mathcal{F}_{r_j}$ subsets of π, which means that user m has fully cached the content of all but one XORs (see Eq. (4.13)), thus can remove them from Eq. (4.17). Further, the remaining XOR, due to its structure (cf. Eq. (4.13)), is decodable by user m.

4.1.2 Calculating the DoF performance

Showing that each desired subfile is transmitted exactly once The first task here is to show that, for a given fixed subfile $W_{\sigma,\tau}^{d_k}$, each of the $K\gamma + L$ sub-subfiles (defined by the same fixed set (σ, τ, k) and are differentiated using the $K\gamma + L$ different $\phi \in [K\gamma + L]$ – the notation of which, as you may recall, we suppress), will appear in $K\gamma + L$ different transmissions $\mathbf{x}_{\lambda,\pi}^s$, for some λ, π, s.

For any arbitrary subfile $W_{\sigma,\tau}^{d_k}$, the labeling (σ, τ, k) defines the set of active users $\lambda \cup \pi = \sigma \cup \tau \cup \{k\}$. Let us recall that $\lambda \cap \pi = \emptyset, \sigma \cap \tau = \emptyset$, that $\sigma \subset \lambda$, and that $|\sigma| = L - 1, |\lambda| = L, |\pi| = |\tau| = K\gamma$. For a fixed (σ, τ, k), let us consider the two complementary cases; case i) $k \in \lambda$, and case ii) $k \notin \lambda$.

In case i), since $|\sigma \cup \tau \cup k| = K\gamma + L$ and because $|\sigma \cup \tau| = K\gamma + L - 1$ (which means that $k \notin \sigma \cup \tau$), we can conclude that $\lambda = \sigma \cup \{k\}$. Given also that

$$\pi = (\sigma \cup \tau \cup \{k\}) \setminus \lambda = \tau$$

means that fixing (σ, τ, k), points to a single λ and a single π. For any fixed (λ, π), in our algorithm, Step 5 identifies L specific sub-subfiles which are defined by the same (σ, τ, k), thus can be differentiated by L different $\phi \in [K\gamma+L]$; these L sub-subfiles of $W_{\sigma,\tau}^{d_k}$ will appear in transmissions $\mathbf{x}_{\lambda,\pi}^{s}$, $s = 0, 1, \ldots, L-1$.

In case ii) the fact that $k \notin \lambda$, implies that — for a given fixed (σ, τ, k) (which also defines the set of active users) — there can be $K\gamma$ different sets λ which take the following form

$$\lambda = \sigma \cup \tau(i), \quad i \in [K\gamma].$$

This means that fixing (σ, τ, k) corresponds to $K\gamma$ different possible sets λ. Since for a fixed (σ, τ, k) the union of $\lambda \cup \pi$ is fixed, we can conclude that each fixed (σ, τ, k) is associated to $K\gamma$ different pairs (λ, π).

Now, having chosen a specific pair (λ, π), where we remind that $k \in \pi$, we can see from Step 5 of Algorithm 4.2 that user k can belong in exactly 1 set $\mathcal{F}_{r_i}, i \in [L]$, let that be $\mathcal{F}_{r_j}$, which means that from all L transmissions of Step 5, a sub-subfile belonging to category $W_{\sigma,\tau}^{d_k}$ will be transmitted in exactly one transmission, i.e. the transmission which will have XOR

$$X_{\tau(i)\cup\mathcal{F}_{r_j}}^{\pi\setminus\mathcal{F}_{r_j},\sigma}. \tag{4.18}$$

In total, for all the different (λ, π) sets, subfile $W_{\sigma,\tau}^{d_k}$ will be transmitted $K\gamma + L$ times.

Finally, since we showed that an arbitrary subfile, $W_{\sigma,\tau}^{d_k}$, will be transmitted exactly $K\gamma + L$ times, this implies that all subfiles of interest will be transmitted by spanning through all possible λ, π sets. □

Calculating the DoF performance The resulting DoF can now be easily seen to be $D_L(K, \gamma) = L + K\gamma$ by recalling that each transmission includes $K\gamma + L$ different subfiles, and by recalling that no subfile is ever repeated. A quick verification, accounting for the subpacketization

$$S_{L,\text{CSI}} = \binom{K}{K\gamma}\binom{K - K\gamma - 1}{L - 1}(K\gamma + L)$$

and for the number of iterations in each step, gives that

$$T_{L,\text{CSI}}(K, \gamma) = \frac{\overbrace{\binom{K}{L}}^{Step\ 1}\overbrace{\binom{K-L}{K\gamma}}^{Step\ 3} \cdot \overbrace{L}^{Step\ 5}}{\binom{K}{K\gamma}\binom{K-K\gamma-1}{L-1}(K\gamma+L)} = \frac{K(1-\gamma)}{K\gamma+L} \tag{4.19}$$

which implies the DoF of

$$D_L(K,\gamma) = \frac{K(1-\gamma)}{T_{L,\text{CSI}}(K,\gamma)} = L + K\gamma. \tag{4.20}$$

The following example employs the complete notation $W^{\phi,d_k}_{\sigma,\tau}$ in order to demonstrate the iteration over all subfiles.

Similar to before, we use $A^{(\phi)}_{\sigma,\tau} \triangleq W^{(\phi),d_1}_{\sigma,\tau}$, $B^{(\phi)}_{\sigma,\tau} \triangleq W^{(\phi),d_2}_{\sigma,\tau}$, and so on.

Example 4.3. *Consider a transmitter with $L = 2$ antennas, serving $K = 4$ users whose caches allow for a cumulative caching redundancy $K\gamma = 2$. Each file is split into*

$$S = \overbrace{(K\gamma + L)}^{\phi}\overbrace{\binom{K - K\gamma - 1}{L-1}}^{\sigma}\overbrace{\binom{K}{K\gamma}}^{\tau} = 24$$

sub-subfiles and the following are the $\binom{K}{L}\binom{K-L}{K\gamma}L = 12$ transmissions that will satisfy all the users' requests.

$$\mathbf{x}^{34}_{12,1} = \mathcal{H}^{-1}_{12}\begin{bmatrix} A^{(1)}_{2,34}C^{(1)}_{2,14} \\ B^{(1)}_{1,34}D^{(1)}_{1,23}\end{bmatrix}, \mathbf{x}^{34}_{12,2} = \mathcal{H}^{-1}_{12}\begin{bmatrix} A^{(2)}_{2,34}D^{(1)}_{2,13} \\ B^{(2)}_{1,34}C^{(1)}_{1,24}\end{bmatrix}$$
$$\mathbf{x}^{12}_{34,1} = \mathcal{H}^{-1}_{34}\begin{bmatrix} B^{(1)}_{4,13}C^{(1)}_{4,12} \\ A^{(1)}_{3,24}D^{(1)}_{3,12}\end{bmatrix}, \mathbf{x}^{12}_{34,2} = \mathcal{H}^{-1}_{34}\begin{bmatrix} A^{(1)}_{4,23}C^{(2)}_{4,12} \\ B^{(1)}_{3,14}D^{(2)}_{3,12}\end{bmatrix}$$
$$\mathbf{x}^{13}_{24,1} = \mathcal{H}^{-1}_{24}\begin{bmatrix} A^{(2)}_{4,23}B^{(2)}_{4,13} \\ C^{(2)}_{2,14}D^{(2)}_{2,13}\end{bmatrix}, \mathbf{x}^{13}_{24,2} = \mathcal{H}^{-1}_{24}\begin{bmatrix} B^{(3)}_{4,13}C^{(3)}_{4,12} \\ A^{(3)}_{2,34}D^{(3)}_{2,13}\end{bmatrix}$$

$$\mathbf{x}^{24}_{13,1} = \mathcal{H}^{-1}_{13}\begin{bmatrix} A^{(2)}_{3,24}B^{(2)}_{3,14} \\ C^{(2)}_{1,24}D^{(2)}_{1,23}\end{bmatrix}, \mathbf{x}^{24}_{13,2} = \mathcal{H}^{-1}_{13}\begin{bmatrix} A^{(3)}_{3,24}D^{(2)}_{3,12} \\ B^{(3)}_{1,34}C^{(3)}_{1,24}\end{bmatrix}$$
$$\mathbf{x}^{23}_{14,1} = \mathcal{H}^{-1}_{14}\begin{bmatrix} A^{(3)}_{4,23}B^{(4)}_{4,13} \\ D^{(3)}_{1,23}C^{(4)}_{1,24}\end{bmatrix}, \mathbf{x}^{23}_{14,2} = \mathcal{H}^{-1}_{14}\begin{bmatrix} A^{(4)}_{4,23}C^{(4)}_{4,12} \\ B^{(4)}_{1,34}D^{(4)}_{1,23}\end{bmatrix}$$
$$\mathbf{x}^{14}_{23,1} = \mathcal{H}^{-1}_{23}\begin{bmatrix} A^{(4)}_{3,24}B^{(3)}_{3,14} \\ C^{(3)}_{2,14}D^{(4)}_{2,13}\end{bmatrix}, \mathbf{x}^{14}_{23,2} = \mathcal{H}^{-1}_{23}\begin{bmatrix} B^{(4)}_{3,14}D^{(4)}_{3,12} \\ C^{(4)}_{2,14}A^{(4)}_{2,34}\end{bmatrix}.$$

As we see, the delay is $T_{2,CSI}\left(4,\frac{1}{2}\right) = \frac{12}{24} = \frac{1}{2}$ and the DoF are

$$D_{2,CSI}\left(4,\frac{1}{2}\right) = \frac{K(1-\gamma)}{T_{2,CSI}\left(4,\frac{1}{2}\right)} = 4. \tag{4.21}$$

Example 4.4. *We consider the $L = 2$ MISO BC with $K = 6$ users and $\gamma = 2/3$*

($K\gamma = 4$). Then, the required 30 *transmissions are*

$$\mathbf{x}_{12,1}^{3456} = \mathcal{H}_{12}^{-1}\begin{bmatrix} A_{2,3456}^{(1)} C_{2,1456}^{(1)} D_{2,1356}^{(1)} \\ B_{1,3456}^{(1)} E_{1,2346}^{(1)} F_{1,2345}^{(1)} \end{bmatrix},\ \mathbf{x}_{12,2}^{3456} = \mathcal{H}_{12}^{-1}\begin{bmatrix} A_{2,3456}^{(2)} E_{2,1346}^{(1)} F_{2,1345}^{(1)} \\ B_{1,3456}^{(2)} C_{1,2456}^{(1)} D_{1,2356}^{(1)} \end{bmatrix}$$

$$\mathbf{x}_{13,1}^{2456} = \mathcal{H}_{13}^{-1}\begin{bmatrix} A_{3,2456}^{(1)} B_{3,1456}^{(1)} D_{3,1256}^{(1)} \\ C_{1,2456}^{(2)} E_{1,2346}^{(2)} F_{1,2345}^{(2)} \end{bmatrix},\ \mathbf{x}_{13,2}^{2456} = \mathcal{H}_{13}^{-1}\begin{bmatrix} A_{3,2456}^{(2)} E_{3,1246}^{(1)} F_{3,1245}^{(1)} \\ C_{1,2456}^{(3)} B_{1,3456}^{(3)} D_{1,2356}^{(2)} \end{bmatrix}$$

$$\mathbf{x}_{14,1}^{2356} = \mathcal{H}_{14}^{-1}\begin{bmatrix} A_{4,2356}^{(1)} B_{4,1356}^{(1)} C_{4,1256}^{(1)} \\ D_{1,2356}^{(3)} E_{1,2346}^{(3)} F_{1,2345}^{(3)} \end{bmatrix},\ \mathbf{x}_{14,2}^{2356} = \mathcal{H}_{14}^{-1}\begin{bmatrix} A_{4,2356}^{(2)} E_{4,1236}^{(1)} F_{4,1235}^{(1)} \\ D_{1,2356}^{(4)} B_{1,3456}^{(4)} C_{1,2456}^{(4)} \end{bmatrix}$$

$$\mathbf{x}_{15,1}^{2346} = \mathcal{H}_{15}^{-1}\begin{bmatrix} A_{5,2346}^{(1)} B_{5,1346}^{(1)} C_{5,1246}^{(1)} \\ E_{1,2346}^{(4)} D_{1,2356}^{(5)} F_{1,2345}^{(4)} \end{bmatrix},\ \mathbf{x}_{15,2}^{2346} = \mathcal{H}_{15}^{-1}\begin{bmatrix} A_{5,2346}^{(2)} D_{5,1236}^{(1)} F_{5,1234}^{(1)} \\ E_{1,2346}^{(5)} B_{1,3456}^{(5)} C_{1,2456}^{(5)} \end{bmatrix}$$

$$\mathbf{x}_{16,1}^{2345} = \mathcal{H}_{16}^{-1}\begin{bmatrix} A_{6,2345}^{(1)} B_{6,1345}^{(1)} C_{6,1245}^{(1)} \\ F_{1,2345}^{(5)} D_{1,2356}^{(6)} E_{1,2346}^{(6)} \end{bmatrix},\ \mathbf{x}_{16,2}^{2345} = \mathcal{H}_{16}^{-1}\begin{bmatrix} A_{6,2345}^{(2)} D_{6,1235}^{(1)} E_{6,1234}^{(1)} \\ F_{1,2345}^{(6)} B_{1,3456}^{(6)} C_{1,2456}^{(6)} \end{bmatrix}$$

$$\mathbf{x}_{23,1}^{1456} = \mathcal{H}_{23}^{-1}\begin{bmatrix} B_{3,1456}^{(2)} A_{3,2456}^{(3)} D_{3,1256}^{(2)} \\ C_{2,1456}^{(2)} E_{2,1346}^{(2)} F_{2,1345}^{(2)} \end{bmatrix},\ \mathbf{x}_{23,2}^{1456} = \mathcal{H}_{23}^{-1}\begin{bmatrix} B_{3,1456}^{(3)} E_{3,1246}^{(2)} F_{3,1245}^{(2)} \\ C_{2,1456}^{(3)} A_{2,3456}^{(3)} D_{2,1356}^{(2)} \end{bmatrix}$$

$$\mathbf{x}_{24,1}^{1356} = \mathcal{H}_{24}^{-1}\begin{bmatrix} B_{4,1356}^{(2)} A_{4,2356}^{(3)} C_{4,1256}^{(2)} \\ D_{2,1356}^{(3)} E_{2,1346}^{(3)} F_{2,1345}^{(3)} \end{bmatrix},\ \mathbf{x}_{24,2}^{1356} = \mathcal{H}_{24}^{-1}\begin{bmatrix} B_{4,1356}^{(3)} E_{4,1236}^{(2)} F_{4,1235}^{(2)} \\ D_{2,1356}^{(4)} A_{2,3456}^{(4)} C_{2,1456}^{(4)} \end{bmatrix}$$

$$\mathbf{x}_{25,1}^{1346} = \mathcal{H}_{25}^{-1}\begin{bmatrix} B_{5,1346}^{(2)} A_{5,2346}^{(3)} C_{5,1246}^{(2)} \\ E_{2,1346}^{(4)} D_{2,1356}^{(5)} F_{2,1345}^{(4)} \end{bmatrix},\ \mathbf{x}_{25,2}^{1346} = \mathcal{H}_{25}^{-1}\begin{bmatrix} B_{5,1346}^{(3)} D_{5,1236}^{(2)} F_{5,1234}^{(2)} \\ E_{2,1346}^{(5)} A_{2,3456}^{(5)} C_{2,1456}^{(5)} \end{bmatrix}$$

$$\mathbf{x}_{26,1}^{1345} = \mathcal{H}_{26}^{-1}\begin{bmatrix} B_{6,1345}^{(2)} A_{6,2345}^{(3)} C_{6,1245}^{(2)} \\ F_{2,1345}^{(5)} D_{2,1356}^{(6)} E_{2,1346}^{(6)} \end{bmatrix},\ \mathbf{x}_{26,2}^{1345} = \mathcal{H}_{26}^{-1}\begin{bmatrix} B_{6,1345}^{(3)} D_{6,1235}^{(2)} E_{6,1234}^{(2)} \\ F_{2,1345}^{(6)} A_{2,3456}^{(6)} C_{2,1456}^{(6)} \end{bmatrix}$$

$$\mathbf{x}_{34,1}^{1256} = \mathcal{H}_{34}^{-1}\begin{bmatrix} C_{4,1256}^{(3)} A_{4,2356}^{(4)} B_{4,1356}^{(4)} \\ D_{3,1256}^{(3)} E_{3,1246}^{(3)} F_{3,1245}^{(3)} \end{bmatrix},\ \mathbf{x}_{34,2}^{1256} = \mathcal{H}_{34}^{-1}\begin{bmatrix} C_{4,1256}^{(4)} E_{4,1236}^{(3)} F_{4,1235}^{(3)} \\ D_{3,1256}^{(4)} A_{3,2456}^{(4)} B_{3,1456}^{(4)} \end{bmatrix}$$

$$\mathbf{x}_{35,1}^{1246} = \mathcal{H}_{35}^{-1}\begin{bmatrix} C_{5,1246}^{(3)} A_{5,2346}^{(4)} B_{5,1346}^{(4)} \\ E_{3,1246}^{(4)} D_{3,1256}^{(5)} F_{3,1245}^{(4)} \end{bmatrix},\ \mathbf{x}_{35,2}^{1246} = \mathcal{H}_{35}^{-1}\begin{bmatrix} C_{5,1246}^{(4)} D_{5,1236}^{(3)} F_{5,1234}^{(3)} \\ E_{3,1246}^{(5)} A_{3,2456}^{(5)} B_{3,1456}^{(5)} \end{bmatrix}$$

$$\mathbf{x}_{36,1}^{1245} = \mathcal{H}_{36}^{-1}\begin{bmatrix} C_{6,1245}^{(3)} A_{6,2345}^{(4)} B_{6,1345}^{(4)} \\ F_{3,1245}^{(5)} D_{3,1256}^{(6)} E_{3,1246}^{(6)} \end{bmatrix},\ \mathbf{x}_{36,2}^{1245} = \mathcal{H}_{36}^{-1}\begin{bmatrix} C_{6,1245}^{(4)} D_{6,1235}^{(3)} E_{6,1234}^{(3)} \\ F_{3,1245}^{(6)} A_{3,2456}^{(6)} B_{3,1456}^{(6)} \end{bmatrix}$$

$$\mathbf{x}_{45,1}^{1236} = \mathcal{H}_{45}^{-1}\begin{bmatrix} D_{5,1236}^{(4)} A_{5,2346}^{(5)} B_{5,1346}^{(5)} \\ E_{4,1236}^{(4)} C_{4,1256}^{(5)} F_{4,1235}^{(4)} \end{bmatrix},\ \mathbf{x}_{45,2}^{1236} = \mathcal{H}_{45}^{-1}\begin{bmatrix} D_{5,1236}^{(5)} C_{5,1246}^{(5)} F_{5,1234}^{(4)} \\ E_{4,1236}^{(5)} A_{4,2356}^{(5)} B_{4,1356}^{(5)} \end{bmatrix}$$

$$\mathbf{x}_{46,1}^{1235} = \mathcal{H}_{46}^{-1}\begin{bmatrix} D_{6,1235}^{(4)} A_{6,2345}^{(5)} B_{6,1345}^{(5)} \\ F_{4,1235}^{(5)} C_{4,1256}^{(6)} E_{4,1236}^{(6)} \end{bmatrix},\ \mathbf{x}_{46,2}^{1235} = \mathcal{H}_{46}^{-1}\begin{bmatrix} D_{6,1235}^{(5)} C_{6,1245}^{(5)} E_{6,1234}^{(4)} \\ F_{4,1235}^{(6)} A_{4,2356}^{(6)} B_{4,1356}^{(6)} \end{bmatrix}$$

$$\mathbf{x}_{56,1}^{1234} = \mathcal{H}_{56}^{-1}\begin{bmatrix} E_{6,1234}^{(5)} A_{6,2345}^{(6)} B_{6,1345}^{(6)} \\ F_{5,1234}^{(5)} C_{5,1246}^{(6)} D_{5,1236}^{(6)} \end{bmatrix},\ \mathbf{x}_{56,2}^{1234} = \mathcal{H}_{56}^{-1}\begin{bmatrix} E_{6,1234}^{(6)} C_{6,1245}^{(6)} D_{6,1235}^{(6)} \\ F_{5,1234}^{(6)} A_{5,2346}^{(6)} B_{5,1346}^{(6)} \end{bmatrix}$$

By examining any of the above transmitted vectors we can deduce that each transmission serves a total of 6 *users, while the feedback cost is* 2 *training slots for CSIT and* 2 *training slots for global CSIR.*

4.2 Low CSI with Single-Antenna Subpacketization (CSAS)

In this section we will describe a new cache-aided transmission scheme designed for the L-antenna MISO BC channel with K cache-aided users, where each user is equipped with a cache of normalized size $\gamma \in (0,1)$. As discussed previously, this new algorithm can achieve the DoF of $D_L(K,\gamma) = L + K\gamma$ with L CSI cost per-transmission. At the same time it significantly reduces the subpacketization requirements compared to the algorithm presented in the previous section.

Theorem 4.2. *In the L-antenna MISO BC with K single-antenna users, each equipped with a cache of normalized size γ, the DoF of $D_L(K,\gamma) = L + K\gamma$ is achievable with per-transmission CSI cost $C = L$ and subpacketization of*

$$S_{L,CSAS} = (L + K\gamma)\binom{K}{K\gamma}. \tag{4.22}$$

Intuition on the design The main premise is to transmit in each slot $K\gamma + L$ subfiles using a vector of L messages. We achieve this by creating a vector comprized of L elements which is further multiplied by a ZF precoder. The entries of the vector consist of one XOR, comprized of $K\gamma + 1$ subfiles (created exactly as in the algorithm of [1]), and $L-1$ uncoded subfiles. We continue with the placement and delivery phases.

Placement phase

Initially, each file is subpacketized into $S_1 = \binom{K}{K\gamma}$ subpackets, which are further split into $K\gamma + L$ smaller packets. We will assume that $T_1 \triangleq \frac{K(1-\gamma)}{1+K\gamma}$ is an integer, while extending the scheme to non-integer values requires a little higher subpacketization. Users' caches are filled according to

$$\mathcal{Z}_{k\in[K]} = \big\{W_\tau^{n,\phi} :\ \tau \subset [K], |\tau| = K\gamma, k \in \tau, \quad \forall \phi \in [K\gamma + L], \forall n \in [N]\big\}. \tag{4.23}$$

The purpose of index ϕ is to deliver, every time a "fresh" subfile, making a total of $K\gamma + L$ subfiles for each associated index τ. We will refrain from using this ϕ index in the following algorithm, in order to keep the notation more clear, and we will prove in Corollary 4.1 that each subfile is transmitted exactly $K\gamma + L$ times.

Delivery phase

In each delivery slot, as discussed above, we will create a vector of size $L \times 1$, where one of its entries will be a XOR comprized of $K\gamma + 1$ subfiles, while the remaining $L-1$ entries will be uncoded subfiles. Then, the vector will be

multiplied by a $L \times L$ precoder matrix, which is calculated as the normalized inverse of the channel between the L-antenna transmitter and a subset of the $K\gamma + L$ users, namely one of the users of the XOR and the $L-1$ users that will be the recipients of the uncoded messages. The process is provided in the form of a pseudo-code in Algorithm 4.3 and will be further described in the following paragraph. We remind that $\mathcal{H}_{\lambda}^{-1}$ denotes the normalized inverse of the channel matrix formed between the L antenna transmitter and the users in set λ, while $\beta_{\tau,k} \subseteq [K] \setminus \tau$ is a set of $L-1$ elements, which are selected to be the following elements of element $k \in [K] \setminus \tau$ in the ordered set $[K] \setminus \tau$.

1 **for** *all* $\sigma \subseteq [K], |\sigma| = K\gamma + 1$ *(pick XOR)* **do**
2 **for** *all* $s \in \sigma$ *(pick precoded user)* **do**
3 Set: $\tau = \sigma \setminus \{s\}$
4 Set: $\lambda = \{s\} \cup \beta_{\tau,s}$.
5 Transmit:

$$\mathbf{x}_{s,\tau} = \mathcal{H}_{\lambda}^{-1} \cdot \begin{bmatrix} \bigoplus_{k \in \sigma} W_{\sigma \setminus \{k\}}^{d_k} \\ W_{\tau}^{d_{\beta_{\tau,s}(1)}} \\ \vdots \\ W_{\tau}^{d_{\beta_{\tau,s}(L-1)}} \end{bmatrix} \tag{4.24}$$

6 **end**
7 **end**

Algorithm 4.3: Delivery Phase of CSAS Scheme

Detailed description of Algorithm 4.3 The algorithm begins by selecting set $\sigma \subset [K]$ comprized of $K\gamma + 1$ users. For these users, the algorithm will form a XOR in the same way as does the algorithm of [1].

Further, in Step 2, the algorithm selects one user, s, from the users of set σ, which user will be assisted by precoding. It is easy to see that the subfile index that this user will receive is $\sigma \setminus \{s\} = \tau$.

Moreover, in Step 4 is formed set λ that contains all the precoding-assisted users. Apart from user $s \in \sigma$ that was selected in Step 2, the remaining $L-1$ precoding-assisted users are selected by calculating set $\beta_{\tau,s}$, as sthe $L-1$ consecutive elements of element s in set $[K] \setminus \tau$. For example, if $\sigma = \{1,2,3\}$, $K = 5$ and $L = 2$ and $s = 1$, then $[K] \setminus \tau = \{1,4,5\}$ thus, $\beta_{\tau,s} = \{4\}$, as 4 is the consecutive element of element s. The users of set $\tau \cup \{s\} \cup \beta_{\tau,s}$ form the $L + K\gamma$ users that will receive a subfile in this slot.

For the above selected users, the algorithm creates a $L \times 1$ vector, where one of the elements is a XOR designed for the users in set σ, while the remaining elements correspond to subfiles described by index $\tau = \sigma \setminus \{s\}$ and intended for the users in set $\beta_{\tau,s}$.

Further, the algorithm forms the precoder matrix $\mathcal{H}_{\lambda}^{-1}$ such that it is the normalized inverse of channel matrix between the L-antenna transmitter and the users in $\lambda = \{s\} \cup \beta_{\tau,s}$. Finally, the transmitted vector is created by multiplying the precoder matrix with the vector containing the messages.

Decoding Process We begin with the users of set λ i.e., the users that are "precoding-assisted". Due to the design of the precoder matrix $\mathcal{H}_{\lambda}^{-1}$, we can see that these users will receive either the XORed message (user s) or each of the uncoded messages to the respective user i.e.,

$$y_{k\in\lambda} = \mathbf{h}_k^T \mathbf{x}_{s,\tau} = \begin{cases} \oplus_{m\in\sigma} W_{\sigma\backslash\{m\}}^{d_m}, & k = s \\ W_{\tau}^{d_k}, & \text{else} \end{cases} \tag{4.25}$$

where for simplicity we have suppressed the noise. It is easy to see that users in set $\beta_{\tau,s}$ will be assisted by precoding, thus will only "see" the uncoded subfile that they want. Further, user s will receive XOR $\oplus_{m\in\sigma} W_{\sigma\backslash\{m\}}^{d_m}$ which can proceed to decode using its cached content.

On the other hand, users in set τ will be receiving a linear combination of all L messages, which will proceed to decode using both the CSIT knowledge and their cached subfiles. The received message at some user $k \in \tau$ takes the form

$$y_{k\in\tau} = \mathbf{h}_k^T \mathbf{x}_{s,\tau} \tag{4.26}$$

$$= \mathbf{h}_k^T \mathbf{h}_{\lambda\backslash\{s\}}^{\perp} \bigoplus_{m\in\sigma} W_{\sigma\backslash\{m\}}^{d_m} + \mathbf{h}_k^T \sum_{i=1}^{L-1} \mathbf{h}_{\lambda\backslash\beta_{\tau,s}(i)}^{\perp} W_{\tau}^{d_{\beta_{\tau,s}(i)}}. \tag{4.27}$$

Examining Eq. (4.27), we can see that the subfiles that are included in the summation term have all been cached by all receivers of set τ and as such they can be removed from the equation. What remains is XOR $\oplus_{k\in\sigma} W_{\sigma\backslash\{k\}}^{d_k}$ which, by design, is decodable by all users that belong in set τ.

Corollary 4.1. *In Algorithm 4.3, each requested subfile $W_{\tau}^{d_k}$ is transmitted exactly $K\gamma + L$ times.*

Proof. We begin by showing the $K\gamma+1$ element of the requested. Since each XOR $\oplus_{k\in\sigma} W_{\sigma\backslash\{k\}}^{d_k}$ is transmitted $K\gamma + 1$ times (cf. Step 2 of Alg. 4.3) i.e., by picking every time a different user of the set σ to be precoding-assisted, it follows that every subfile included in the XOR will be transmitted $K\gamma + 1$ times.

Further, we want to prove that for any user $k \in [K]$ and any subfile described by index τ, this subfile will be transmitted exactly $L - 1$ times as part of the uncoded elements. Let us consider user k that needs to receive subfile $W_{\tau}^{d_k}$. First, we can see that this subfile will be transmitted as part of the uncoded elements of the information message only when σ, i.e. the set of users that will be receiving the XOR, is of the form

$$\sigma = \tau \cup \{s\} \tag{4.28}$$

and where $s \in [K] \setminus (\tau \cup \{k\})$, thus s can take any of the $K - K\gamma - 1$ different values.

We can discern two cases, namely $K - K\gamma = L$ and $K - K\gamma > L$. In the first case i.e., $K - K\gamma = L$, it is clear that for every $s \in [K] \setminus \tau \setminus \{k\}$ then $\beta_{\tau,s} = [K] \setminus (\tau \cup \{s\})$, thus k will be included in each transmission, which amounts to $L - 1$ different subfiles.

In the second case, where the size of set $[K] \setminus \tau$ is bigger than L, only a subset of the users will be selected every time in order to form sets $\beta_{\tau,s}, \forall s \in [K] \setminus \tau \setminus \{k\}$. Using $p_k \in \{1, 2, ..., K - K\gamma\}$, $k \in [K] \setminus \tau$ to denote the position of element k in set $[K] \setminus \tau$, we can see that $k \in \beta_{\tau,s}$ iff

$$p_k = (p_s + l) \mod (K - K\gamma) - 1, \ l \in \{2, 3, ..., L\}. \tag{4.29}$$

But the condition of Eq. (4.29) is true for a single value of l when s is fixed, thus it can hold for exactly $L - 1$ different values s. □

4.3 Joint Consideration of CSIT and Subpacketization (JCS)

Examining the two solutions from [47, 56] we can point out that each one provides benefits towards addressing their respective goal, but does so with exacerbating the other bottleneck, i.e. the work in [56] addresses the subpacketization bottleneck but requires a very high feedback cost while work in [47] addresses the CSI bottleneck but requires a high subpacketization cost.

In contrast, our recent work in [80] made progress towards unifying these two extremes, using multiple antennas to help reduce the subpacketization and at the same time requiring a CSIT cost that is untangled from the DoF.

4.3.1 Main Result

Theorem 4.3. *In the L-antenna MISO BC with K single-antenna users, each equipped with a cache of normalized size $\gamma \in (0, 1)$, the DoF*

$$D_L(K, \gamma) = (1 - \gamma)(L + K\gamma) \tag{4.30}$$

is achievable with per-transmission CSI cost $C = L$ and subpacketization of

$$S_{L,JCS} = L_c \binom{\frac{K}{L_c}}{K\gamma}, \quad \text{where } L_c = \frac{L + K\gamma}{1 + K\gamma}. \tag{4.31}$$

Remark 4.1. *We observe that, as the number of antennas increases, the subpacketization reduction – with respect to the single-stream case (cf. Eq. (1.8)) while, naturally the subpacketization savings compared to other multi-antenna coded caching algorithms (see [2, 10, 11, 47]), are even larger – can be very substantial. For example, if $L = K\gamma + 2$, the subpacketization approximately reduces by a (multiplicative) factor of $S_r = 2^{K\gamma}$.*

Furthermore, in the limit of asymptotically large K, and for $L \gg 1$, we see that the multiplicative reduction in subpacketization takes the form

$$\lim_{K\to\infty} S_r = \lim_{K\to\infty} \left(\frac{L+K\gamma}{1+K\gamma}\right)^{K\gamma} = \lim_{K\to\infty} \left(\frac{L+K\gamma}{K\gamma}\right)^{K\gamma}$$
$$= \lim_{K\to\infty} \left(\frac{L/\gamma}{K}+1\right)^{K\gamma} = e^{\frac{L}{\gamma}\gamma} = e^L \tag{4.32}$$

implying that every additional antenna, in addition to increasing the DoF, also reduces subpacketization by a factor of e.

4.3.2 Scheme Description

Scheme Intuition The main idea that allows for these exponential subpacketization reductions, borrows from the result of [81] (see also Chapter 5), where it was shown that the DoF of an L-antenna MISO BC system with K_c cache-aided users ($\gamma_c > 0$) and K_n cache-less[7] ($\gamma_n = 0$) users is equal to $D_L^{\text{het}}(K_c, \gamma_c, K_n, \gamma_n) = L + K_c\gamma_c$, which exactly matches the DoF (and the delay) of an equivalent homogeneous system with $K = K_c + K_n$ users each equipped with a cache of normalized size $\gamma = \frac{K_c\gamma_c}{K_n+K_c}$.

In this work, we exploit this idea of having 'cache-less' users that benefit from full caching gains, in order to achieve a reduction in the problem dimensionality by a factor of L_c. This will be achieved by partitioning each file into L_c parts and then by grouping the users into L_c groups, where each group of K/L_c users will cache content that is exclusively from just one of the L_c parts of the library content. Hence for each such part, the associated group of $\frac{K}{L_c}$ users will be 'forced' to store each file with a large redundancy $K\gamma$, while simultaneously all remaining $K - \frac{K}{L_c}$ users will not cache this part at all. This further means that for each particular part, there is a group of $\frac{K}{L_c}$ cache-aided users, and the rest can be considered to be cache-less.

We proceed with the placement and delivery phases.

Placement Phase

We start by dividing the users into $L_c = \frac{L+K\gamma}{1+K\gamma}$ groups

$$\mathcal{K}_1 = \left\{1, ..., \frac{K}{L_c}\right\}, ..., \mathcal{K}_{L_c} = \left\{(L_c-1)\frac{K}{L_c}+1, ..., K\right\}. \tag{4.33}$$

Further, we split each file $W^n, n \in [N]$ into $L_c = \frac{L+K\gamma}{1+K\gamma}$ parts $\{W_1^n, ..., W_{L_c}^n\}$, and then each part into $\binom{\frac{K}{L_c}}{K\gamma}$ subfiles i.e.,

$$W_p^n \to \left\{W_{p,\tau}^n,\ \tau \subset \mathcal{K}_p,\ |\tau| = K\gamma\right\}, p \in [L_c] \tag{4.34}$$

[7] We note that this DoF can be achieved while $K_n \leq \frac{L-1}{\gamma}$.

hence, a total subpacketization requirement of

$$S_{L,\text{JCS}} = L_c \binom{\frac{K}{L_c}}{K\gamma}. \tag{4.35}$$

Caching at user $k_p \in \mathcal{K}_p,\ p \in [L_c]$ takes the form

$$\mathcal{Z}_{k_p} = \left\{ W^n_{p,\tau} : k_p \in \tau,\ \tau \subset \mathcal{K}_p, |\tau| = K\gamma, \forall n \in [N] \right\}$$

naturally corresponding to a normalized cache size

$$\frac{|\mathcal{Z}_{k_p}|}{S} = \frac{\binom{K/L_c-1}{K\gamma-1}}{L_c \cdot \binom{K/L_c}{K\gamma}} = \frac{1}{L_c}\frac{K\gamma}{\frac{K}{L_c}} = \gamma. \tag{4.36}$$

Delivery Phase

As mentioned above, the delivery algorithm is based on the method of [81] which merges, in the same transmission, cache-aided and cache-less users. The delivery is divided into L_c sub-phases, where in sub-phase $q \in [L_c]$ the objective is to deliver to all users, the part of their requested file corresponding to partition (labeled by) q. Hence, users of group $\mathcal{K}_q$ act as the cache-aided users, while the remaining users, $[K] \setminus \mathcal{K}_q$, act as cache-less users.

We will focus on describing the delivery for one of the L_c data parts (for part $q \in [L_c]$); for the other parts, corresponding to different q, we simply exchange the roles of the cache-aided and the cache-less users. We remind that the goal is to transmit simultaneously to $L + K\gamma$ users[8]. To do so, we create an L dimensional data vector, where one of its entries is the standard XOR (cf. [1])

$$X_\sigma = \bigoplus_{k\in\sigma} W^{d_k}_{\sigma\setminus\{k\}},\ \ \sigma \subseteq \mathcal{K}_q,\ \ |\sigma| = K\gamma+1 \tag{4.37}$$

intended for some $K\gamma + 1$ ('cache-aided') users in $\mathcal{K}_q$, while the remaining $L-1$ entries of the data vector are $L-1$ uncoded subfiles intended for set $G_n \subseteq [K] \setminus \mathcal{K}_q$ of $L-1$ 'cache-less' users, where these $L-1$ uncoded subfiles are carefully picked to have the same index $\tau \subset \sigma$. This data vector is then ZF precoded, and the transmitted vector takes the form

$$\mathbf{x}^\tau_\sigma = \mathcal{H}^{-1}_{\{k_q\}\cup G_n} \begin{bmatrix} X_\sigma, \\ W^{d_{G_n(1)}}_\tau, \\ \vdots \\ W^{d_{G_n(L-1)}}_\tau \end{bmatrix}^T. \tag{4.38}$$

Decoding at the cache-less users directly benefits from the ZF precoder, which delivers one stream to each of the L–1 cache-less users, and one stream

[8] In the majority of the slots, since the achieved DoF is slightly smaller than $L + K\gamma$

(the XOR) to the cache-aided user k_q who will subsequently 'cache-out' the interfering messages from the XOR to get its desired message $W^{d_{k_q}}_{\sigma\setminus\{k_q\}}$.

On the other hand, any other 'cache-aided' receiver $k \in \tau$, will not benefit from precoding and will rather receive maximal interference in the form

$$\begin{aligned} y^{\tau}_{\sigma}(k \in \tau) &= \mathbf{h}_k^T \mathbf{x}^{\tau}_{\sigma} + w_k \\ &= \mathbf{h}_k^T \mathbf{h}^{\perp}_{G_n} X_{\sigma} + \mathbf{h}_k^T \sum_{i \in G_n} \mathbf{h}^{\perp}_{G_n \setminus \{i\} \cup \{k_q\}} W^{d_i}_{\tau} + w_k. \end{aligned}$$

Transmitted messages $W^{d_i}_{\tau}$ intended for the cache-less users have been picked to be completely known at all the cache-aided users in $\tau = \sigma \setminus \{k_q\}$, allowing each user in set τ to cache-out these messages, with the additional use of their acquired CSIR[9]. This leaves each user in $\tau \subset \sigma$ with receiving only X_{σ}, from which they can naturally decode their desired message.

Matching Algorithm

In the previous section we described the data vector that consists of one XOR, X_{σ}, and $L-1$ uncoded subfiles each described by the same index τ. In this section we focus on how the XOR/subfile-index pairs are picked, so that the decoding process can be performed successfully at all participating $L + K\gamma$ users.

Focusing on a specific part of the files, labeled by $q \in [L_c]$, the goal is to successfully communicate all possible XORs $X_{\sigma},\ \sigma \subseteq \mathcal{K}_q,\ |\sigma| = 1+K\gamma$ (for all 'cache-aided' users in $\mathcal{K}_q$) and at the same time to communicate each subfile $W^{d_i}_{\tau},\ \forall i \in [K] \setminus \mathcal{K}_q,\ \forall \tau \subset \mathcal{K}_q,\ |\tau| = K\gamma$ for the remaining users.

We begin by forming a bipartite graph, where nodes on the right-hand-side (RHS) represent each of the XORs, while each node on the left-hand-side (LHS) represents a set of $L-1$ subfiles with the same index τ, but belonging to a different file (each intended for a different user).

The set of all nodes $\mathcal{L}_{p,\tau}$ on the LHS of the graph is

$$\left\{ \mathcal{L}_{p,\tau} :\ p \in \left[\frac{K}{L + K\gamma} \right],\ \tau \subset \mathcal{K}_q,\ |\tau| = K\gamma \right\} \tag{4.39}$$

where p designates a class of $L-1$ cache-less users (there are $\frac{1}{L-1}(K - \frac{K}{L_c}) = \frac{K}{L+K\gamma}$ such classes), while τ designates the subfile index.

On the other hand, the RHS of the graph consists of two types of nodes. The first set of nodes $\mathcal{R}_{\sigma}$ takes the form

$$\{\mathcal{R}_{\sigma} : \sigma \subseteq \mathcal{K}_q, \quad |\sigma| = K\gamma + 1\} \tag{4.40}$$

and each node corresponds to a XOR X_{σ}, while the second set of nodes $\mathcal{R}_{\emptyset,s}$ takes the form

$$\left\{ \mathcal{R}_{\emptyset,s} : s \in \left[\frac{K\gamma}{1 + K\gamma} \binom{K/L_c}{K\gamma} \right] \right\} \tag{4.41}$$

[9]The fact that L uplink and downlink training slots can provide for this (global) CSIR, is easy to see as is discussed in Section 4.1.1.

and each node corresponds to an empty message.

The problem at hand is to match each node $\mathcal{L}_{p,\tau}$ to a single and unique node $\mathcal{R}_\sigma$, which is equivalent to finding a *perfect matching* (cf. [82]). Each class-index pair $\mathcal{L}_{p,\tau}$ can share an edge with node $\mathcal{R}_\sigma$ iff $\tau \subset \sigma$ ($\forall p \in \left[\frac{K}{K\gamma+L}\right]$). Further, $\mathcal{L}_{p,\tau}$ shares an edge with any node $\mathcal{R}_{\emptyset,s}$.

Thus, there are $\frac{K\gamma}{1+K\gamma}\binom{K/L_c}{K\gamma}$ possible edges from any LHS node $\mathcal{L}_{p,\tau}$, and these are the edges to any node of the second type of RHS nodes and to exactly $\frac{K}{L_c} - K\gamma$ nodes of the first type of RHS nodes[10]. Since each node has the same number of edges, then it is guaranteed that there exists a perfect matching (cf. [82]), while this perfect matching can be calculated through the low complexity algorithm of [83].

Calculating a perfect matching indicates which XOR X_σ or which empty message will be transmitted with a class-index pair (p, τ). Thus, we transmit all XORs and their respective subfiles, while for pairs that are matched to an empty set we only transmit uncoded subfiles.

Calculation of Delivery Time Focusing on the dataset partition labeled by q, we can observe that the transmission will end when all 'cache-less users' (i.e., those in set $[K] \setminus \mathcal{K}_q$) receive all their requested subfiles. Using the fact that in each transmission we communicate a desired subfile to a each of the $L-1$ users, implies that the number of transmissions is equal to the number of LHS nodes, which in turn implies a delay of

$$T_{L,\text{JCS}}(K,\gamma) = \frac{L_c \frac{K}{L+K\gamma}\binom{K/L_c}{K\gamma}}{L_c\binom{K/L_c}{K\gamma}} = \frac{K}{L+K\gamma} \tag{4.42}$$

and a DoF performance

$$D_{L,\text{JCS}}(K,\gamma) = \frac{K(1-\gamma)}{T_{L,\text{JCS}}(K,\gamma)} = (L+K\gamma)(1-\gamma). \tag{4.43}$$

Example

We assume a MISO BC setting where the transmitter is equipped with $L = 7$ antennas and $K = 18$ users each equipped with a cache of normalized size $\gamma = \frac{1}{9}$. In this setting the subpacketization required by the algorithm in [1] is $S_1 = \binom{18}{2} = 153$ and, as previously discussed, does not require any CSIT, while this work requires a subpacketization of $S_L = 3\binom{6}{2} = 45$ and a CSIT cost of $C = L = 7$ training slots.

Further, the proposed scheme can achieve the DoF of $D_{L,\text{JCS}}(K,\gamma) = 8$, while it improves upon the work in [1], which achieves DoF $D_1(K,\gamma) = 3$. A comparison over the CSIT cost, subpacketization requirements and achieved DoF between many multi-antenna algorithms as well as the algorithm of [1] is provided in Table 4.1.

[10] This happens because $(p,\tau) \leftrightarrow \sigma$ iff $\tau \subset \sigma$, thus for a specific τ we have $\sigma = \tau \cup \{k\}$: $k \in \mathcal{K}_q \setminus \tau$. This means that a τ can be matched to $|\mathcal{K}_q \setminus \tau| = \frac{K}{L_c} - K\gamma$ nodes.

$K=18, K\gamma=2, L=7$	CSIT Cost per slot	Subpacketization	DoF
MN [1]	0	153	3
MS [10]	9	10^6	9
Subpacketization [56]	9	9	9
Low CSI [47]	7	$7 \cdot 10^6$	9
CSAS (Sec. 4.2)	7	1377	9
JCS Scheme [80]	7	45	8

Table 4.1: Comparison of the DoF, Subpacketization and CSIT requirements of algorithms [1, 10, 47, 56, 80] and algorithm of Section 4.2 for the setting with $K = 18$ users, each with cache of normalized size $\gamma = \frac{1}{18}$ and a base station equipped with $L = 7$ antennas.

We begin with the placement phase, where users are divided into $L_c = \frac{L+K\gamma}{1+K\gamma} = 3$ groups i.e.,

$$\mathcal{K}_1 = \{1, 2, 3, 4, 5, 6\}$$
$$\mathcal{K}_2 = \{7, 8, 9, 10, 11, 12\}$$
$$\mathcal{K}_3 = \{13, 14, 15, 16, 17, 18\}.$$

Further, each subfile is divided into $S_{L,\text{JCS}} = L_c\binom{K/L_c}{K\gamma}$ subfiles

$$W^n \to \left\{W^n_{q,\tau},\ q \in [L_c],\ \tau \subset \mathcal{K}_p,\ |\tau| = K\gamma\right\}. \tag{4.44}$$

Focusing on the first part of the file partition (i.e., focusing on part $q = 1$), for any file $n \in [N]$, the users cache as follows

$$\mathcal{Z}_1 = \{W^n_{1,12}, W^n_{1,13}, W^n_{1,14}, W^n_{1,15}, W^n_{1,16}\}$$
$$\mathcal{Z}_2 = \{W^n_{1,12}, W^n_{1,23}, W^n_{1,24}, W^n_{1,25}, W^n_{1,26}\}$$
$$\mathcal{Z}_3 = \{W^n_{1,13}, W^n_{1,23}, W^n_{1,34}, W^n_{1,35}, W^n_{1,36}\}$$
$$\mathcal{Z}_4 = \{W^n_{1,14}, W^n_{1,24}, W^n_{1,34}, W^n_{1,45}, W^n_{1,46}\}$$
$$\mathcal{Z}_5 = \{W^n_{1,15}, W^n_{1,25}, W^n_{1,35}, W^n_{1,45}, W^n_{1,56}\}$$
$$\mathcal{Z}_6 = \{W^n_{1,16}, W^n_{1,26}, W^n_{1,36}, W^n_{1,46}, W^n_{1,56}\}$$
$$\mathcal{Z}_7 = \ldots = \mathcal{Z}_{16} = \emptyset.$$

In order to determine the transmission pairs, we need to find a perfect matching over the formed bipartite graph. One such possible matching is depicted in Figure 4.1.

Delivery and Decoding As discussed above, a node on the LHS corresponds to a class of $L - 1 = 6$ cache-less users and the subfile each will receive, while nodes on the RHS represent the XORs, each destined to $K\gamma+1$ users. The matching informs us of the set of $K\gamma + L$ users and the subfiles

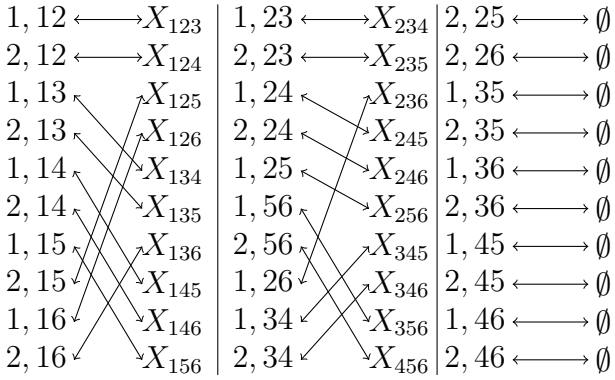

Figure 4.1: The matching of the example of Sec. 4.3.2 split into three subgraphs.

that need to be transmitted. For example, selecting the first XOR-subfile pair, we create the transmission vector

$$\mathbf{x}_{123}^{12} = \mathcal{H}_{\lambda}^{-1} \begin{bmatrix} W_{1,23}^{d_1} \oplus W_{1,13}^{d_2} \oplus W_{1,12}^{d_3} \\ W_{1,12}^{d_7} \\ \vdots \\ W_{1,12}^{d_{12}} \end{bmatrix}$$

where $\lambda = \{3, 7, 8, 9, 10, 11, 12\}$. Decoding at users in λ is direct since $\mathcal{H}_{\lambda}^{-1}$ separates the messages into L parallel streams. From these users, user 3 will also use its cache to extract its desired message from the XOR. The remaining users, users 1 and 2, receive a linear combination of L elements. For example, user 1 will receive

$$y_{123}^{12}(1) = \mathbf{h}_1^T \mathbf{h}_{\lambda\setminus\{3\}}^{\perp} W_{1,23}^{d_1} \oplus W_{1,13}^{d_2} \oplus W_{1,12}^{d_3} + \mathbf{h}_1^T \sum_{k=7}^{12} \mathbf{h}_{\lambda\setminus\{k\}}^{\perp} W_{1,12}^{d_k} + w_1$$

where we can see that the contents of the last summation can all be found in the cache of user 1, who can then remove them, and proceed to again use its cache to extract its desired information from the remaining XOR. A similar process takes place at user 2.

Calculation of Time The bipartite graph of Figure 4.1 (corresponding to part $q = 1$) implies 30 transmissions per part, which means that there will be a total of 90 transmissions, each of normalized duration $\frac{1}{S_{L,\text{JCS}}} = \frac{1}{45}$.

Consequently the overall delivery time takes the form

$$T_{7,JCS}\left(18, \frac{1}{9}\right) = \frac{90}{45} = 2. \tag{4.45}$$

Chapter 5

Uneven Cache sizes and the Multiple-Antenna Benefit

Within the context of coded caching, the work reveals the interesting connection between multiple transmitters and cache-content heterogeneity, effectively showing that multiple antennas can completely remove the penalties typically associated to cache-size unevenness.

First, we show that when both cache-aided and cache-less users need to be served by a single antenna transmitter then the optimal strategy is to treat each group separately, which is proved using Index Coding tools [84].

Further, we turn our focus on the MISO BC or its Degrees-of-Freedom equivalent Interference Channel. First, we identify the performance limits of the extreme case where cache-aided users coincide with users that do not have caches and then we expand the analysis to the case where both user groups are cache-aided but with heterogeneous cache-sizes. In the first case, the main contribution is a new algorithm that employs perfect matchings on a bipartite graph allowing to serve both user types simultaneously. In terms of performance the algorithm can offer, for a wide range of parameters the full multiplexing as well as the full coded-caching gains to both cache-aided as well as cache-less users, while when this is not possible, we show that the performance improvement achieves an L-fold boost.

Finally, in the heterogeneous setting where both user types are cache-enabled, we reveal the surprising finding that adding cumulative normalized cache Γ_2 to the (previously) cache-less users and $L-1$ antennas to the system, increases the Degrees of Freedom by a *multiplicative* factor of $\Gamma_2 + L$.

Related Work

The cache-size unevenness problem has extensively been researched for the single-antenna case. The work in [85] considered heterogeneous caches under the decentralized setting assumption. Further, the work in [86] splits the caches into layers, where the first layer is equal to the smallest cache in the network, the second layer is equal to the difference between the second smallest cache and the smallest cache and so on. Placement is performed

in a layer-by-layer basis, where the each layer is filled using the algorithm of [1] adjusted to the number of users that are involved in this layer and the cache size corresponding to this layer. In [87], the authors formulate the single-stream, heterogeneous cache problem through a set of optimization constraints. Each file is broken into 2^K subfiles and the size of each subfile is determined by solving the underlying optimization problem. Further, in [88] the authors study the device-to-device setting (D2D) where users are equipped with caches of heterogeneous sizes.

In [89] the authors study a two-user setting, where users have different cache sizes and are able to receive information from a common channel and at the same time each user is connected by an individual link, of potentially different capacity, to the main server. For this setting the authors characterize an achievable region. In [90] the authors study the heterogeneous setting under the assumption of users belonging in one of two cache groups, either the higher cache-size group with cache of normalized size γ_1 or the smaller cache-sized group with normalized cache-size γ_2. The authors propose that user caches be divided into two parts, where the first part is of normalized size γ_2 and the placement follows the MN [1] approach using parameters (K, γ_2). For the users with bigger caches, the remaining is filled with content that is designed to reduce the perceived interference from the transmissions. Authors show that their proposed solution achieves a performance gap of at most 1.11 from the proposed method of [87], while at the same time avoiding the complexity of solving the optimization problem.

A main theme of the above works is that cache-size heterogeneity shows a DoF loss compared to its equivalent homogeneous setting.

Setting and brief summary of contributions

In the current setting, we will study the role of multiple antennas in tackling the penalties associated with heterogeneous caches. Specifically, we will first consider a system where a set of K_1 users are equipped with caches, $\gamma_1 > 0$, while the remaining $K_2 = K - K_1$ users are cache-less, $\gamma_2 = 0$. In the single-antenna case we will show that under the assumption of uncoded placement the optimal solution is to treat each group separately, thus showing that a single-stream system is severely penalized by the presence of cache-less users.

Further, we shift the analysis to the study of the multiple antenna case, where we will show that for a wide range of parameters we are able to simultaneously treat both user groups, thus achieving the DoF of the corresponding homogeneous setting i.e.,

$$D_L(K_1, \gamma_1, K - K_1, 0) = K_1\gamma_1 + L \;=\; D_L\left(K, \gamma = \gamma_1 \frac{K_1}{K}\right). \tag{5.1}$$

Moreover, when the DoF of the homogeneous system cannot be achieved, we will show that the DoF performance is L times higher than the single antenna case, which performance we will prove that is exactly optimal under the assumption of uncoded placement.

We will then proceed to explore how adding caches to the second group of users changes the system performance i.e., the $K_2 = K - K_1$ users are now endowed with caches of normalized size $\gamma_2 \in (0, \gamma_1)$. First, we will show that the DoF of

$$D_L(K_1, \gamma_1, K_2, \gamma_2) = L + K_1\gamma_1 + K_2\gamma_2 \tag{5.2}$$

can be achieved for a broad range of parameters, thus showing that multiple-antennas can help reduce the effects of cache-size unevenness. One notable result shows that the same performance boost experienced in the cache-less case when adding $L-1$ antennas (by a multiplicative factor of L) can be also achieved by adding caches to the cache-less group.

Specifically, beginning from the single antenna setting with the cache-aided and cache-less groups, by adding a cumulative cache equal to Γ_2 at the cache-less group and $L-1$ antennas we can achieve a multiplicative DoF boost up to $\Gamma_2 + L$.

We proceed with the theorems and some interesting corollaries.

5.1 Main Results

We begin with the case that involves cache-less users ($\gamma_2 = 0$), and then we will generalize it to the broader case where $\gamma_2 \in (0, \gamma_1)$.

5.1.1 Cache-aided and cache-less users

We start with a result that exemplifies — in the single stream case of $L = 1$ — the problem with having cache-aided users coexisting with cache-less users. We will denote with

$$T_m^{(i)} \triangleq \frac{K_i(1-\gamma_i)}{m + K_i\gamma_i} \tag{5.3}$$

the delay needed to serve K_i cache-aided users with normalized cache size γ_i (in the absence of any cache-less users) using m antennas, where this performance is exactly optimal in the single stream case (under uncoded cache placement [9, 35]), while it is order optimal, with a gap of at most 2 from the optimal in the multi-antenna case (under the assumptions of one-shot and linear schemes) [11].

Theorem 5.1. *In a single-stream BC with K_1 cache-aided users equipped with cache of normalized size γ_1 and with K_2 additional cache-less users, the optimal delay, under the assumption of uncoded placement, takes the form*

$$\begin{aligned} T_1(K_1, \gamma_1, K_2, \gamma_2 = 0) &= T_1^{(1)} + K_2 \\ &= \frac{K_1(1-\gamma_1)}{1 + K_1\gamma_1} + K_2. \end{aligned} \tag{5.4}$$

Proof. The proof is relegated in Section 5.4.1. □

The above reveals that in the single stream case, every time a single cache-less user is added, there is a delay penalty of an entire unit of time, revealing that the two types of users can only be treated separately. If such separation were to be applied in the multi-antenna case, the achievable performance would be

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{K_1(1-\gamma_1)}{L + K_1\gamma_1} + \frac{K_2}{L} \tag{5.5}$$

and the K_2 cache-less users would experience the multiplexing gain of L, but would experience no caching gain.

We proceed with the main result.

Theorem 5.2. *In the MISO BC with $L \geq 1$ antennas, K_1 cache-aided users equipped with cache of fractional size γ_1, and $K_2 \geq (L-1)T_1^{(1)}$ cache-less users, the delivery time*

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = T_1^{(1)} + \frac{K_1 - (L-1)T_1^{(1)}}{\min\{L, K_2\}} \tag{5.6}$$

is achievable and within a factor of 2 from optimal, while if $K_2 \leq (L-1)T_1^{(1)}$ then

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{K_2 + K_1(1-\gamma_1)}{K_1\gamma_1 + L} \tag{5.7}$$

is achievable and within a factor of 3 from optimal under the assumption of linear and one-shot schemes.

Proof. The achievability part of the proof can be found in Section 5.2.1, while the outer bound and gap calculations can be found in Section 5.4.2. □

Furthermore we have the following which accentuates the multiplicative and optimal nature of the gains from adding antennas.

Theorem 5.3. *Starting from the single-antenna BC with K_1 cache-aided users with caches of normalized size γ_1 and $K_2 = (\tilde{L}-1)T_1^{(1)}$ cache-less users ($\gamma_2 = 0$), where $\tilde{L}$ takes any positive value, going from 1 to $L \leq \tilde{L}$ antennas, reduces delay by L times and is the optimal performance under the assumption of uncoded cache placement.*

Proof. The calculation of the performance comes directly from Theorems 5.1 and 5.2, while the proof of optimality of the scheme employs index coding and is shown in Section 5.4.3. □

Let us proceed with a few corollaries that explore some of the ramifications of the above theorem. Eq. (5.5) helps us place the following corollary into context.

Corollary 5.1. *In the L-antenna, $(K_1, \gamma_1, K_2, \gamma_2 = 0)$ MISO BC with $K_2 \leq (L-1)T_1^{(1)}$, all cache-aided and cache-less users can experience full multiplexing gain L as well as full caching gain $K_1\gamma_1$.*

Proof. The proof is direct from Eq. (5.7). □

Example 5.1. *In a setting with $K_1 = 5$ cache-aided users each equipped with a cache of normalized size $\gamma_1 = \frac{1}{5}$ and $K_2 = 2$ cache-less users, the single-antenna, optimal delay under the assumption of uncoded placement is*

$$T_1\left(5, \frac{1}{5}, 2, 0\right) = \frac{K_1(1-\gamma_1)}{K_1\gamma_1 + 1} + K_2 = 4. \tag{5.8}$$

By adding one more antenna, i.e. $L = 2$, the delay becomes

$$T_2\left(5, \frac{1}{5}, 2, 0\right) = \frac{K_1(1-\gamma_1) + K_2}{K_1\gamma_1 + 2} = 2 \tag{5.9}$$

yielding a performance improvement by a factor of 2.

Note 5.1. *We quickly note that the above multiplicative boost of the DoF is in contrast to the additive DoF boost (additive multiplexing gain) experienced in systems with only cache-aided users [10], showing the important role of antennas in systems with cache heterogeneity.*

We proceed with another corollary which can be placed into context, by noting that in a system with L antennas and K_2 *cache-less* users, adding one more antenna would allow (without added delay costs) the addition of only a diminishing number of $\frac{K_2}{L}$ extra cache-less users.

Corollary 5.2. *Starting from the basic single-stream BC with K_1 cache-aided users equipped with caches of normalized size γ_1, then adding an extra $L-1$ transmit antennas, allows for the addition of*

$$K_2 = (L-1)T_1^{(1)} \approx \frac{L-1}{\gamma_1} \tag{5.10}$$

extra cache-less users, at no added delay costs.

Proof. This is direct from Theorem 5.2. □

The following takes another point of view and explores the benefits of injecting cache-aided users into legacy (cache-less) MISO BC systems. To put the following corollary into context, we recall that in a L transmit-antenna MISO BC serving $K_2 \geq L$ cache-less users, the optimal delay is $\frac{K_2}{L}$.

Corollary 5.3. *In a MISO BC with $K_2 \geq L$ cache-less users, introducing K_1 additional cache-aided users with $\gamma_1 \geq \frac{L}{K_2}$, incurs delay*

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) \leq \frac{K_2}{L-1}$$

and thus we can add an infinite number of cache-aided users with a delay increase by a factor that is at most $\frac{L}{L-1}$.

Proof. This is direct from Theorem 5.2. □

Multiple antennas for 'balancing' cache-size unevenness In the variety of works (cf. [85, 86, 91, 92]) that explore the single-stream coded caching setting in the presence of uneven cache sizes, we see that having cache-size asymmetry induces delay penalties and that the preferred cache-size allocation is the uniform one. The following corollary addresses this issue, in the multi-antenna setting.

Corollary 5.4. *The L-antenna, $(K_1, \gamma_1, K_2, \gamma_2 = 0)$ MISO BC with $K_2 \leq (L-1)T_1^{(1)}$ cache-less users, incurs the same achievable delay*

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{K_2 + K_1(1-\gamma_1)}{L + K_1\gamma_1} = \frac{K(1-\gamma_{av})}{L + K\gamma_{av}}$$

as the order optimal homogeneous K-user MISO BC with homogeneous caches of normalized size $\gamma_{av} = \frac{K_1\gamma_1}{K}$ (same cumulative cache $K_1\gamma_1 = K\gamma_{av}$).

Proof. This is direct from Theorem 5.2. □

Example 5.2. *Let us assume the $(K_1 = 5, \gamma_1 = 1/5, K_2 = 2, \gamma_2 = 0)$ MISO BC setting with $L = 2$ antennas. The performance of this setting, as shown in a previous example (Eq. (5.9)) is $T_2 = 2$.*

Furthermore, the performance of the $L = 2$ antenna MISO BC system with $K = 7$ cache-aided users and $\gamma = \gamma_1 \frac{K_1}{K} = \frac{1}{7}$ is (cf. [10, 11]) is

$$T_2\left(7, \frac{1}{7}\right) = \frac{K(1-\gamma)}{L + K\gamma} = 2. \tag{5.11}$$

5.1.2 Coexistence of users with different cache sizes

We now proceed to lift the constraint of cache-less users and consider the more general scenario of $\gamma_2 \in (0, \gamma_1)$.

Theorem 5.4. *In the L-antenna $(K_1, \gamma_1, K_2, \gamma_2 > 0)$ MISO BC, if $T_1^{(1)} \geq T_{L-1}^{(2)}$, then the delivery time matches that of the corresponding homogeneous system with K users of equally sized caches $\gamma = \frac{K_1\gamma_1 + K_2\gamma_2}{K}$, i.e.*

$$T_L(K_1, \gamma_1, K_2, \gamma_2) = \frac{K_1(1-\gamma_1) + K_2(1-\gamma_2)}{L + K_1\gamma_1 + K_2\gamma_2} \tag{5.12}$$

while if $T_1^{(1)} < T_{L-1}^{(2)}$ it takes the form

$$T_L(K_1, \gamma_1, K_2, \gamma_2) = \frac{K_1(1-\gamma_1)}{K_1\gamma_1 + 1} + \frac{K_2(1-\gamma_2) - T_1^{(1)}(L - 1 + K_2\gamma_2)}{\min\{K_2, L + K_2\gamma_2\}}.$$

Proof. The proof is constructive and is detailed in Section 5.3. □

Remark 5.1. *Theorems 5.2 and 5.4 show how adding either one more antennas or enabling with caches the cache-less users provides the same increase in the DoF. Most importantly, by either increasing the number of antennas or*

increasing the caches of the weaker users helps to decrease the heterogeneity of the system and allows to achieve the homogeneous performance. For example, let us assume the L-antenna $(K_1, \gamma_1, K_2 = L \cdot T_1^{(1)}, \gamma_2)$ MISO BC. Then, the performance is given by Eq. (5.6) i.e.,

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = T_1^{(1)} + \frac{T_1^{(1)}}{L} = T_1^{(1)} \frac{L+1}{L}. \tag{5.13}$$

Next, we will show that increasing either the number of antennas by 1 or adding a small cache to each of the cache-less users such that $K_2\gamma_2 = 1$ will result to the same DoF performance. First, increasing the number of antennas to $L+1$, corresponds to the case described by Eq. 5.7 i.e.,

$$T_{L+1}\left(K_1, \gamma_1, K_2, \gamma_2 = 0\right) = T_1^{(1)}. \tag{5.14}$$

Further, by adding a small cache to each of the cache-less users such that $K_2\gamma_2 = 1$ we can easily see that the performance corresponds to Eq. (5.12), thus

$$\begin{aligned} T_L\left(K_1, \gamma_1, K_2, \gamma_2 = \frac{1}{K_2}\right) &= \frac{K_1(1-\gamma_1) + K_2 - 1}{L + K_1\gamma_1 + 1} \\ &= T_1^{(1)} - \frac{1}{L + 1 + K_1\gamma_1} \end{aligned} \tag{5.15}$$

where the last term corresponds to the local caching gain.

What the above says is that beginning from the single-stream case with $K_2 = (\tilde{L} - 1)T_1^{(1)}$ cache-less users, then adding caches to those users and/or increasing the number of antennas up to a total DoF of $\tilde{L}$, will lead to a multiplicative increase of the DoF. This is an outcome of the exploitation of the multiple antennas as a means of spatially separating users and hence treating *in the same transmission* both user types.

Example 5.3. *Let us assume the single antenna system with $K_1 = 7$ cache-aided users equipped with normalized caches of size $\gamma_1 = \frac{1}{7}$ and $K_2 = 10$ cache-less users. First, we will calculate the performance of the above setting and then we will proceed with adding one more antenna, i.e. $L' = 2$ and finally, we will add caches to the cache-less users.*

The first setting's performance is calculated using Eq. (5.4)

$$T_1\left(7, 1/7, 10, 0\right) = \frac{7-1}{2} + 10 = 13. \tag{5.16}$$

Further, the second setting's performance is given by Eq. (5.6)

$$T_2\left(7, 1/7, 10, 0\right) = \frac{7-1}{2} + \frac{7}{2} = \frac{13}{2}. \tag{5.17}$$

Finally, the third setting's performance is given by Eq. (5.13)

$$T_2\left(7, 1/7, 10, 1\right) = \frac{7-1}{2} + \frac{3}{3} = 4. \tag{5.18}$$

From the above we can see that doubling the number of antennas will halve the system delay. Furthermore, if we also equip cache-less users with caches of cumulative size $\Gamma_2 = 1$, while having $L' = 2$ antennas, we can see that the delay is reduced by more than a multiplicative factor of 3, compared to the original setting, which amounts to a nmultiplicative reduction equal to $L' + \Gamma_2$ and a further additive reduction attributed to the local caching gain.

Note 5.2. *It is interesting to note, as will be evidenced in the description of the algorithm and the decoding process at the users, that in the scenario where the cache-aided users co-exist with the cache-less users, that the decoding process at the cache-less users makes use of simple precoding elements, and as such can be implemented by any current mobile device.*

Thus, not only the algorithm is able to provide the increased gains, but at the same time it can be readily implementable in the scenarios described in the introduction.

5.2 Description of the Schemes

We begin with the description of the case where the cache-aided users co-exist with the cache-less users. This scheme will then serve as the basis for the algorithm presented in the case where both user types have positive cache sizes.

In both of these cases, the challenge of the algorithm lies in properly combining the delivery of content towards each of the two types of users, such that subfiles intended to one set are either "cacheable" or will be "nulled-out" via Zero-Force (ZF) precoding.

Notation We remind that for $m \in \mathbb{N}$ we denote

$$T_m^{(i)} \triangleq \frac{K_i(1-\gamma_i)}{m + K_i\gamma_i}, \quad i \in \{1,2\}. \tag{5.19}$$

Further, for sets σ, β we define the two XORs X_σ and $X_{\sigma,\beta}$ as

$$X_\sigma = \bigoplus_{k\in\sigma} W^{d_k}_{\sigma\setminus\{k\}} \tag{5.20}$$

$$X_{\sigma,\beta} = \bigoplus_{k\in\sigma} W^{d_k}_{\beta\cup\sigma\setminus\{k\}}. \tag{5.21}$$

5.2.1 Placement and delivery in the presence of cache-less users

The first scheme that we will present is designed to serve the demands of both cache-aided and cache-less users in the same transmission. The multi-antenna advantage that we will exploit is, compared to the single-stream case, the ability to spatially separate users, allowing unwanted interference towards

cache-less users to be "nulled-out". We build the transmission message by forming a vector comprized of one XOR, intended for some $K_1\gamma_1 + 1$ cache-aided users, and $L-1$ uncoded subfiles, each intended for a cache-less user. The uncoded messages are nulled-out at a specific cache-aided user, as well as its non-intended cache-less users. Further, the XOR is "steered-away" from all the $L-1$ participating cache-less users, thus allowing the cache-less users to successfully decode their intended messages. Moreover, the cache-aided, precoder-assisted user is also able to decode its message, since the precoding process "hides" the unintended, uncoded messages, leaving only the XOR, which naturally can be decoded. Finally, we need to examine the decoding at the remaining $K_1\gamma_1$ cache-aided users. Since there are no other spatial degrees-of-freedom that we can exploit, it follows that these $K_1\gamma_1$ users will receive a linear combination of all L messages. In order for these users to decode their message (which is contained in the XOR) they need to "cache-out" the $L-1$ uncoded messages and since any subfile is cached at most at $K_1\gamma_1$ users, it follows that each uncoded subfile needs to have the same index, which is uniquely defined by the $K_1\gamma_1$ cache-aided users.

In order to communicate the requested files it suffices to transmit for the cache-aided users all XORs, generated as in the algorithm of [1], while for the cache-less users all the subfiles for each of their requested files. The transmit message generation that was described in the previous paragraph reveals that if we transmit a specific XOR, e.g. χ, then the subfiles need to have the same index, i.e. τ, where $\tau \subset \chi$. This design can be viewed as the problem of matching a XOR index χ with a subfile index τ, such that, at the end, each XOR will be matched to a single and unique subfile index (injective).

In other words, the above problem can be viewed as a perfect matching on a bipartite graph, where each node on the left-hand-side (LHS) represents one of the different XORs, while each node one the right-hand-side (RHS) represents a collection of $L-1$ subfiles, intended for some $L-1$ cache-less users, of the same index τ.

We proceed with the description of the placement and delivery phases.

Placement Phase

Initially, each file $W^n,\ n \in [N]$, is divided into

$$S_L(K_1, \gamma_1, K_2, 0) = K_1(1-\gamma_1)\binom{K_1}{K_1\gamma_1} \tag{5.22}$$

subfiles, where subfiles are named according to

$$W^n \to \{W_\tau^{n,\phi},\ \tau \subset \mathcal{K}_1,\ |\tau| = K_1\gamma_1,\ \phi \in \mathcal{K}_1 \setminus \tau\}$$

while cache $\mathcal{Z}_k$ of cache-aided user $k \in \mathcal{K}_1$ is filled according to

$$\mathcal{Z}_k = \{W_\tau^{n,\phi} : k \in \tau, \forall \phi \in \mathcal{K}_1 \setminus \tau,\ \forall n \in [N]\}. \tag{5.23}$$

This is identical to the original placement in [1] (for problem parameters K_1, γ_1), and the extra subpacketization (corresponding to ϕ) will facilitate the aforementioned combinatorial problem of matching XORs with uncoded subfiles.

Delivery Phase

We will first focus on the case of $K_2 = (L-1)T_1^{(1)}$, where the delay

$$T_L(K_1, \gamma_1, K_2, 0) = \frac{K_2 + K_1(1-\gamma_1)}{K_1\gamma_1 + L} \tag{5.24}$$

can be achieved by consistently treating $K_1\gamma_1 + L$ users. The extension to an arbitrary number of cache-less users is based on the above algorithm and will be described later on.

Matching Problem As we have seen, the demands of the cache-aided users are treated by default by sending each XOR X_χ. At the same time, we are able to treat $L-1$ cache-less users, under the condition that their received subfile index τ is the same and that $\tau \subset \chi$.

Thus, the challenge presented in the creation of a transmitted vector is to match a XOR index χ with a subfile index $\tau \subset \chi$ such that, at the end, each χ is matched to a unique τ, in the case where $T_1^{(1)} = 1$ while if $T_1^{(1)} > 1$, then χ needs to be matched to exactly $T_1^{(1)}$ different τ. This constitutes a perfect matching over a bipartite graph, where the left-hand-side (LHS) set of nodes represent the $\binom{K_1}{K_1\gamma_1+1}$ different χ indices, while a node of the right-hand-side (RHS) represents one of the $T_1^{(1)}$ copies of the $\binom{K_1}{K_1\gamma_1}$ different τ intended for some $L-1$ cache-less recipients. A node of the LHS, χ is connected to a node of the RHS τ iff $\tau \subset \chi$.

This type of problem is guaranteed to have a solution when each node from one side, e.g. LHS, is connected to exactly $d \in \mathbb{N}$ nodes of the other side, e.g. RHS (see [82]). In our problem, it is easy to see that each node of the LHS is connected to $d = T_1^{(1)} \cdot (K_1\gamma_1 + 1) = K_1(1-\gamma_1)$ nodes of the RHS.

Since an algorithm that finds such a solution may have high complexity (for example see [93]) we, instead, present an algorithm that requires a slightly higher subpacketization, but can provide an instant solution to the above matching problem. Specifically, the subpacketization of Eq. (5.22) contains the term $K_1(1-\gamma_1)$, thus creating $K_1(1-\gamma_1)$ copies of each XOR X_χ, while the same holds for every subfile τ intended for the cache-less users. Our algorithm achieves a perfect matching by matching node (ϕ, τ), where $\phi \in \mathcal{K}_1 \setminus \tau$, with one of the XORs $X_{\{\phi\}\cup\tau}$ of the LHS.

Transmission The delivery phase, in the form of pseudo-code, is presented in Algorithm 5.1, which we further describe in this paragraph. Transmission commences by splitting the cache-less users into $T_1^{(1)}$ groups with $L-1$ users in each group (Step 2). Then we pick set $\tau \subset \mathcal{K}_1$, $|\tau| = K_1\gamma_1$ (Step 3), which

1 $T_1^{(1)} = \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1}$ (assume $T_1^{(1)} \in \mathbb{N}$).
2 Group cacheless users:

$$g_1 = \{K_1 + 1, K_1 + 2, ..., K_1 + L - 1\}, ...,$$
$$g_{T_1^{(1)}} = \{K_1 + (L-1) \cdot (T_1^{(1)} - 1), ..., K\}.$$

3 **for** *all* $\tau \subset \mathcal{K}_1,\ |\tau| = K_1\gamma_1$ *(pick file index)* **do**
4 | **for** *all* $\phi \in \mathcal{K}_1 \setminus \tau$ *(pick precoded user)* **do**
5 | | Set $\chi = \tau \cup \{\phi\}$
6 | | **for** *all* $t \in \left[T_1^{(1)}\right]$ *(pick cacheless group)* **do**
7 | | | Transmit:

$$\mathbf{x}_{\tau,\phi}^t = \mathcal{H}_{\{\phi\}\cup g_t}^{-1} \begin{bmatrix} X_\chi \\ W_\tau^{d_{g_t(1),\phi}} \\ \vdots \\ W_\tau^{d_{g_t(L-1),\phi}} \end{bmatrix}.$$

8 | | **end**
9 | **end**
10 **end**

Algorithm 5.1: Transmission in the Cache-less Case

set serves two purposes. First, it identifies the cache-aided users that will not be assisted by precoding and second, it identifies the subfile index that the selected cache-less users will receive.

Next, cache-aided user ϕ is picked from the remaining set of cache-less users $\mathcal{K}_1 \setminus \tau$ (Step 4). Further, set g_t containing some $L-1$ cache-less users is picked (Step 6).

The transmitted vector is created by calculating the precoder matrix $\mathcal{H}_{\{\phi\}\cup g_t}^{-1}$ such that it forms the normalized inverse of the channel between the L-antenna transmitter and users of set $\{\phi\} \cup g_t$. The precoder matrix is multiplied by the information vector, which is comprized of XOR X_χ (intended for users $\tau \cup \{\phi\}$) and the $L-1$ subfiles, all sharing index τ (intended for the cache-less users of set g_t) (Step 7).

Decodability In each transmitted vector, we can identify two sets of users, i) those that are assisted by precoding (set $\{\phi\} \cup g_t$) and ii) those that are not (set τ). For the "precoding-aided" set we can immediately recognize that due to the form of the precoder they will receive only their intended message. In the special case of user ϕ, decoding the received XOR will also require the use of its cache.

Users belonging to the second set (set τ) will be receiving a linear combi-

nation of all L messages i.e.,

$$y^t_{\tau,\phi}(k \in \tau) = \mathbf{h}_k^T \mathbf{h}_\tau^\perp X_\chi + \sum_{i=1}^{L-1} \mathbf{h}_k^T \mathbf{h}^\perp_{\lambda \setminus \{g_t(i)\}} W_\tau^{d_{g_t(i)},\phi} + w_k \tag{5.25}$$

where $\lambda = \phi \cup g_t$. We can see that all the subfiles included in the summation are cached at all users of set τ, thus can be removed from the equation. What remains is XOR X_χ, which can be decoded (this is direct from [1]) by all members of set χ.

Transmitting unique subfiles every time At this point the reader may have noticed that the secondary subfile index, associated with the subfile of the cache-aided users, is not identified in Alg. 5.1. This is intentional, since every time we transmit subfile $W_\tau^{d_k}$, we pick a new upper index such that all such indices have been picked. We continue to show that the number of times a subfile is transmitted is exactly $K_1 - K_1\gamma_1$.

Proof. Let us assume we are interested in delivering $W_\mu^{d_k}$ to a user belonging to the set of cache-less users. We can see that the subfile index μ uniquely defines Step 3, while the user's number k defines Step 6. Then the algorithm goes over all possible $\phi \in \mathcal{K}_1 \setminus \mu$ (Step 4), thus at the end subfile $W_\mu^{d_k}$ will be delivered exactly $K_1 - K_1\gamma_1$ times to cache-less user k.

Now we turn our focus to some cache-aided user k and examine how many times this user will receive subfile $W_\mu^{d_k}$. We need to count the number of times the subfile is delivered when user k is assisted by precoding as well as the number of times this same subfile is transmitted when user k is not assisted by precoding.

When user k is assisted by precoding it follows that the remaining cache-aided users are uniquely defined by μ, i.e. $\tau = \mu$, thus there is only one iteration for Steps 3 and 4. Then, Algorithm 5.1 will iterate Step 6 a total of $t = T_1^{(1)}$ times, thus transmitting $W_\mu^{d_k}$ a total of $T_1^{(1)} = \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1}$ times.

Further, when user k is not assisted by precoding, then $k \in \tau$ and the set of active cache-aided users is $\chi = \tau \cup \mu$. Given that the set of cache-aided users is fixed and that $k \in \tau$, it follows that the algorithm will iterate Steps 3 and 4 a total of $K_1\gamma_1$ times, which correspond to each of the remaining cache-aided users $(\chi \setminus \{k\})$ being chosen to be assisted by precoding. Moreover, for each of these iterations Step 6 is repeated $T_1^{(1)}$ times.

In total, the number if times subfile $W_\tau^{d_k}$ is transmitted, when k is a cache-aided user, either assisted by precoding or not, is

$$T_1^{(1)} + K_1\gamma_1 \cdot T_1^{(1)} = K_1(1-\gamma_1) \tag{5.26}$$

which concludes the proof. □

Generalization of scheme and calculation of delay

We have seen that Algorithm 5.1 requires $T_1^{(1)} \in \mathbb{N}$. It is clear that if $(L-1) \cdot T_1^{(1)} \notin \mathbb{N}$, which represents the threshold beyond which we cannot serve

all cache-less users with the maximum DoF, then the number of cache-less users K_2 must be either smaller or higher than $(L-1)\cdot T_1^{(1)}$, both of which will be treated in the following paragraph.

If, on the other hand, $(L-1)\cdot T_1^{(1)} \in \mathbb{N}$, while $T_1^{(1)} \notin \mathbb{N}$, then we can simply increase the subpacketization by a multiplicative factor of $L-1$. This will create a bipartite graph with $(L-1)K_1(1-\gamma_1)\binom{K_1}{K_1\gamma_1+1}$ nodes on the LHS and $(L-1)(K_1\gamma_1+1)\binom{K_1}{K_1\gamma_1}$ nodes on the RHS i.e., both numbers are integers, thus the perfect matching can be achieved.

The other two constraints that we need to address, in order to generalize our algorithm are the cases where $K_2 \gtrless (L-1)T_1^{(1)}$.

First, if $K_2 < (L-1)T_1^{(1)}$ we proceed as in Algorithm 5.1 but when the demands of the cache-less users have been completely satisfied, then we move to treat only the cache-less users (through any multi-antenna algorithm, such as [10,11,47,56]), which we can, naturally, achieve with DoF $D_L(K_1,\gamma_1) = L + K_1\gamma_1$, thus achieving the consistent DoF of $D_L(K_1,\gamma_1,K_2,\gamma_2=0) = L + K_1\gamma_1$ for the whole duration of the transmission.

Finally, for the case of $K_2 > (L-1)T_1^{(1)}$, delivery is split into two sub-phases. During the first sub-phase, we simply employ Algorithm 5.1 on the first $(L-1)T_1^{(1)}$ cache-less users while simultaneously completing the delivery to all K_1 cache-aided users. This is done at a rate of $K_1\gamma_1 + L$ users at a time. Then in the second sub-phase we treat the remaining $K_2 - (L-1)T_1^{(1)}$ cache-less users via ZF-precoding, thus L users at a time. The above sum up to total delay

$$T_L(K_1,\gamma_1,K_2,\gamma_2=0) = T_1^{(1)} + \frac{K_2-(L-1)T_1^{(1)}}{\min\{L, K_1-(L-1)T_1^{(1)}\}}. \tag{5.27}$$

Delay Calculation Following the steps of Algorithm 5.1, corresponding to the case of $K_2 = \frac{K_1(1-\gamma_1)}{K_1\gamma_1+1}$, we have

$$\begin{aligned} T_L\big(K_1,\gamma_1,\frac{K_1(1-\gamma_1)}{K_1\gamma_1+1},0\big) &= \frac{\binom{K_1}{K_1\gamma_1}K_1(1-\gamma_1)\frac{K_1(1-\gamma_1)}{K_1\gamma_1+1}}{K_1(1-\gamma_1)\binom{K_1}{K_1\gamma_1}} \\ &= \frac{K_1(1-\gamma_1)}{K_1\gamma_1+1} = \frac{K_1(1-\gamma_1)+K_2}{K_1\gamma_1+L}. \end{aligned} \tag{5.28}$$

5.2.2 Cache-less users example ($\gamma_2 = 0$)

In this example, we will consider the $L=2$-antenna MISO BC, where $K_1 = 5$, users have caches of normalized size $\gamma_1 = \frac{1}{5}$, while $K_2 = 2$ users have no caches.

First, each file $W^n,\ n \in [N]$ is subpacketized as follows

$$W^n \to \{W^{n,\phi}_\tau,\ \tau \subset \mathcal{K}_1,\ |\tau| = K_1\gamma_1,\ \phi \in \mathcal{K}_1 \setminus \tau\}. \tag{5.29}$$

The caches of the users in set $\mathcal{K}_1$ are filled according to Eq. (5.23), so for example, the cache of the first user contains

$$\mathcal{Z}_1 = \{W_1^{n,2}, W_1^{n,3}, W_1^{n,4}, W_1^{n,5}, \quad \forall n \in [N]\}.$$

Before we describe set of all transmitted vectors that can satisfy the users' demands, we analyze a transmitted vector and its decoding at each user.

Transmission and decoding for specific set of users The goal is to treat $K_1\gamma_1 + L = 3$ users in each time-slot. Let us look in detail one transmitted vector, where we treat cache-aided users 1 and 2 together with cache-less user 6. In this case, we transmit

$$\mathbf{x}_{1,2}^1 = \mathcal{H}_{26}^{-1} \begin{bmatrix} A_2^1 B_1^2 \\ F_1^2 \end{bmatrix}. \tag{5.30}$$

At the reception of a message, user 2 will receive – due to ZF precoding and the design of the precoding matrix $\mathcal{H}_{26}^{-1}$ – only the XORed message $A_2^1 B_1^2$, where it can proceed to cache-out A_2^1 and thus to decode the desired subfile B_1^2. User 6 will receive, again due to precoding, only its respective desired message F_1^2. Finally, user 1 will receive a linear combination of $A_2^1 B_1^2$ and F_1^2, as follows

$$y_1 = \mathbf{h}_1^T \mathbf{h}_6^{\perp} A_2^1 B_1^2 + \mathbf{h}_1^T \mathbf{h}_2^{\perp} F_1^2 + w_1 \tag{5.31}$$

First, by caching out F_1^2, user 1 can decode the XOR, while by XORing out subfile B_1^2 it can decode its desired message.

Sequence of transmissions We now proceed with the entire sequence of the 40 transmissions. Given that each file is subpacketized into $K_1(1-\gamma_1)\binom{K_1}{K_1\gamma_1} = 5(1-\frac{1}{5})\binom{5}{1} = 20$ subpackets, the 40 transmissions will correspond to the desired delay of $T = \frac{K_2 + K_1(1-\gamma_1)}{L + K_1\gamma_1} = 2$. The transmissions are:

$$\mathbf{x}_{1,2}^1 = \mathcal{H}_{26}^{-1} \begin{bmatrix} A_2^1 B_1^2 \\ F_1^2 \end{bmatrix}, \qquad \mathbf{x}_{1,2}^2 = \mathcal{H}_{27}^{-1} \begin{bmatrix} A_2^3 B_1^3 \\ G_1^2 \end{bmatrix}, \mathbf{x}_{1,3}^1 = \mathcal{H}_{36}^{-1} \begin{bmatrix} A_3^1 C_1^3 \\ F_1^3 \end{bmatrix}$$
$$\mathbf{x}_{1,3}^2 = \mathcal{H}_{37}^{-1} \begin{bmatrix} A_3^2 C_1^2 \\ G_1^3 \end{bmatrix}, \qquad \mathbf{x}_{1,4}^1 = \mathcal{H}_{46}^{-1} \begin{bmatrix} A_4^1 D_1^4 \\ F_1^4 \end{bmatrix}, \mathbf{x}_{1,4}^2 = \mathcal{H}_{47}^{-1} \begin{bmatrix} A_4^2 D_1^2 \\ G_1^4 \end{bmatrix}$$
$$\mathbf{x}_{1,5}^1 = \mathcal{H}_{56}^{-1} \begin{bmatrix} A_5^1 E_1^5 \\ F_1^5 \end{bmatrix}, \qquad \mathbf{x}_{1,5}^2 = \mathcal{H}_{57}^{-1} \begin{bmatrix} A_5^2 E_1^2 \\ G_1^5 \end{bmatrix}, \mathbf{x}_{2,1}^1 = \mathcal{H}_{16}^{-1} \begin{bmatrix} A_2^4 B_1^4 \\ F_2^1 \end{bmatrix}$$
$$\mathbf{x}_{2,1}^2 = \mathcal{H}_{17}^{-1} \begin{bmatrix} A_2^5 B_1^5 \\ G_2^1 \end{bmatrix}, \qquad \mathbf{x}_{2,3}^1 = \mathcal{H}_{36}^{-1} \begin{bmatrix} B_3^2 C_2^3 \\ F_2^3 \end{bmatrix}, \mathbf{x}_{2,3}^2 = \mathcal{H}_{37}^{-1} \begin{bmatrix} B_3^1 C_2^1 \\ G_2^3 \end{bmatrix}$$

$$\mathbf{x}_{2,4}^1 = \mathcal{H}_{46}^{-1}\begin{bmatrix} B_4^2 D_2^4 \\ F_2^4 \end{bmatrix}, \quad \mathbf{x}_{2,4}^2 = \mathcal{H}_{47}^{-1}\begin{bmatrix} B_4^1 D_2^1 \\ G_2^4 \end{bmatrix}, \mathbf{x}_{2,5}^1 = \mathcal{H}_{56}^{-1}\begin{bmatrix} B_5^2 E_2^5 \\ F_2^5 \end{bmatrix}$$
$$\mathbf{x}_{2,5}^2 = \mathcal{H}_{57}^{-1}\begin{bmatrix} B_5^1 E_2^1 \\ G_2^5 \end{bmatrix}, \quad \mathbf{x}_{3,1}^1 = \mathcal{H}_{16}^{-1}\begin{bmatrix} A_3^4 C_1^4 \\ F_3^1 \end{bmatrix}, \mathbf{x}_{3,1}^2 = \mathcal{H}_{17}^{-1}\begin{bmatrix} A_3^5 C_1^5 \\ G_3^1 \end{bmatrix}$$
$$\mathbf{x}_{3,2}^1 = \mathcal{H}_{26}^{-1}\begin{bmatrix} B_3^4 C_2^4 \\ F_3^2 \end{bmatrix}, \quad \mathbf{x}_{3,2}^2 = \mathcal{H}_{27}^{-1}\begin{bmatrix} B_3^5 C_2^5 \\ G_3^2 \end{bmatrix}, \mathbf{x}_{3,4}^1 = \mathcal{H}_{46}^{-1}\begin{bmatrix} C_4^3 D_3^4 \\ F_3^4 \end{bmatrix}$$
$$\mathbf{x}_{3,4}^2 = \mathcal{H}_{47}^{-1}\begin{bmatrix} C_4^1 D_3^1 \\ G_3^4 \end{bmatrix}, \quad \mathbf{x}_{3,5}^1 = \mathcal{H}_{56}^{-1}\begin{bmatrix} C_5^3 E_3^5 \\ F_3^5 \end{bmatrix}, \mathbf{x}_{3,5}^2 = \mathcal{H}_{57}^{-1}\begin{bmatrix} C_5^1 E_3^1 \\ G_3^5 \end{bmatrix}$$
$$\mathbf{x}_{4,1}^1 = \mathcal{H}_{16}^{-1}\begin{bmatrix} A_4^3 D_1^3 \\ F_4^1 \end{bmatrix}, \quad \mathbf{x}_{4,1}^2 = \mathcal{H}_{17}^{-1}\begin{bmatrix} A_4^5 D_1^5 \\ G_4^1 \end{bmatrix}, \mathbf{x}_{4,2}^1 = \mathcal{H}_{26}^{-1}\begin{bmatrix} B_4^3 D_2^3 \\ F_4^2 \end{bmatrix}$$
$$\mathbf{x}_{4,2}^2 = \mathcal{H}_{27}^{-1}\begin{bmatrix} B_4^5 D_2^5 \\ G_4^2 \end{bmatrix}, \quad \mathbf{x}_{4,3}^1 = \mathcal{H}_{36}^{-1}\begin{bmatrix} C_4^2 D_3^2 \\ F_4^3 \end{bmatrix}, \mathbf{x}_{4,3}^2 = \mathcal{H}_{37}^{-1}\begin{bmatrix} C_4^5 D_3^5 \\ G_4^3 \end{bmatrix}$$
$$\mathbf{x}_{4,5}^1 = \mathcal{H}_{56}^{-1}\begin{bmatrix} D_5^4 E_4^5 \\ F_4^5 \end{bmatrix}, \quad \mathbf{x}_{4,5}^2 = \mathcal{H}_{57}^{-1}\begin{bmatrix} D_5^1 E_4^1 \\ G_4^5 \end{bmatrix}, \mathbf{x}_{5,1}^1 = \mathcal{H}_{16}^{-1}\begin{bmatrix} A_5^3 E_1^3 \\ F_5^1 \end{bmatrix}$$
$$\mathbf{x}_{5,1}^2 = \mathcal{H}_{17}^{-1}\begin{bmatrix} A_5^4 E_1^4 \\ G_5^1 \end{bmatrix}, \quad \mathbf{x}_{5,2}^1 = \mathcal{H}_{26}^{-1}\begin{bmatrix} B_5^3 E_2^3 \\ F_5^2 \end{bmatrix}, \mathbf{x}_{5,2}^2 = \mathcal{H}_{27}^{-1}\begin{bmatrix} B_5^4 E_2^4 \\ G_5^2 \end{bmatrix}$$
$$\mathbf{x}_{5,3}^1 = \mathcal{H}_{36}^{-1}\begin{bmatrix} C_5^2 E_3^2 \\ F_5^3 \end{bmatrix}, \quad \mathbf{x}_{5,3}^2 = \mathcal{H}_{37}^{-1}\begin{bmatrix} C_5^4 E_3^4 \\ G_5^3 \end{bmatrix}, \mathbf{x}_{5,4}^1 = \mathcal{H}_{46}^{-1}\begin{bmatrix} D_5^2 E_4^2 \\ F_5^4 \end{bmatrix}$$
$$\mathbf{x}_{5,4}^2 = \mathcal{H}_{47}^{-1}\begin{bmatrix} D_5^3 E_4^3 \\ G_5^4 \end{bmatrix}.$$

The 40 slots, each of duration

$$t_s = \left(K_c(1-\gamma)\binom{K_c}{K_c\gamma}\right)^{-1} = \frac{1}{20}, \tag{5.32}$$

imply a delay $T = 2$, which is also the same delay that would be needed in the symmetric case where the $K = 7$ users would have an identical $\gamma_{av} = \frac{1}{7}$ (same cumulative cache $K\gamma_{av} = 1$).

5.3 Two types of cache-aided users

In this section we consider the L-antenna MISO BC setting, where both user types are equipped with caches of heterogeneous capacities i.e., γ_1, γ_2, where $\gamma_2 \in (0, \gamma_1)$. Contrary to the cache-less users case, here, treating the two types simultaneously is indeed possible, even when utilizing one transmit antenna. The elusive goal that is presented in this setting has been to achieve the same performance as the respective homogeneous system i.e, when $\gamma_{av} = \frac{K_1\gamma_1 + K_2\gamma_2}{K}$. What we will show here is that this performance can indeed be achieved in the multi-antenna case, for a wide range of parameters.

First, we will focus on proving the result of Eq. (5.12), where we can see that each transmission serves exactly $L + K_1\gamma_1 + K_2\gamma_2$ users. The main idea is to create an $L \times 1$ information vector, which we will further multiply with an $L \times L$ precoder matrix to form the transmitting vector. The elements of the

created vector belong in one of 4 types. One element corresponds to a XOR of $1+K_1\gamma_1$ subfiles intended for some users of set $\mathcal{K}_1$, while another element corresponds to a XOR of $1+K_2\gamma_2$ subfiles which is intended for some users of set $\mathcal{K}_2$. The remaining $L-2$ elements will carry $L_1-1,\ L_1\in[1,L-1]$ uncoded messages for users of set $\mathcal{K}_1$ and $L_2-1,\ L_2\in[1,L-1]$ uncoded messages for users of set $\mathcal{K}_2$, where $L_1+L_2=L$, and where the exact values of variables L_1 and L_2 are calculated by solving

$$\frac{K_1(1-\gamma_1)}{L_1+K_1\gamma_1}=\frac{K_2(1-\gamma_2)}{L_2+K_2\gamma_2} \tag{5.33}$$

and assuming that $L_1\geq 1$.

In other words, in the above problem we allocate L_1 streams to one set of users and L_2 streams to the other set of users. This observation will allow us to view the problem at hand as a concatenation of two multi-antenna problems. In what follows, we will use the multi-antenna Coded Caching algorithm which we present in detail in Section 4.2. Further, we will assume that L_1 and L_2 are integers, while we relegate the non-integer case in Section 5.5.

Placement Phase

We split each file $W^n,\ n\in[N]$ into

$$S_L(K_1,\gamma_1,K_2,\gamma_2)=(K_1\gamma_1+L_1)\binom{K_1}{K_1\gamma_1}(K_2\gamma_2+L_2)\binom{K_2}{K_2\gamma_2} \tag{5.34}$$

subfiles, i.e. each subfile $W^{n,\phi_1,\phi_2}_{\tau_1,\tau_2}$ is characterized by 4 indices, $\phi_1\in[K_1\gamma_1+L_1]$, $\tau_1\subset\mathcal{K}_1,\ |\tau_1|=K_1\gamma_1$ and $\phi_2\in[K_2\gamma_2+L_2]$, $\tau_2\subset\mathcal{K}_2,\ |\tau_2|=K_2\gamma_2$, where indices τ_1 and τ_2 reveal which users have cached this subfile from sets $\mathcal{K}_1$ and $\mathcal{K}_2$, respectively.

The caches of the users are filled as follows

$$\mathcal{Z}_{k_1\in\mathcal{K}_1}=\{W^{n,\phi_1,\phi_2}_{\tau_1,\tau_2}\quad:k_1\in\tau_1,\forall\tau_2,\phi_1,\phi_2\} \tag{5.35}$$

$$\mathcal{Z}_{k_2\in\mathcal{K}_2}=\{W^{n,\phi_1,\phi_2}_{\tau_1,\tau_2}\quad:k_2\in\tau_2,\forall\tau_1,\phi_1,\phi_2\} \tag{5.36}$$

where it is easy to see that the above placement respects the cache-size constraint of each user.

Delivery Phase

Algorithm 5.2 describes the delivery phase in the form of a pseudo-code. As mentioned above, the algorithm works as a concatenation of two multi-antenna Coded Caching schemes. Specifically, we can see that having selected the XOR for set $\mathcal{K}_1$, along with the precoded user (Steps 1 and 2), then the algorithm goes over all possible combinations of XORs and their respective users corresponding to set $\mathcal{K}_2$. In the case of users of set $\mathcal{K}_1$, this allows to deliver all the index pairs (ϕ_2,τ_2) that correspond to the other set of users.

for *all* $\sigma_1 \subseteq [K_1]$, $|\sigma_1| = K_1\gamma_1 + 1$ **do**
  for *all* $s_1 \in \sigma_1$ **do**
    for *all* $\sigma_2 \subseteq [K_2]$, $|\sigma_2| = K_2\gamma_2 + 1$ **do**
      for *all* $s_1 \in \sigma_2$ **do**
        Set: $\tau_1 = \sigma_1 \setminus \{s_1\}$
        $\tau_2 = \sigma_2 \setminus \{s_2\}$
        $\lambda = \{s_1\} \cup \{s_2\} \cup \beta_{\tau_1,s_1} \cup \beta_{\tau_2,s_2}$.
        Transmit:

$$\mathbf{x}_{s_2,\tau_2}^{s_1,\tau_1} = \mathcal{H}_\lambda^{-1} \cdot \begin{bmatrix} X_{\sigma_1,\beta_{\tau_1,s_1}} \\ W_{\tau_1,\tau_2}^{d_{\beta_{\tau_1,s_1}}(1)} \\ \vdots \\ W_{\tau_1,\tau_2}^{d_{\beta_{\tau_1,s_1}}(L_1-1)} \\ X_{\sigma_2,\beta_{\tau_2,s_2}} \\ W_{\tau_1,\tau_2}^{d_{\beta_{\tau_2,s_2}}(1)} \\ \vdots \\ W_{\tau_1,\tau_2}^{d_{\beta_{\tau_2,s_2}}(L_2-1)} \end{bmatrix}$$

      end
    end
  end
end

Algorithm 5.2: Transmission Process of the multi-antenna Coded Caching setting with users of heterogeneous cache-sizes.

Decoding Process The decoding process is similar to that of Algorithm 4.3. For the users in set λ i.e., the "precoded" users, we can see that they receive only one of the L messages, thus they either decode using a ZF precoder (users in $\lambda \setminus \{s_1\} \setminus \{s_2\}$) or they use a ZF decoder and continue to decode their respective XOR by use of their cached content.

The remaining users will receive a linear combination of all L messages, which they can decode using the acquired CSI and their stored content. For example, user $k \in \tau_1$ will receive

$$\begin{aligned} y_k =& \mathbf{h}_k^T \mathbf{h}_{\lambda\setminus\{s_1\}}^{\perp} X_{\sigma_1,\tau_2} + \mathbf{h}_k^T \sum_{i=1}^{L-1} \mathbf{h}_{\lambda\setminus\beta_{\tau_1,s_1(i)}}^{\perp} W_{\tau_1,\tau_2}^{d_{\beta_{\tau_1,s_1(i)}}} \\ &+ \mathbf{h}_k^T \mathbf{h}_{\lambda\setminus\{s_2\}}^{\perp} X_{\sigma_2,\tau_1} + \mathbf{h}_k^T \sum_{i=1}^{L-1} \mathbf{h}_{\lambda\setminus\beta_{\tau_2,s_2(i)}}^{\perp} W_{\tau_1,\tau_2}^{d_{\beta_{\tau_2,s_2(i)}}}. \end{aligned} \tag{5.37}$$

We can see that terms $2, 3$, and 4 of Eq. (5.37) are completely known to

any receiver of set τ_1, thus can be removed. The remaining term is XOR X_{σ_1,τ_2} which, by design, is decodable by any user belonging in set τ_1.

5.3.1 Extension to the remaining cases

In this section we will prove the result of Eq. (5.13), which corresponds to the case where the number of streams that should be allocated to the group with the higher cache size is less than one. In this case, we simply treat the users of set $\mathcal{K}_1$ using one stream and allocate the remaining $L_2 = L - 1$ streams for the second set of users, $\mathcal{K}_2$.

At some point in the transmission all the files requested by set $\mathcal{K}_1$ have been successfully communicated, while users of set $\mathcal{K}_2$ require more transmissions to completely receive their files. This is because we transmit at a rate of $K_1\gamma_1 + 1$ subfiles to the users of set $\mathcal{K}_1$ and with rate of $L - 1 + K_2\gamma_2$ subfiles to the users of set $\mathcal{K}_2$, while $\frac{K_1(1-\gamma_1)}{1+K_1\gamma_1} < \frac{K_2(1-\gamma_2)}{L-1+K_2\gamma_2}$.

To complete the transmission of files to the second set of users, we employ any multi-antenna cache-aided algorithm (e.g. [10, 56]) and proceed to transmit at a rate of $L + K_2\gamma_2$ subfiles at a time. Thus, the completion time corresponding to the two parts of transmission takes the form

$$T_L(K_1,\gamma_1,K_2,\gamma_2) = \frac{K_1(1-\gamma_1)}{K_1\gamma_1+1} + \frac{K_2(1-\gamma_2) - T_1^{(1)}(L-1+K_2\gamma_2)}{\min\{K_2, L+K_2\gamma_2\}}. \tag{5.38}$$

5.3.2 Two Type Cache-aided Example

In this section we present an example that illustrates the mechanics of the two user case. Specifically, we will focus on the $L = 3$-antenna MISO BC, where $K_1 = 5$ users of set $\mathcal{K}_1$ are equipped with caches of normalized size $\gamma_1 = \frac{2}{5}$, while $K_2 = 4$ users of set $\mathcal{K}_2$ are equipped with caches of normalized size $\gamma_2 = \frac{1}{4}$. For this setting, the number of streams (cf. Eq. (5.33)) should be divided as $L_1 = 1$ and $L_2 = 2$.

We begin by splitting each file into

$$S_3\left(5, \frac{2}{5}, 4, \frac{1}{4}\right) = (K_1\gamma_1 + L_1)(K_1\gamma_2 + L_2)\binom{K_1}{K_1\gamma_1}\binom{K_2}{K_2\gamma_2} = 360$$

subfiles, where subfile $W^{n,\phi_1,\phi_2}_{\tau_1,\tau_2}$ has indices $\tau_1 \subset [5]$, $|\tau_1| = 2$, $\phi_1 \in [3]$, $\tau_2 \subset [4]$, $|\tau_2| = 1$, $\phi_2 \in [3]$.

Placement Phase

This phase is carried out according to Eq. (5.35)-(5.36) where, for example, the caches of users $1 \in \mathcal{K}_1$ and $6 \in \mathcal{K}_2$ are filled as

$$\mathcal{Z}_1 = \{W^{n,\phi_1,\phi_2}_{12,\tau_2}, W^{n,\phi_1,\phi_2}_{13,\tau_2}, W^{n,\phi_1,\phi_2}_{14,\tau_2}, W^{n,\phi_1,\phi_2}_{15,\tau_2}, \forall\tau_2,\phi_1,\phi_2\}$$
$$\mathcal{Z}_6 = \{W^{n,\phi_1,\phi_2}_{\tau_1,6}, \forall\tau_1,\phi_1,\phi_2\}.$$

Delivery Phase

For notational simplicity, we abstain from using indices ϕ_1, ϕ_2. Further, we will only present one iteration of the algorithmic steps 1-2, that delivers the first XOR ($A_{23}B_{13}C_{12}$ intended for users $1,2,3$) of the user set $\mathcal{K}_1$, while it goes through over all of the remaining steps i.e., Steps $3-8$.

$$\mathbf{x}_{1,23}^{6,7} = \mathcal{H}_{168}^{-1}\begin{bmatrix} A_{23,7}B_{13,7}C_{12,7} \\ F_{23,7}G_{23,6} \\ H_{23,7} \end{bmatrix}, \qquad \mathbf{x}_{1,23}^{7,6} = \mathcal{H}_{178}^{-1}\begin{bmatrix} A_{23,6}B_{13,6}C_{12,6} \\ F_{23,7}G_{23,6} \\ H_{23,6} \end{bmatrix}$$

$$\mathbf{x}_{1,23}^{6,8} = \mathcal{H}_{167}^{-1}\begin{bmatrix} A_{23,8}B_{13,8}C_{12,8} \\ F_{23,8}H_{23,6} \\ G_{23,8} \end{bmatrix}, \qquad \mathbf{x}_{1,23}^{8,6} = \mathcal{H}_{187}^{-1}\begin{bmatrix} A_{23,6}B_{13,6}C_{12,6} \\ F_{23,8}H_{23,6} \\ G_{23,6} \end{bmatrix}$$

$$\mathbf{x}_{1,23}^{6,9} = \mathcal{H}_{167}^{-1}\begin{bmatrix} A_{23,9}B_{13,9}C_{12,9} \\ F_{23,9}I_{23,6} \\ G_{23,9} \end{bmatrix}, \qquad \mathbf{x}_{1,23}^{9,6} = \mathcal{H}_{197}^{-1}\begin{bmatrix} A_{23,6}B_{13,6}C_{12,6} \\ F_{23,9}I_{23,6} \\ G_{23,6} \end{bmatrix}$$

$$\mathbf{x}_{1,23}^{7,8} = \mathcal{H}_{179}^{-1}\begin{bmatrix} A_{23,8}B_{13,8}C_{12,8} \\ G_{23,8}H_{23,7} \\ I_{23,8} \end{bmatrix}, \qquad \mathbf{x}_{1,23}^{8,7} = \mathcal{H}_{189}^{-1}\begin{bmatrix} A_{23,7}B_{13,7}C_{12,7} \\ G_{23,8}H_{23,7} \\ I_{23,7} \end{bmatrix}$$

$$\mathbf{x}_{1,23}^{7,9} = \mathcal{H}_{178}^{-1}\begin{bmatrix} A_{23,9}B_{13,9}C_{12,9} \\ G_{23,9}I_{23,7} \\ H_{23,9} \end{bmatrix}, \qquad \mathbf{x}_{1,23}^{9,7} = \mathcal{H}_{196}^{-1}\begin{bmatrix} A_{23,7}B_{13,7}C_{12,7} \\ G_{23,9}I_{23,7} \\ F_{23,7} \end{bmatrix}$$

$$\mathbf{x}_{1,23}^{8,9} = \mathcal{H}_{186}^{-1}\begin{bmatrix} A_{23,9}B_{13,9}C_{12,9} \\ H_{23,9}I_{23,8} \\ F_{23,9} \end{bmatrix}, \qquad \mathbf{x}_{1,23}^{9,8} = \mathcal{H}_{196}^{-1}\begin{bmatrix} A_{23,8}B_{13,8}C_{12,8} \\ H_{23,9}I_{23,8} \\ F_{23,8} \end{bmatrix}.$$

Decoding Process

The decoding process follows the steps of Algorithm 4.3. First, the members of set λ, i.e. the precoded users, will receive one of the L messages, which can decode using their cached content. Further, the users of set $\tau_1 \cup \tau_2$ will receive a linear combination of all L messages, which can decode using the acquired CSIT and their cached content.

As an example, we will look at the decoding of transmitted message $\mathbf{x}_{1,23}^{6,7}$ at any intended user. First, we can see that the precoded users are $1,6,8$ and these users will receive

$$y_{1,23}^{6,7}(k \in \{1,6,8\}) = \mathbf{h}_k^T \mathbf{h}_{\{1,6,8\}\setminus\{k\}}^{\perp} \begin{cases} A_{23,7}B_{13,7}C_{12,7}, & k=1 \\ F_{23,7}G_{23,6}, & k=6 \\ H_{23,7}, & k=8 \end{cases}$$

where naturally any of these users can decode its intended subfile. For the remaining users (users $2,3,7$) the receiving message takes the form

$$y_{1,23}^{6,7}(k \in \{2,3,7\}) = \mathbf{h}_k^T \mathbf{h}_{\{6,8\}}^{\perp} A_{23,7}B_{13,7}C_{12,7} + \mathbf{h}_k^T \mathbf{h}_{\{1,8\}}^{\perp} F_{23,7}G_{23,6} + \mathbf{h}_k^T \mathbf{h}_{\{1,6\}}^{\perp} H_{23,7}.$$

We can easily see that each of these users can decode its desired subfile by caching-out any other interfering message.

5.4 Bounds and Converses

5.4.1 Proof of Theorem 5.1

Toward proving Theorem 5.1, we adapt the approach of [35], to lower bound the delay for the case where, out of the K users, only K_1 users have a cache. The bound will prove tight for all

$$L \geq \frac{K_2}{T_1^{(1)}(K_1, \gamma_1)} - 1 = \frac{K_2(1+K_1\gamma_1)}{K_1(1-\gamma_1)} - 1. \tag{5.39}$$

The proof (for $L = 1$) tracks closely steps[1] from [35] which — for the case of $K_1 = K$ (where all users have caches) — employed index coding to bound the performance of coded caching. Some of these steps are sketched here for the sake of completeness. Particular care is taken here to properly construct the bound's counting arguments in a way that accounts for the fact that specific symmetries that are essential to the approach in [35], do not directly hold here, simply because the set $\mathcal{K}_1 = [K_1]$ of users that enjoy side information is only a subset of the users that request files.

We will begin with lower bounding, first for the case of $L = 1$, the delay $T(\boldsymbol{d}, \chi)$ for any generic caching-delivery strategy χ and any demand vector $\boldsymbol{d} \in \mathcal{D}_{wc} \triangleq \{\boldsymbol{d} : \ d_i \neq d_j, \ i, j \in [K], \ i \neq j\}$ whose K entries are all different. In the following, we use $\mathcal{Z}_i$ to denote the cache of each user i, where naturally $\mathcal{Z}_i = \emptyset$ for $i \in \mathcal{K}_2 \triangleq [K] \setminus \mathcal{K}_1$.

Distinct caching problems and their corresponding index coding equivalents We first follow closely the approach in [35] to describe the association between index coding and our specific caching scenario here. As in [35], each caching problem (defined by a demand vector $\boldsymbol{d} \in \mathcal{D}_{wc}$) is converted into an index coding problem, by having each requested file W^{d_i} split into 2^{K_1} disjoint subfiles $W^{d_i}_{\mathcal{T}}, \mathcal{T} \in 2^{[K_1]}$, where $\mathcal{T} \subset [K_1]$ indicates the set of users that have $W^{d_i}_{\mathcal{T}}$ cached. Since no subfile of the form $W^{d_i}_{\mathcal{T}}, \ \mathcal{T} \ni i$ is requested, the index coding problem here is defined by

$$K_1 2^{K_1 - 1} + K_2 2^{K_1}$$

requested subfiles, which form the nodes of 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 $W^{d_i}_{\mathcal{T}}, \mathcal{T} \not\ni i$) and $\mathcal{E}_{\mathcal{G}}$ is the set of direct edges of the graph. We recall that an edge from node $W^{d_i}_{\mathcal{T}}$ to $W^{d_{i'}}_{\mathcal{T}'}$ exists if and only if $i' \in \mathcal{T}$.

As in [35], this allows us to lower bound $T(\boldsymbol{d}, \chi)$ by using the index-coding converse from [94] which says that for a given $\boldsymbol{d}, \chi$ — with corresponding

[1] We note in advance that a naive adaptation of the approach in [35], where we would simply account for a reduced sum-cache constraint $K_1 M$ corresponding to a redundancy $t = \frac{K_1 M}{N}$, would yield a loose bound; for example when $L = 1$, this naive bound would be $T \geq \frac{K-t}{t+1}$ which would then translate to $T \geq \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1} + \frac{K_2}{1+K_1\gamma_1}$ which is loose as the optimal delay will turn out to be $T = \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1} + K_2$.

side information graph $\mathcal{G}_{\boldsymbol{d}} = (\mathcal{V}_{\mathcal{G}}, \mathcal{E}_{\mathcal{G}})$ with $\mathcal{V}_{\mathcal{G}}$ vertices/nodes and $\mathcal{E}_{\mathcal{G}}$ edges — the delay is bounded as

$$T \geq \sum_{\mathcal{V} \in \mathcal{V}_{\mathcal{J}}} |\mathcal{V}| \tag{5.40}$$

for every acyclic induced subgraph $\mathcal{J}$ of $\mathcal{G}_{\boldsymbol{d}}$, 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 $\mathcal{V}$.

The following describes the acyclic graphs, and also directly shows that these remain acyclic after they are enlarged to account for the content requested by the cache-less users. In the following we will consider permutations $\sigma \in S_{K_1}$ from the symmetric group S_{K_1}, and, for a given demand vector $\boldsymbol{d}$, we will use $\mathcal{A}_{\boldsymbol{d}} \triangleq \cup_{i \in [K] \setminus [K_1]} W^{d_i}$ to denote the union of all content in $\boldsymbol{d}$ that is requested by the users in $[K] \setminus [K_1]$.

Lemma 5.1. *For any $\boldsymbol{d}$ and any $\sigma \in S_{K_1}$, an acyclic subgraph $\mathcal{J}_{\boldsymbol{d},\sigma}$ of $\mathcal{G}_{\boldsymbol{d}}$, is designed here to consist of all subfiles $\{W_{\mathcal{T}}^{d_{\sigma(i)}}, \ \forall i \in [K_1], \forall \mathcal{T} \subseteq [K_1] \setminus \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}\}$, and the enlarged graph $\mathcal{J}_{\boldsymbol{d},\sigma} \cup \mathcal{A}_{\boldsymbol{d}}$ is also acyclic.*

Proof. The proof that the subgraph $\mathcal{J}_{\boldsymbol{d},\sigma}$ is acyclic is direct from [35, Lemma 1]. The proof that $\mathcal{J}_{\boldsymbol{d},\sigma} \cup \mathcal{A}_{\boldsymbol{d}}$ is also an acyclic graph, i.e., that the addition (on the original $\mathcal{J}_{\boldsymbol{d},\sigma}$) of all the nodes corresponding to $\mathcal{A}_{\boldsymbol{d}}$ does not induce any cycles, follows by first recalling that a directed edge from node $W_{\mathcal{T}}^{d_i}$ to $W_{\mathcal{T}'}^{d_{i'}}$ exists if and only if $i' \in \mathcal{T}$, which thus tells us that an edge cannot be drawn from any node representing content from $\mathcal{A}_{\boldsymbol{d}}$, because any cache-less user $i \in K \setminus [K_1]$ cannot belong to any such $\mathcal{T}$ simply because $\mathcal{T} \subset [K_1]$. □

Given the acyclic subgraph $\mathcal{J}_{\boldsymbol{d},\sigma} \cup \mathcal{A}_{\boldsymbol{d}}$, we combine Lemma 5.1 with (5.40) to get

$$T(\boldsymbol{d}, \chi) \geq T^{LB}(\sigma, \boldsymbol{d}, \chi) \tag{5.41}$$

where

$$\begin{aligned} T^{LB}(\sigma, \boldsymbol{d}, \chi) &\triangleq \sum_{\mathcal{V} \in \mathcal{V}_{\mathcal{J}_{\boldsymbol{d},\sigma}} \cup \mathcal{A}_{\boldsymbol{d}}} |\mathcal{V}| \\ &= \sum_{\mathcal{T} \subseteq [K_1] \setminus \{\sigma(1)\}} |W_{\mathcal{T}}^{\boldsymbol{d}_{\boldsymbol{\sigma}(\mathbf{1})}}| + \sum_{\mathcal{T} \subseteq [K_1] \setminus \{\sigma(1), \sigma(2)\}} |W_{\mathcal{T}}^{\boldsymbol{d}_{\boldsymbol{\sigma}(\mathbf{2})}}| + \ldots \\ &+ \sum_{\mathcal{T} \subseteq [K_1] \setminus \{\sigma(1), \ldots, \sigma(K_1)\}} |W_{\mathcal{T}}^{\boldsymbol{d}_{\boldsymbol{\sigma}(\boldsymbol{K_1})}}| + |\mathcal{A}_{\boldsymbol{d}}|. \end{aligned} \tag{5.42}$$

Then, as in [35], we average over worst-case demands to get

$$\begin{aligned} T^* &\triangleq \min_{\chi} \max_{\boldsymbol{d} \in \mathcal{D}_{W_c}} T(\boldsymbol{d}, \chi) \\ &\geq \min_{\chi} \max_{\boldsymbol{d} \in \mathcal{D}_{W_c}} \max_{\sigma \in S_{K_1}} T^{LB}(\sigma, \boldsymbol{d}, \chi) \\ &\geq \min_{\chi} \frac{1}{|\mathcal{D}_{W_c}|} \frac{1}{|S_{K_1}|} \sum_{\sigma \in S_{K_1}} \sum_{\boldsymbol{d} \in \mathcal{D}_{W_c}} T^{LB}(\sigma, \boldsymbol{d}, \chi) \\ &\geq \min_{\chi} \frac{1}{P(N,K) K_1!} \sum_{\sigma \in S_{K_1}} \sum_{\boldsymbol{d} \in \mathcal{D}_{W_c}} T^{LB}(\sigma, \boldsymbol{d}, \chi) \end{aligned} \tag{5.43}$$

where in the above we use $P(N,K) \triangleq \frac{N!}{(N-K)!}$.

Rewriting the summation in (5.43), we get

$$\sum_{\sigma\in S_{K_1}} \sum_{\boldsymbol{d}\in\mathcal{D}_{W_c}} T^{LB}(\sigma,\boldsymbol{d},\chi) = \tag{5.44}$$

$$\sum_{i=0}^{K_1} \sum_{n\in[N]} \sum_{\mathcal{T}\subseteq[K_1]:|\mathcal{T}|=i} |W_{\mathcal{T}}^n| \cdot \underbrace{\sum_{\sigma\in S_{K_1}} \sum_{\boldsymbol{d}\in\mathcal{D}_{W_c}} \mathbb{1}_{\mathcal{V}_{\mathcal{J}_{\boldsymbol{d},\sigma}}}(W_{\mathcal{T}}^n)}_{\triangleq Q_i(W_{\mathcal{T}}^n)} + |\mathcal{A}_{\boldsymbol{d}}|$$

where $\mathcal{V}_{\mathcal{J}_{\boldsymbol{d},\sigma}}$ is the set of vertices in the acyclic component subgraph $\mathcal{J}_{\boldsymbol{d},\sigma}$ for a given $\boldsymbol{d},\sigma$ pair, and where $\mathbb{1}_{\mathcal{V}_{\mathcal{J}_{\boldsymbol{d},\sigma}}}(W_{\mathcal{T}}^n)$ denotes the indicator function which takes the value of 1 only if $W_{\mathcal{T}}^n \subset \mathcal{V}_{\mathcal{J}_{\boldsymbol{d},\sigma}}$, else it is set to zero.

Counting arguments accounting for cache-less users Our aim is to count the number of times, $Q_i(W_{\mathcal{T}}^n)$, that any specific subfile $W_{\mathcal{T}}^n$ appears in the summation in (5.44). To do this, we follow the same counting arguments as in [95, Section VII-C] which derives $Q_i(W_{\mathcal{T}}^n)$ for the case where K users share $\Lambda \leq K$ caches, where each cache $r \subset [\Lambda]$ serves Λ_r users. Adapting these steps[2] in [95, Section VII-C] gives that

$$\begin{aligned} Q_i = Q_i(W_{\mathcal{T}}^n) &\triangleq \sum_{\sigma\in S_{K_1}} \sum_{\boldsymbol{d}\in\mathcal{D}_{W_c}} \mathbb{1}_{\mathcal{V}_{\mathcal{J}_{\boldsymbol{d},\sigma}}}(W_{\mathcal{T}}^n) \\ &= \binom{N-1}{K-1} \sum_{r=1}^{K_1} P(K_1-i-1, r-1)(K_1-r)! \\ &\quad \cdot (K-1)!(K_1-1)!(K_1-i). \end{aligned} \tag{5.45}$$

Setting $x_i \triangleq \sum_{n\in[N]} \sum_{\mathcal{T}\subseteq[K_1]:|\mathcal{T}|=i} |W_{\mathcal{T}}^n|$ and recalling that

$$N = \sum_{i=0}^{K_1} x_i = \sum_{i=0}^{K_1} \sum_{n\in[N]} \sum_{\mathcal{T}\subseteq[K_1]:|\mathcal{T}|=i} |W_{\mathcal{T}}^n| \tag{5.46}$$

we combine (5.43), (5.44) and (5.45), to get

$$T \geq \sum_{i=0}^{K_1} \frac{Q_i}{P(N,K)K_1!} x_i. \tag{5.47}$$

We now resume counting to calculate $\frac{Q_i}{\Lambda! P(N,K)}$ for each $i = 0, 1, \ldots, K_1$.

[2]The following expression could not have been derived, had we simply substituted K for K_1, in the corresponding Q_i expression in [35]. Such a naive approach would have essentially corresponded to treating the cache-less and cache-aided cases separately, and would not have allowed us to guarantee, among other things, that both cache-less and cache-aided users request different files.

Applying (5.45), we see that

$$\begin{aligned}
\frac{Q_i}{K_1!P(N,K)} &= \frac{(N-1)!(N-K)!}{(K-1)!(N-K)!K_1!N!} \\
&\cdot \sum_{r=1}^{K_1}(K-1)!(K_1-i)P(K_1-i-1,r-1)(K_1-r)! \\
&= \frac{1}{K_1!N}\sum_{r=1}^{K_1}(K_1-i)P(K_1-i-1,r-1)(K_1-r)! \\
&= \frac{1}{K_1!N}\sum_{r=1}^{K_1}\frac{(K_1-i)(K_1-i-1)!(K_1-r)!}{(K_1-i-r)!} \\
&= \frac{1}{K_1!N}\sum_{r=1}^{K_1}\frac{(K_1-i)!(K_1-r)!}{(K_1-i-r)!} \\
&= \frac{1}{N}\sum_{r=1}^{K_1}\frac{(K_1-i)!(K_1-r)!i!}{K_1!(K_1-i-r)!i!} \\
&= \frac{1}{N}\sum_{r=1}^{K_1}\frac{\binom{K_1-r}{i}}{\binom{K_1}{i}} = \frac{\binom{K_1}{i+1}}{\binom{K_1}{i}N} = \frac{K_1-i}{(i+1)N}.
\end{aligned} \tag{5.48}$$

Now substituting (5.48) into (5.47), we get that

$$T(\chi) \geq \sum_{i=0}^{K_1}\frac{K_1-i}{(i+1)N}x_i + \frac{K_1!P(N,K)}{K_1!P(N,K)}\underbrace{|\mathcal{A}_{\boldsymbol{d}}|}_{K_2} \tag{5.49}$$

where the use of the fraction $\frac{K_1!P(N,K)}{K_1!P(N,K)} = 1$ is meant to remind us the number of times acyclic graphs corresponding to $\mathcal{A}_{\boldsymbol{d}}$ were invoked in the summation in (5.44), and where we also note that the expression above follows from the fact that all $\boldsymbol{d} \in \mathcal{D}_{wc}$ force $|\mathcal{A}_{\boldsymbol{d}}| = K_2$.

Optimization At this point we observe that the crucial constant $\frac{K_1-i}{(i+1)N}$ derived for the part of the subgraph corresponding to cache-aided users, matches exactly the number $\frac{K_1-i}{(i+1)N}$ derived in [35] for the $K = K_1$ case where all users can have a cache. Consequently, under the same file-size constraint given in (5.46), and given the current cache-size constraint $\sum_{i=0}^{K_1} i \cdot x_i \leq K_1 M$, the expression in (5.49) serves as a lower bound on the delay of scheme χ whose cache placement implies the set $\{x_i\}$.

Then, following the exact minimization steps in [37, Proof of Lemma 2], we get

$$T(\chi) \geq \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1} + K_2 \tag{5.50}$$

for integer $K_1\gamma_1$, whereas for all other values of $K_1\gamma_1$, this is extended to its convex lower envelop.

This concludes lower bounding $\max_{\boldsymbol{d}} \in \mathcal{D}_{wc} T(\boldsymbol{d}, \chi)$, and thus — given that the right hand side of (5.50) is independent of χ — lower bounds the performance for any scheme χ, hence concluding the proof of the converse for Theorem 5.1 for the case of $L = 1$.

5.4.2 Converse and gap to optimal of Theorem 5.2

Let us first consider the gap to optimal for the case of $K_2 \geq (L-1)T_1^{(1)}$, where we recall that $T_1^{(1)} = \frac{K_1(1-\gamma_1)}{1+K_1\gamma_1}$.

We have seen that when $K_2 = \alpha(L-1)T_1^{(1)} \geq (L-1)T_1^{(1)}$ (i.e., when $\alpha \geq 1$), the achievable delay in Eq. (5.6) takes the form

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{(L-1)T_1^{(1)} + K_1(1-\gamma_1)}{K_1\gamma_1 + L} + \frac{K_2 - (L-1)T_1^{(1)}}{L} \tag{5.51}$$

$$= \frac{(L-1)T_1^{(1)} + K_1(1-\gamma_1)}{L + K_1\gamma_1} + (\alpha - 1)\frac{L-1}{L}T_1^{(1)} \tag{5.52}$$

$$= \frac{T_1^{(1)}}{L}(\alpha L - \alpha + 1). \tag{5.53}$$

For a lower bound on the minimum possible delay, we use

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) \geq \frac{\min\{K_2, L\}}{L} = \min\{1, \frac{\alpha(L-1)T_1}{L}\} \tag{5.54}$$

corresponding to the optimal delay required to satisfy only the cache-less users. A quick calculation of the ratio between Eq. (5.51) and Eq. (5.54), provides with the multiplicative gap of

$$g = \frac{\alpha L - \alpha + 1}{\alpha L - \alpha} = 1 + \frac{1}{\alpha(L-1)} \leq 2. \tag{5.55}$$

When $K_2 < L$, then $\alpha(L-1)T_1^{(1)} < L$, which again gives

$$g = \frac{T_1^{(1)}}{L}(\alpha L - \alpha + 1) < \frac{T_1^{(1)}(\alpha L - \alpha - 1)}{\alpha(L-1)T_1^{(1)}} \leq 2. \tag{5.56}$$

For the case of $K_n = \alpha(L-1)T_1, \quad \alpha \leq 1$, the lower bound takes the form

$$T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) \geq \max\left\{\frac{\min\{L, K_2\}}{L}, \frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1 + L}\right\} \tag{5.57}$$

where the first term corresponds to the optimal performance of an 'easier' system where all the cache-aided users are removed, and where the second term corresponds to an easier system where all cache-less users are removed, and where — for this latter type of system, we know from [11] that treating $K_1\gamma_1 + L$ users at a time is at most a factor of 2 from optimal, under the

assumptions of linear and one-shot schemes. Combining Eq. (5.57) with the achievable $T_L(K_1, \gamma_1, K_2, \gamma_2 = 0) = \frac{K_2+K_1(1-\gamma_1)}{K_1\gamma_1+L}$ from Eq. (5.7), yields a gap of

$$g = \frac{\frac{K_2+K_1(1-\gamma_1)}{K_1\gamma_1+L}}{\max\left\{\frac{K_2}{L}, \frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}\right\}}.$$

To bound this gap, note that if $\frac{K_2}{L} > \frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}$ then we know from before that

$$g = \frac{\frac{K_2+K_1(1-\gamma_1)}{K_1\gamma_1+L}}{\frac{K_2}{L}} = \frac{L}{K_1\gamma_1} + \frac{\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}}{\frac{K_2}{L}} \leq 1 + 2,$$

where we used that $\frac{K_2}{L} > \frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}$.

Similarly when $\frac{K_2}{L} < \frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}$, the gap is bounded as

$$G = \frac{\frac{K_2+K_1(1-\gamma_1)}{K_1\gamma_1+L}}{\frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}} = \frac{\frac{K_2}{K_1\gamma_1+L}}{\frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}} + 2 \leq 3$$

where the last step considers that $\frac{K_2}{L} < \frac{1}{2}\frac{K_1(1-\gamma_1)}{K_1\gamma_1+L}$.

This concludes the proof of Theorem 5.2. □

5.4.3 Proof of Theorem 5.3

In order to form an outer bound, we will consider a system with an L-antenna transmitter, which transmitter can communicate L distinct messages $\{W_1, ..., W_L\}$ while user $k \in [K]$ will receive linear combination $\mathcal{L}_k(W_1, ..., W_L)$ of these messages. The main difference between the considered system and the system model is that, in this case, each user can select any arbitrary linear combination and then inform its decision to the transmitter. Further, we will assume that each user requests a different file, which rules out any natural multicasting opportunities.

By making use of the result from Theorem 5.1 we can see that, in order for any single transmitted message to be decoded it may contain information only for the cache-aided or the cache-less users. This means that we can separate the messages into two sets, i.e. $\{W_n\}_{n=1}^{P}$ and $\{W_c\}_{c=1}^{C}$, where the former are intended for the cache-less users, while the later are intended for the cache-aided users and where $P + C = L$.

Using the above allocation of messages to each group, we further assume that each message intended for the cache-aided users contains $K\gamma+1$ subfiles, thus the total amount of information transmitted to the cache-aided users can be at most $C(K\gamma + 1)$. Thus, the delivery time takes the form

$$T \geq \min_{C} \max\left\{\frac{T_1^{(1)}(\tilde{L} - 1)}{L - C}, \frac{K_c(1-\gamma)}{C(1+K\gamma)}\right\} = T_1^{(1)}\frac{\tilde{L}}{L}. \tag{5.58}$$

Since the performance of the system matches the lower bound of Eq. (5.58), then the delay provided in Theorem 5.3 is optimal under the assumption of uncoded placement.

5.5 Extension of the cache-aided scheme

In this section we present an extension of Algorithm 5.2 to accommodate any values $L_1, L_2 \in [1, L-1]$, such that $L_1 + L_2 = L$. The main premise is to increase the per-type subpacketization by value $d \in \mathbb{N}$, such that $d{\cdot}L_1, d{\cdot}L_2 \in \mathbb{N}$, thus the total subpacketization becomes

$$S = d^2(L_1 + K_1\gamma_1)\binom{K_1}{K_1\gamma_1}(L_2 + K_2\gamma_2)\binom{K_2}{K_2\gamma_2}. \tag{5.59}$$

The new scheme works by repeating d^2 times Alg. 5.2, with the difference that some transmissions will allocate $\lceil L_1 \rceil$ streams to users of set $\mathcal{K}_1$ and at the same time it will allocate $\lfloor L_2 \rfloor$ streams to set $\mathcal{K}_2$ and in some transmissions will allocate $\lfloor L_1 \rfloor$ streams to users of set $\mathcal{K}_1$ and at the same time it will allocate $\lceil L_2 \rceil$ streams to set $\mathcal{K}_2$.

This way, it allows the average allocation of L_1 and L_2 streams to each user type, which leads to the DoF $D_L(K_1, \gamma_1, K_2, \gamma_2) = L + K_1\gamma_1 + K_2\gamma_2$.

Chapter 6

Channel Unevenness Bottleneck

In this chapter, our focus is concentrated on the worst-user effect that is caused by the uneven channel nature of wireless channels, which is intertwined with the multicasting transmission structure of Coded Caching. This performance bottleneck has its roots in the observation that in a wireless setting users will experience different channel capacities thus, a multicast message intended to convey information to many users at a time will have to be delivered with the worst user's rate so as to be decodable by all users. The uneven channels bottleneck does not only affect a single user, but instead the whole system performance, thus even a single user's low-capacity channel can reduce the performance of all users. This performance degradation can be seen as an important drawback in the implementation of Coded Caching in wireless communications.

The model of interest is the wireless Broadcast Channel (BC) with a single antenna transmitter that serves K cache-aided, single antenna receivers with potentially different channel rates.

Related Work

Many previous works have considered user channel qualities not being equal, in the context of Coded Caching. Specifically, a lot of effort has been recently made in understanding the impact of coded caching under the more realistic scenario where the transmit Signal-to-Noise Ratio (SNR) is finite. In this direction many interesting results, showed that the performance of single-stream Coded Caching can outperform that of multiple streams in the low SNR region [41], while the works in [43, 96] (see also [97]) revealed the importance in designing the multicast beamformers in order to take into account the fact that users are cache-aided. Further, the work in [74] studied the SNR performance of multi-antenna algorithms depending on their subpacketization requirements.

Moreover, the works in [98, 99] used feedback to select the user XORs that can maximize the sum-rate and a fairness function that ensures the that users with lower rates can also receive their requested content in a timely manner.

The works in [100,101] (see also [102–104] studied broadcast channels with channel erasures and showed how an erasure to one user can still be useful in the other users as side information.

Works [77,78] studied the setting with K transmitters each storing either part or the whole library, and which transmitters where tasked with serving K cache-aided receivers. In this setting, the direct links are "strong" thus are able to facilitate a high rate, while the cross links are "weak". The proposed setting considered the case where no Channel State Information at the Transmitters (CSIT) was available to the transmitters and it was shown that the lack of CSIT can be ameliorated by exploiting the topological factor of the channel and the multicast transmissions.

Further, the work in [79] studied a multiple-antenna setting where the antennas were used as a means to increase diversity and achieve higher rates. The work showed that in order to achieve a scalable transmission rate by transmitting a single XORed message (as in the algorithm of [1]) but from a total of L antennas would require the number of antennas to scale as $\mathcal{O}(\ln K)$ in order for a scaling, with the number of users rate to be achieved.

A similar setting to the one proposed in this work was studied in works [105,106]. Specifically, the work in [105] studied the single antenna channel where each user has a - potentially - different channel strength in the finite Signal-to-Noise-Ratio (SNR) region. The authors proposed algorithms that could outperform the naive implementation of the algorithm of [1]. Further, the work in [106] studied the single antenna channel where the K users are split into two categories, those with full-rate channels and those with reduced rate channels. The authors designed a superposition code-based algorithm and showed its order optimal performance.

Another similar setting to the one presented in this chapter, was explored in [107] where the authors divided each file into $2^K - 1$ subfiles (each for a different user subset) and further used opportunistic scheduling of users and superposition coding to maximize the rate.

Moreover, the work in [41] (see also [108]) exploited multiple antennas to combat the worst user effect, by transmitting a XOR message, as in the scheme of [1] and multiplied it by a beamforming vector designed to allocate power in such a way that will "raise" the worst user's channel.

Further, the work in [109] has considered the multiple-antenna channel with asymmetric channel qualities and created the content placement and subsequent delivery phases in such a way to allow for a scaling rate.

Results Overview

Here we will propose a new algorithm that is designed to elevate the single-antenna performance when users are experiencing uneven channels. The algorithm borrows the placement strategy and generation of XORs from [1] but delivers these XORs in a more efficient manner, based on statistical knowledge of the user channel strengths and through the use of superposition coding. For this proposed setting we prove a performance bound and

further continue to show that our proposed algorithm performs within a multiplicative gap of 4 from this bound.

A very interesting result that is derived from this work is that, under the assumption of users being sorted with respect to their channel strength, i.e. $\alpha_k \leq \alpha_{k+1}, \forall k \in [K]$, when the channel qualities satisfy

$$\alpha_k \geq 1 - e^{-k\cdot\gamma}, \quad \forall k \in [K] \tag{6.1}$$

then the system can perform exactly as if there was no channel degradation, thus showing how the "strong" users can elevate the "weaker" ones, and further showing that systems with smaller cache sizes are more immune to channel unevenness.

6.1 System Model

We consider the single-input-single-output (SISO), wireless, broadcast channel (BC) with K single-antenna receivers.

The objective is to design the pre-fetching at the caches and the delivery of requests in such a way to minimize the delivery time $T(K, \gamma, \boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ denotes the vector containing the channel strength of each of the K users. The transmission/reception model follows the Generalized Degrees of Freedom (GDoF) framework of [45] (see also [46, 49]), thus a message received at some user $k \in [K]$ takes the form

$$y_k = \sqrt{P^{\alpha_k}} h_k x + w_k, \tag{6.2}$$

where P represents the transmitting power, $x \in \mathbb{C}$ is the output signal of the transmitter satisfying the power constraint $\mathbb{E}\{|x|^2\} \leq 1$ and $h_k \in \mathbb{C}$ corresponds to the channel coefficient of user k. Further, $\alpha_k \in (0, 1]$ represents the normalized, by factor $\log P$, link strength of user k. Finally, $w_k \sim \mathbb{C}\mathcal{N}(0, 1)$ represents the Gaussian noise at user k. From the above we can deduce the average signal-to-noise-ratio (SNR) at user $k \in [K]$ as

$$\mathbb{E}\{|y_k|^2\} = P^{\alpha_k} \tag{6.3}$$

which amounts to a (normalized) user rate of $r_k = \alpha_k$. It should be noted that without loss of generality $\alpha = 1$ corresponds to the highest possible channel strength, i.e. $\alpha_K = 1$.

For the model under study, we consider the most generic case, where each user has – potentially – a distinct link strength, namely, any two channel strength parameters $\alpha_k,\ k \in [K]$ may be different from each other. Without loss of generality, we assume an ascending ordering of the strength set $\boldsymbol{\alpha} \triangleq \{\alpha_k\}_{k=1}^K$, i.e., $\alpha_{k+1} \geq \alpha_k$, for any $k \in [K]$.

Notation We will use X_σ to denote a XOR generated as in the algorithm of [1] i.e.,

$$X_\sigma = \bigoplus_{k\in\sigma} W^{d_k}_{\sigma\setminus\{k\}}. \tag{6.4}$$

Further, we will use sets $\mathcal{X}_k,\ k \in [K]$ to denote the sets of XORs whose minimum numbered user is user k, i.e.

$$\mathcal{X}_k = \Big\{ X_\sigma \ \text{ iff } \ \min\{\sigma\} = k \Big\}, \tag{6.5}$$

where for $k \leq K - K\gamma - 1$, it follows that

$$|\mathcal{X}_k| = \binom{K-k+1}{K\gamma+1} - \binom{K-k}{K\gamma+1} \overset{(\star)}{=} \binom{K-k}{K\gamma}, \tag{6.6}$$

where $(\star)$ comes directly from Pascal's triangle and further,

$$\left| \bigcup_{m=1}^{k} \mathcal{X}_m \right| = \binom{K}{K\gamma+1} - \binom{K-k}{K\gamma+1}. \tag{6.7}$$

6.2 Main Results

In this section we present[1] the achievable performance of the algorithm of [1] under a naive implementation and further, in Theorem 6.1 we describe the achievable performance of our scheme along with its order optimality.

Proposition 6.1. *In the SISO-BC with user channel strength $\alpha_k \leq 1$ and caching of fraction γ at each user, a naive implementation of the algorithm of [1] would amount to the delivery time of*

$$T_{uc}(K, \gamma, \boldsymbol{\alpha}) = \frac{1}{\binom{K}{K\gamma}} \sum_{\sigma \subseteq [K],\ |\sigma| = K\gamma+1} \max_{i \in \sigma} \left\{ \frac{1}{\alpha_i} \right\}. \tag{6.8}$$

Proof. The proof comes directly by allocating normalized time T_σ to XOR X_σ, according to

$$T_\sigma = \max_{w \in \sigma} \left\{ \frac{1}{\alpha_w} \right\}$$

which is required by the user with the weakest channel, i.e. $U_w,\ w \in \sigma$, of XOR X_σ, to decode almost surely the transmitted message. □

Theorem 6.1. *In the K-user SISO Broadcast Channel with single antenna receivers, receiver channel strengths $\{\alpha_k\}_{k=1}^{K}, \alpha_k \leq \alpha_{k+1},\ \forall k \in [K]$ and each receiver equipped with a cache of normalized size γ, the achieved worst-case delivery time takes the form*

$$T_{sc}(K, \gamma, \boldsymbol{\alpha}) = \max_{w \in [K]} \left\{ \frac{1}{\alpha_w} \cdot \frac{\binom{K}{K\gamma+1} - \binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}} \right\} \tag{6.9}$$

and has a multiplicative gap from the optimal performance of at most 4.

[1] We note that the derivation of the expression in Proposition 6.1 has also been calculated in previous works which focused on a finite SNR region analysis of Coded Caching (see [41, 105]).

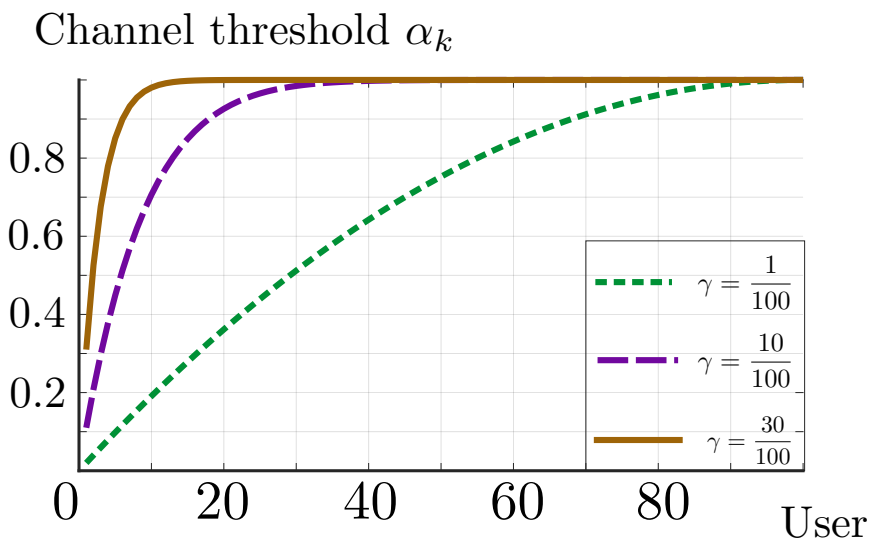

Figure 6.1: The threshold $\alpha_{th,w}$ for each $w \in [K]$ with respect to different values of γ.

Proof. The achievability part of the scheme is described in Algorithm 6.1 of Section 6.3, while a performance bound as well as the multiplicative performance gap between the proposed algorithm and bound are proved in Section 6.4.1. □

Corollary 6.1. *In a K-user SISO BC with receiver channel strengths $\{\alpha_w\}_{w=1}^K$ satisfying $\alpha_w \leq \alpha_{w+1}$, $\forall w \in [K]$ and each receiver equipped with a cache of normalized size γ, the maximally known performance*

$$T(K, \gamma, \boldsymbol{\alpha} = \mathbf{1}) = \frac{K(1-\gamma)}{1+K\gamma} \tag{6.10}$$

i.e., achieved when all uses have full-rate channels $\left(\{\alpha_w = 1\}_{w=1}^K\right)$ can also be achieved if users have degraded channels as follows

$$\alpha_w \geq \alpha_{th,w} = 1 - \frac{\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma+1}} \approx 1 - e^{-w\gamma}, \quad \forall w \in [K]. \tag{6.11}$$

Proof. The proof is direct by use of Eq. (6.9), the Sterling approximation $\binom{n}{k} \approx \left(\frac{n}{k}\right)^k$ and the limit

$$\lim_{K\to\infty} \left(1 - \frac{b}{K}\right)^K = e^{-b}. \tag{6.12}$$

□

Remark 6.1. *In Figure 6.1 we can see the threshold channel strengths for a setting with $K = 100$ users and different values of γ, that are required to achieve the same performance as in the non-degraded version.*

As illustrated in Figure 6.1 and from Eq. (6.11), we can deduce that as the cache size grows, and more precisely as the coded caching gain increases, achieving the the maximum performance of the non-degraded channel would require a big portion of the user channels needs to have the maximum channel strength.

Example 6.1. *Let us consider the wireless Single-Input-Single-Output (SISO) BC with K users, each equipped with a cache of normalized size γ and let us further assume that the channel strength of the first user is equal to $\alpha_1 = \frac{1}{K} + \gamma$, while the remaining $K-1$ users have the maximum channel strengths i.e., $\alpha_2 = ... = \alpha_K = \alpha = 1$.*

Assuming a naive implementation of the algorithm of [1] – where the designed XORs (which are designed for the equal-strength case) are sent sequentially one after the other – the achieved delay, T_1^{uc}, under uneven channel strengths would take the form

$$T_1^{uc} = (1-\gamma)\frac{1}{\alpha_1} + \frac{(K-1)(1-\gamma)}{1+K\gamma}\frac{1}{\alpha} \tag{6.13}$$

$$= \frac{1-\gamma}{\frac{1+K\gamma}{K}} + \frac{(K-1)(1-\gamma)}{1+K\gamma} \approx 2T_1. \tag{6.14}$$

Thus, a naive implementation of the algorithm of [1] when even a single user is experiencing a bad channel can almost double the experienced delivery time, and which has the equivalent effect of reducing by half the multicasting gain.

6.3 Placement and Delivery Algorithms

In this section, we present the achievable scheme, beginning with the placement phase at the receivers, then continuing with the description of the transmission policy during the delivery phase and concluding with the decoding process at the users.

6.3.1 Placement Phase

We begin by filling the caches of the users with content from the library. Our approach borrows from the placement algorithm of [1] thus, does not assume any knowledge of the channel strengths at the time of the placement. To this end, each file $W^n, n \in [N]$, is subpacketized into $S = \binom{K}{K\gamma}$ subfiles and the cache of user $k,\ k \in [K]$ is given by

$$\mathcal{Z}_k = \{W_\tau^n : \tau \subset [K], k \in \tau, |\tau| = K\gamma, \forall n \in [N]\} \tag{6.15}$$

which is easy to show that respects the cache-size constraint.

6.3.2 Delivery Algorithm

During the delivery phase, the goal is to successfully communicate all $\binom{K}{K\gamma+1}$ XORs (cf. Eq. (6.4)). To this end, in every communication slot we will split the available transmission power into $K - K\gamma$ levels (power levels). In each power level we will encode XORs that are meant for some particular user. For example, the highest power level, i.e. Level 1, will contain XORs that are intended for the weakest user (user 1) i.e., XORs belonging in set $\mathcal{X}_1$. Further, power level 2 will contain XORs that are intended for user 2, but are not intended for user 1, i.e., belonging in set $\mathcal{X}_2$ and so on. This power allocation for each XOR allows the weakest user of the XOR to be able to decode it, thus all other users that want to retrieve this XOR can also decode it successfully.

The challenge lies in finding the appropriate power allocation for each power level, such that the overall delay is minimized.

1 Let $\alpha_k \leq \alpha_{k+1},\ \forall k \in [K]$
2 Find $w \in [K]$ such that

$$w = \arg\min_{k\in[K]} \left\{ \frac{\binom{K-k}{K\gamma+1}}{\alpha_k} \right\}. \tag{6.16}$$

3 Set : $\beta_0 = 0$ and for $k \in [K - K\gamma - 1]$ set

$$\beta_i = \frac{|\cup_{k=1}^{i} \mathcal{X}_k|}{|\cup_{k=1}^{w} \mathcal{X}_k|}\alpha_w = \frac{\binom{K}{K\gamma+1} - \binom{K-i}{K\gamma+1}}{\binom{K}{K\gamma+1} - \binom{K-w}{K\gamma+1}}\alpha_w. \tag{6.17}$$

for *all* $k \in [K - K\gamma - 1]$ **do**
4 Encode $x_k \leftarrow \text{NEW}(\mathcal{X}_k)$
5 with power

$$P_k = P^{-\beta_{k-1}} - P^{-\beta_k} \tag{6.18}$$

6 and rate

$$r_k = \beta_k - \beta_{k-1} = \frac{\binom{K-k}{K\gamma}}{\binom{K}{K\gamma+1} - \binom{K-w}{K\gamma+1}}\alpha_w. \tag{6.19}$$

7 **end**
8 Transmit x_k's concurrently.

Algorithm 6.1: Superposition Coding-Based Delivery

In Algorithm 6.1 we describe the power and rate allocation for each XOR. The algorithm begins by identifying the bottleneck user (Step 2). This is done by "demanding" that all XORs intended for users up to and including user $w \in [K]$ to be decodable by user w. Further, in Step 3 are calculated the

power level coefficients β_i.

In Step 4, for every $k \in [K - K\gamma - 1]$ a new XOR is selected from set $\mathcal{X}_k$ and is encoded in message x_k, with power $P_k = P^{-\beta_{k-1}} - P^{-\beta_k}$ (Step 5) and rate $\frac{\binom{K-k}{K\gamma}}{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}\alpha_w$ (Step 6). In Step 8 all the x_k messages are transmitted concurrently.

6.3.3 Decoding at the Users

Let us see the received message at user $k \in [w]$, which takes the form

$$\begin{aligned} y_{k\in[w]} =& h_k\sqrt{P^{\alpha_k}}\sum_{m_1=1}^{k} x_{m_1} + h_k\sqrt{P^{\alpha_k}}\sum_{m_2=k+1}^{K-K\gamma} x_{m_2} \\ =& \underbrace{h_k\sqrt{P^{\alpha_k}}x_1}_{\log P\approx\alpha_k-\alpha_w\cdot\frac{\binom{K}{K\gamma+1}-\binom{K-1}{K\gamma+1}}{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}} + \cdots + \underbrace{h_k\sqrt{P^{\alpha_k}}x_k}_{\log P\approx\alpha_k-\alpha_w\cdot\frac{\binom{K}{K\gamma+1}-\binom{K-k}{K\gamma+1}}{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}} \\ &+ \underbrace{h_k\sqrt{P^{\alpha_k}}\sum_{m_2=k+1}^{K-K\gamma} x_{m_2}}_{\log P\approx\alpha_k-\alpha_w\cdot\frac{\binom{K}{K\gamma+1}-\binom{K-i}{K\gamma+1}}{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}} . \end{aligned} \tag{6.20}$$

From the power and rate allocation for each of these messages (cf. Eq. (6.18) and Eq. (6.19)) we can see that the messages of the second summation have power that is below the noise level, for this particular user through the use of Successive Interference Cancellation (SIC), User k can decode any message x_m, $m \leq k$, which messages are ones that potentially include relevant information for this user.

6.3.4 Delay calculation

The total delay of the scheme is the maximum delivery time over all x_k messages to deliver the set of XORs $\mathcal{X}_k$ i.e.,

$$T_{sc}(K,\gamma,\boldsymbol{\alpha}) = \max_{k\in[K-K\gamma-1]}\left\{\frac{|\mathcal{X}_k|}{\binom{K}{K\gamma}}\cdot\frac{1}{r_k}\right\} \tag{6.21}$$

$$= \max_{w\in[K-K\gamma-1]}\left\{\frac{1}{\alpha_w}\cdot\frac{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}}\right\}. \tag{6.22}$$

6.4 Bounds and Proofs of Converses

6.4.1 Optimality Gap of the Performance of Theorem 6.1

In this section, we prove the multiplicative gap of $G = 4$ between the achievable performance $T_{sc} \triangleq T_{sc}(K,\gamma,\boldsymbol{\alpha})$ of Eq. (6.9) of Theorem 6.1 and

a lower bound, denoted by T_e. We bound the performance of our system using a similar system where, now, users 1 through w (we remind here that user w corresponds to the bottleneck user) have channel capacities of $\alpha_k = \alpha_w \triangleq \alpha, \ \forall k \in [w]$, while the remaining users have the maximal channel capacity of 1. This essentially means that we elevate the capacities of the first w users to the capacity of the channel of user w and we further elevate the capacities of the remaining users to the maximum possible channel capacity. We denote the performance of this elevated system with T_e.

Proof. We begin by identifying a bound on the performance T_e. This is lower bounded by

$$T_e \geq \overbrace{\frac{1}{\alpha}}^{t_1} \overbrace{\frac{1}{2}\frac{w(1-\gamma)}{1+w\cdot\gamma}}^{t_2} \tag{6.23}$$

where term t_1 corresponds to the channel capacity of the first w users, i.e. $\alpha = \alpha_w$, while term t_2 corresponds to the minimum possible delay[2] for a cache-aided system with w users, each able to store fraction γ of the library and which performance bound is proved in [37].

We proceed with bounding the ratio T_{sc}/T_e to which end, we consider two separate cases, namely $w\gamma \gtrless 1$. First, by examining the case of $w\gamma < 1$ the ratio takes the form

$$\frac{T_{sc}}{T_e} \leq \frac{\frac{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}}}{\frac{1}{2}\frac{w(1-\gamma)}{(1+w\gamma)}} \leq \frac{w(1-\gamma)}{\frac{1}{2}\frac{w(1-\gamma)}{(1+w\gamma)}} \leq 4, \tag{6.24}$$

where we used the inequality $\frac{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}} \leq w(1-\gamma)$ which we prove in Section 6.4.2, and which informally says that the above inequality can be interpreted as the amount of information contained in all XORs meant for users 1 through w which is, at most, equal to the amount of information that needs to be transmitted to those users i.e., $w(1-\gamma)$.

Further, in the case of $w\gamma \geq 1$, the bound takes the form

$$\frac{T_{sc}}{T_e} = \frac{\frac{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}}}{\frac{1}{2}\frac{w(1-\gamma)}{(1+w\gamma)}} \leq \frac{\frac{\binom{K}{K\gamma+1}}{\binom{K}{K\gamma}}}{\frac{1}{2}\frac{w(1-\gamma)}{(1+w\gamma)}} \tag{6.25}$$

$$= \frac{\frac{K(1-\gamma)}{1+K\gamma}}{\frac{1}{2}\frac{w(1-\gamma)}{1+w\gamma}} = 2\frac{K(1+w\gamma)}{w(1+K\gamma)} = 2+2\frac{K-w}{w+Kw\gamma} \tag{6.26}$$

$$< 2\left(1+\frac{K+w}{w+wK\gamma}\right) < 2\left(1+\frac{K}{wK\gamma}\right) \leq 4 \tag{6.27}$$

which concludes the proof. □

[2] We need to note, here, that the factor of D_2 that is proven in [37] is slightly smaller, but for the sake of simplicity we use the more convenient $\frac{1}{2}$.

6.4.2 Bound on the difference of Binomials

In this section we prove the following corollary, which is used to derive the bound of Section 6.4.1.

Corollary 6.2. *For every integer $m > 0$, the following inequality holds*

$$\frac{\binom{K}{K\gamma+1} - \binom{K-m}{K\gamma+1}}{\binom{K}{K\gamma}} \leq m(1-\gamma). \tag{6.28}$$

Proof. We begin by proving the following inequality

$$\frac{\binom{K-n-1}{K\gamma}}{\binom{K}{K\gamma}} \leq (1-\gamma), \quad n \geq 0. \tag{6.29}$$

First, we can see that for $n = 0$, the inequality holds. Further, since for any $m > p$, we have that $\binom{K-m}{K\gamma} < \binom{K-p}{K\gamma}$, it follows that the inequality of Eq. (6.29) holds for any $n \geq 0$.

Further, in order to prove the inequality of Eq. (6.28) we will make use of Pascal's triangle, proof by induction and the result of Eq. (6.29). We begin by proving Eq. (6.28) for $m = 1$,

$$\frac{\binom{K}{K\gamma+1} - \binom{K-1}{K\gamma+1}}{\binom{K}{K\gamma}} = \frac{\binom{K}{K\gamma+1} - \frac{K-K\gamma-1}{K}\binom{K}{K-K\gamma-1}}{\binom{K}{K\gamma}} \tag{6.30}$$

$$= \frac{\binom{K}{K\gamma+1}}{\binom{K}{K\gamma}} \frac{K\gamma+1}{K} = (1-\gamma). \tag{6.31}$$

Now, let us assume that Eq. (6.28) holds for some $n \geq 1$. Then, we want to prove that it, also, holds for $n+1$.

$$\frac{\binom{K}{K\gamma+1} - \binom{K-n-1}{K\gamma+1}}{\binom{K}{K\gamma}} = \frac{\binom{K}{K\gamma+1} - \binom{K-n}{K\gamma+1} + \binom{K-n-1}{K\gamma}}{\binom{K}{K\gamma}} \tag{6.32}$$

$$= \frac{\binom{K}{K\gamma+1} - \binom{K-n}{K\gamma+1}}{\binom{K}{K\gamma}} + \frac{\binom{K-n-1}{K\gamma}}{\binom{K}{K\gamma}} \tag{6.33}$$

$$\leq n(1-\gamma) + (1-\gamma) \tag{6.34}$$

where in Eq. (6.32) we used the equality from Pascal's triangle and, then in Eq. (6.34) we used the inequality of Eq. (6.28). □

Chapter 7

Partially Connected Networks

In this chapter we will change the focus from the fully connected settings that we considered so far to partially connected settings, where the connection between a subset of the transmitters and a subset of the receivers could be either non-existent or very weak.

The partially connected settings present very interesting scenarios both DoF-wise and from a practical point of view. Specifically, from the perspective of performance, dealing with partially connected networks allows users to be naturally separated thus allowing for concurrent transmissions and thus for higher DoF. This advantage can further help achieve a much higher performance even if CSI is non-existent (we will explore such a setting in this chapter) or even achieve a much higher DoF performance using smaller amounts of CSI.

The second reason that justifies the design of algorithms for partially connected networks is their practical importance. While in the previous chapters we studied networks where all users are connected to the same set of transmitters or base station, in fact it is possible that many such networks would exist one close to another and that interference might form between transmitters of one network and receivers belonging in another. This observation dictates the use of interference management and, at the same time, caching to jointly help to alleviate this interference.

In this chapter we will study the following two different models.

- The first setting is based on Wyner's network, [110], which seeks to model cellular networks where sets of transmitter - receiver pairs are receiving interference from a nearby cell. For this model we will explore how transmitter cooperation (made possible by the use of a backhaul link and CSIT availability) and receiver side caching can form two complementary resources that elevate the setting's performance.

- The second setting will, again, assume sets of transmitter - receiver pairs, where a message would be receiver with full rate if coming from a corresponding transmitter and with lower rate if coming from any of the remaining transmitters. We explore the setting's performance when both transmitters and receivers are endowed with caches. This model

helps study interfering cells where the assumption of links being either strong or non-existent is relaxed.

Specifically, the first setting of interest, [111], is a variation of Wyner's network [110] according to which there are a set of K transmitters and a set of K receivers and where transmitter k is connected only to users k and $k+1$. According to the model that we will consider, the transmitters have individual access to the backhaul, through which they can fetch a limited amount of bits, and at the same time the receivers are equipped with caches (cf. Figure 7.1). Further, we will assume perfect knowledge of CSIT, which will allow for full transmitter cooperation.

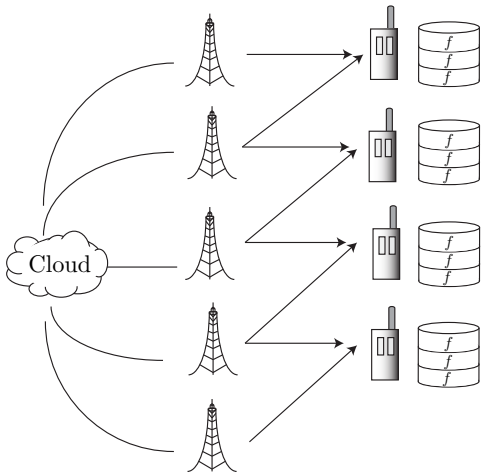

Figure 7.1: The Wyner's network where each transmitter is connected, via an individual link, to the backhaul from where it can fetch $M_T \cdot f$ bits. Each user is equipped with a cache of normalized size γ. Receiver $k+1$ can only receive messages from transmitters k and $k+1$.

The second setting that we will study [77], also considers the sets of K transmitters and K receivers where, now, both transmitters and receivers are equipped with caches (each transmitter can store fraction $\gamma_T \in \left[\frac{1}{K}, 1\right]$ of the library and each receiver is equipped with cache of normalized size $\gamma \in (0,1)$). Moreover, here, we relax the assumption that a transmitter is either connected or not connected to a user, by assuming that each user is connected via a full-rate link to its corresponding transmitter (achieving $\log P$ rate), while it can receive information from all other transmitters via weak links ($\alpha \cdot \log P$, $\alpha \leq 1$), (cf. Figure 7.2). Furthermore, transmitters have knowledge of only the statistical CSI thus, a challenge of this model is the effective transmission of content that resides at only distant transmitters, thus can only be communicated via weak links.

Design Challenge of algorithm

The main objective in the design of the algorithms is to combine the high DoF that are naturally provided by the partially connected settings, with the

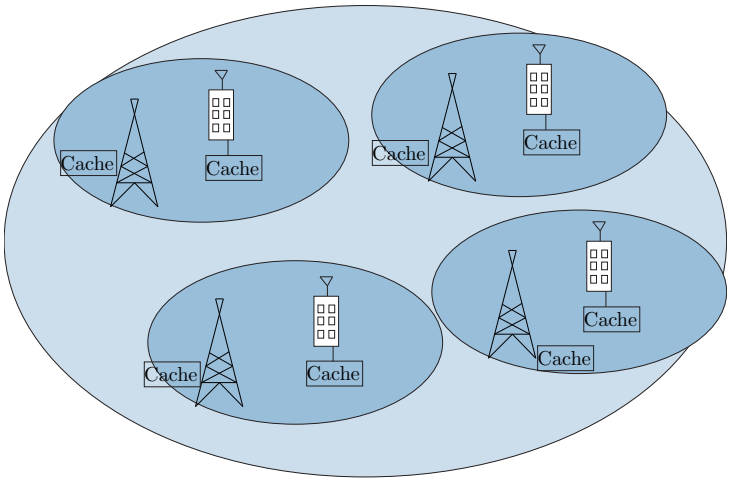

Figure 7.2: The no CSIT network of work [77], where a receiver is connected via a strong link to its respective transmitter and at the same time is receiving messages, via weaker links, from all other transmitters. Both transmitters and receivers are equipped with caches. A challenge of this setting is that CSI is not known at the transmitters, which reduces significantly the performance. Moreover, when requested user content resides exclusively at far-away transmitters, then it can only be communicated via weak links, which significantly degrades the performance.

DoF provided by caching content to the users.

It is important to note that the Coded Caching performance emerges when users are overhearing the same message, which conveys information to all these recipients. On the other hand, when considering that users are separated into L networks without any interference between them, we can see that performance would slightly increase (see [10]) from

$$T_1(K,\gamma) = \frac{K(1-\gamma)}{1+K\gamma} \tag{7.1}$$

to

$$T_L(K,\gamma) = \frac{K(1-\gamma)}{L+K\gamma}. \tag{7.2}$$

Now, in the scenarios that we consider, the users are partially connected and that creates a design challenge of transmitting messages that can be overheard by their intended recipients and a the same time to be kept away from unintended recipients.

Related work

The work in [112] considered a similar setting to ours and designed a caching and delivery policy that led to the characterization of the per user DoF in large Wyner's networks. However, the authors in [112] assumed that each

transmitter can only download from the backhaul messages associated to the receivers connected to it. We relax this restriction here, by *allowing transmitters to download any part of any file, as long as the backhaul constraint is respected* and we show that this added flexibility will lead to superior performance.

Further, the work in [113] considers a K-user partially connected interference network, where each receiver is connected to L transmitters with succeeding indices, and caching is enabled at both transmitters and receivers. Contrary to a transmitter-side caching approach, here the choice of downloaded content at each transmitter is based on receiver demands.

Similarly, the work in [114] considered cache-aided transmitters serving cache-aided receivers in a topological model, where each transmitter is connected to a subset of the receivers. The transmission process takes place under the absence of CSIT and the main goal is to design the caches so that the delivery time can be minimized.

The work in [115] considers the hexagonal cellular network comprized of C cells (one transmitter per cell) and K_R receivers per cell. Both transmitters and receivers are cache-enabled, while receivers are receiving interference from nearby cells and the goal was to characterize the DoF performance of this network.

7.1 Wyner's network on caches: Using caches to alleviate the backhaul

We assume a set of K transmitters, $\mathcal{K}_T \triangleq \{0, 1, ...K-1\}$, and a set of K receivers, $\mathcal{K}_R \triangleq \{0, 1, ...K-1\}$, where transmitter $k \in \mathcal{K}_T$ is connected to receivers k and $k+1$. The received signal at receiver $k+1$ takes the form

$$y_{k+1} = x_{k+1} + h_{k,k+1}x_k + w_{k+1}, \tag{7.3}$$

where $x_k \in \mathbb{C}$ denotes the transmitted signal from transmitter k, that satisfies the average power constraint $\mathbb{E}\{\|x_k\|^2\} \leq P$, $h_{k,k+1} \in \mathbb{C}$ denotes the channel realization between transmitter k and receiver $k+1$, while w_{k+1} corresponds to the channel noise, $w_{k+1} \sim \mathbb{C}\mathcal{N}(0,1)$.

Each receiver is equipped with a cache of normalized size γ. Transmitters are connected by individual links to the backhaul and can each fetch $M_T \cdot f$ bits. Upon complete delivery of all files, the considered performance metric is the asymptotic per-user Degrees of Freedom (puDoF) (i.e., the DoF normalized by the number of users in large networks), as defined in [116]. We use $d(M_T, \gamma)$ to denote the per-user DoF (puDoF) achieved with a backhaul load M_T and a fractional cache size γ. Further, we assume that transmitters know CSI with perfect accuracy and can cooperate.

We will use $[n]_k \triangleq n \mod k$ to denote the modulo operation.

7.1.1 Main Results

Theorem 7.1. *In the Wyner's network with per-transmitter maximum backhaul load $M_T \cdot f$ bits and no caches at the receivers, the per-user DoF $d(M_T, \gamma)$ for any $x \in \mathbb{N}$ satisfy the following:*

$$d\left(\frac{4x^2}{4x-1}, \gamma = 0\right) = \frac{4x-1}{4x}, \tag{7.4}$$

$$d\left(\frac{x+1}{2}, \gamma = 0\right) \geq \frac{2x}{2x+1}. \tag{7.5}$$

Proof. The proof of achievability is based on a modification of the schemes in [117] and [118], and is provided in Section 7.1.4. The converse of Eq. (7.4) follows from [117], where it was shown under an average backhaul load constraint. Since any scheme respecting a maximum load constraint is also respecting the average load constraint with the same value, it follows that the result is tight. □

Theorem 7.2. *In the Wyner's network with per-transmitter maximum backhaul load $M_T \cdot f$ bits and a per-user cache of normalized size γ, the per-user (interference-free) DoF of $d(M_T, \gamma) = 1$ can be achieved with the following pairs for any $x \in \mathbb{N}$:*

$$d\left(\frac{1-\gamma^2}{4\gamma}, \frac{1}{2x+1}\right) = 1, \tag{7.6}$$

$$d\left(\frac{1}{4\gamma}, \frac{1}{2x}\right) = 1. \tag{7.7}$$

Proof. The proof is constructive and presented in Section 7.1.2. □

Corollary 7.1. *Caching a fraction $\gamma = \frac{1}{4x}, x \in \mathbb{N}$ of the library at the receivers can increase the puDoF by an additive factor γ, while simultaneously decreasing the backhaul load by a multiplicative factor of $1-\gamma$.*

Proof. The proof makes use of the results from Eq. (7.4) and Eq. (7.7). Starting from a backhaul load of $M_T = \frac{4x^2}{4x-1}$, and adding a fractional cache-size $\gamma = \frac{1}{4x}$ at each receiver, the new backhaul load becomes

$$M_T' = \frac{4x-1}{4x}\frac{4x^2}{4x-1} = x. \tag{7.8}$$

We conclude the proof by observing through Eq. (7.7) that the pair $(M_T', \gamma) = (x, \frac{1}{4x})$ leads to achieving the full puDoF. □

Remark 7.1. *Observing the result in [112, Theorem 1], we can see that in order to achieve complete interference mitigation, it is required to have a backhaul load of $M_T = 2$ and cache size $\gamma = \frac{1}{6}$. On the contrary, here we can achieve the maximal puDoF with the backhaul - cache-size pairs $(M_T = 2, \gamma = \frac{1}{8})$ and $(M_T = \frac{3}{2}, \gamma = \frac{1}{6})$.*

The key factor enabling our result is that we allow for a more flexible backhaul load, instead of restricting each transmitter to download a specific set of messages which, in turn, allows to utilize transmitter cooperation more efficiently.

for $m \in \{1, ..., \frac{1-\gamma}{2\gamma}\}$ *(Choose a Delivery Network)* **do**
  for $p \in \{0, m\}$ *(Choose Slot of Delivery Net)* **do**
    Transmitter k : $0 \leq [k+p]_{2m} < m$ sends:

$$x_k = \sum_{i=0}^{[k]_{2m}} (-1)^i \prod_{j=k-i}^{k-1} h_{j,j+1} W^{r_{k-i}}_{[k+m-i]_S} \oplus W^{r_{k+m-i}}_{[k-i]_S}$$

    Transmitter k : $m \leq [k+p]_{2m} < 2m-1$ sends:

$$x_k = \sum_{i=[k+1]_m}^{m-1} (-1)^i \prod_{j=k-i}^{k-1} h_{j,j+1} W^{r_{k-i}}_{[k+m-i]_S} \oplus W^{r_{k+m-i}}_{[k-i]_S}$$

    Transmitter k : $[k+p]_{2m} = 2m-1$ sends:

$$x_k = \emptyset.$$

  end
end

Algorithm 7.1: Delivery Phase of the Cache-aided Scheme

7.1.2 Placement and Delivery of Files with Caching at the Receivers

In this section, we describe the scheme leading to the result of Theorem 7.2. We provide the proof of Eq. (7.6), i.e., when the cache size takes values $\gamma = \frac{1}{2x+1}$, $x \in \mathbb{N}$, while noting that Eq. (7.7) would follow by using memory sharing (cf. Section 7.1.6).

Placement Phase

In the placement phase, each file is subpacketized into $S = \frac{1}{\gamma}$ subfiles i.e., for every file W^n, $n \in [N]$ we have $W^n \to \{W^n_0, ..., W^n_{S-1}\}$.

Users cache according to

$$\mathcal{Z}_k = \left\{ W^n_{[k]_S}, \ \forall n \in n \in [N] \right\}. \tag{7.9}$$

Delivery Phase

As discussed above, the delivery phase starts with the request from each user of *any*[1] file from the library. For $\gamma = \frac{1}{2x+1}$, $x \in \mathbb{N}$, the goal is to rely on the smallest possible backhaul load that allows for interference-free reception i.e.

$$\min M_T \quad \text{s.t. } d(M_T, \gamma) = 1. \tag{7.10}$$

[1]We will assume that each user requests a different file, which corresponds to the worst case user demand.

The delivery phase consists of $2x$ transmission slots, where in each slot, we deliver fraction $\gamma = \frac{1}{2x+1}$ of the requested file to every receiver which, along with the cached fraction will amount to the whole file. We will call each successive pair of delivery slots a *Delivery Network* ($DN_m,\ m \in \{1,2,...,x\}$). Thus, there will be a total of x delivery networks. The role of DN_m is to deliver to user $k \in \mathcal{K}_R$ subfiles indexed by $[k \pm m]_S$. To this end, during DN_m the transmitted messages contain XORs (or linear combinations of XORs), where each XOR is formed using two subfiles with difference of indices equal to $[m]_S$. For example, in $DN_m,\ m \in \{1,...,\frac{1-\gamma}{2\gamma}\}$, the two transmitted XORs, intended for user $k \in \mathcal{K}_R$, will be

$$W^{r_k}_{[k+m]_S} \oplus W^{r_{k+m}}_{[k]_S}, \qquad W^{r_k}_{[k-m]_S} \oplus W^{r_{k-m}}_{[k]_S}.$$

Transmission takes place according to Algorithm 7.1. First, we demonstrate how the algorithm succeeds in achieving full puDoF through the following example and subsequently we discuss in detail the steps of Algorithm 7.1.

Example 7.1. *Let us assume that each user can store fraction $\gamma = \frac{1}{5}$ of the library, which corresponds to 4 transmission slots and thus 2 Delivery Networks, namely DN_1 and DN_2. We begin by subpacketizing each file into 5 subfiles and caching at each user according to Eq. (7.9), i.e.,*

$$\begin{aligned}
\mathcal{Z}_0 &= \{W^n_0,\ \forall n \in \{1,2,...,N\}\},\\
\mathcal{Z}_1 &= \{W^n_1,\ \forall n \in \{1,2,...,N\}\},\\
&\vdots\\
\mathcal{Z}_5 &= \{W^n_0,\ \forall n \in \{1,2,...,N\}\}.
\end{aligned}$$

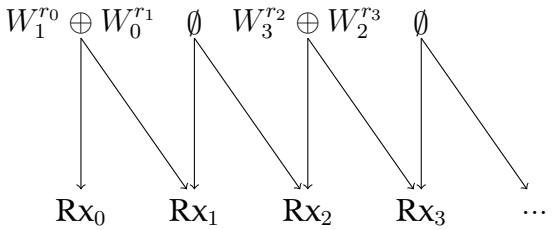

Figure 7.3: Slot 1 of Delivery Network DN_1, for the setting of Example 7.1.

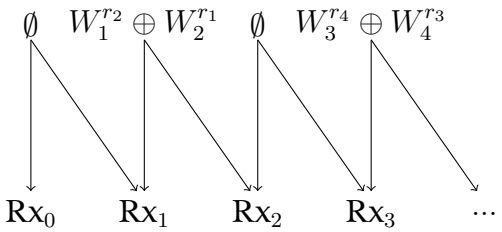

Figure 7.4: Slot 2 of Delivery Network DN_1, for the setting of Example 7.1.

After the request of a single file from each user, the transmission begins with DN_1 and then with DN_2. The first pair of transmission slots are responsible for delivering XORs comprized of subfiles with subsequent indices i.e., $W_1^{r_0} \oplus W_0^{r_1}$, $W_2^{r_1} \oplus W_1^{r_2}$, $W_3^{r_2} \oplus W_2^{r_3}$, $W_4^{r_3} \oplus W_3^{r_4}$ and so on. The transmitted messages at the first 4 transmitters during DN_1 are illustrated in Fig. 7.3-7.4.

The two slots of DN_2 follow after the completion of the two slots of Delivery Network DN_1. Here, the transmitters will communicate subfiles to user k with indices $[k \pm 2]_S$ i.e., $W_2^{r_0} \oplus W_0^{r_2}$, $W_3^{r_1} \oplus W_1^{r_3}$, $W_4^{r_2} \oplus W_2^{r_4}$, $W_0^{r_3} \oplus W_3^{r_5}$, and so on. The transmitted messages for each of the two slots are illustrated in Fig. 7.5-7.6.

In each of the 4 slots from Delivery Networks DN_1 and DN_2, a different subfile is delivered to each receiver, thus completing the delivery of all files[2], while downloading exactly 6 subfiles at each transmitter.

Details of the Delivery Algorithm First, a delivery network is chosen (Step 1), and then one of the two slots of the delivery network is chosen (Step 2). As discussed above, the purpose of delivery network DN_m is to deliver to each receiver $k \in \mathcal{K}_R$ subfiles indexed as $[k \pm m]_S$. During each transmission slot, the transmitters are divided into three non-overlapping sets. The first set (Line 3) is tasked with transmitting new messages and nulling the interference created by the messages of previous transmitters. The second set (Line 4) is tasked with transmitting messages that nullify the interference at their respective receiver, which interfering messages have been generated by transmitters of the first set. Finally, the third set (Line 5) of transmitters remains silent.

Characterizing the Required Backhaul Load

In this section, we will characterize the backhaul load that our algorithm requires in order to achieve interference-free transmission for a given fractional cache-size $\gamma = \frac{1}{2x+1},\ x \in \mathbb{N}$.

We begin by observing (cf. Algorithm 7.1 and Example 7.1) that the backhaul load at each transmitter during a specific Delivery Network is – potentially – different, and the two slots of a delivery network are designed to balance the per-transmitter backhaul load. As an example, in Figures 7.5-7.6 we can see that if a transmitter is silent during one slot of DN_2, then during the other slot it will transmit the linear combination of two XORs, thus will need to fetch from the backhaul 4 subfiles.

Consider a transmitter that, during Slot 2 of DN_m, is silent. This transmitter's index, k, (Line 5 of Algorithm 7.1) must satisfy $[k+m]_{2m} = 2m-1$, which gives $k = (2b-1)m-1,\ b \in \mathbb{N}$. This further means that during Slot 1 of DN_m, the transmitter's load will be characterized by Line 3 of Algorithm 7.1, since $[k]_{2m} = [2bm - m - 1]_{2m} = m-1$. Thus, this transmitter will need to fetch the contents of m XORs, making the total, per-delivery-network, back-

[2]We note that while W^{r_0} is not completely delivered, asymptotically that does not affect the per-user DoF of a large network.

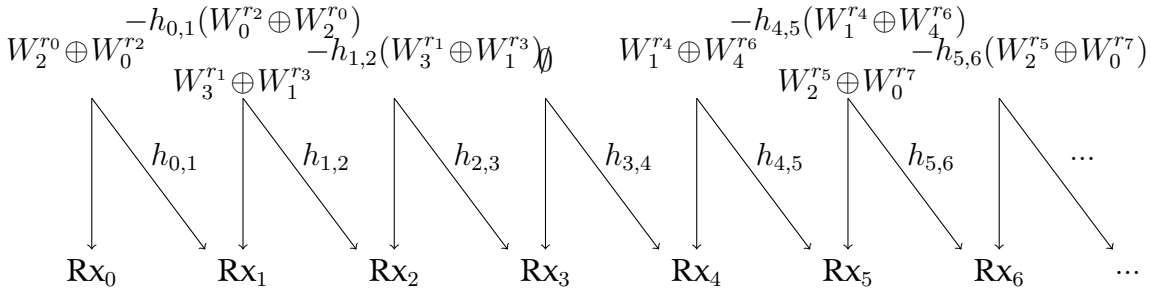

Figure 7.5: Slot 1 of Delivery Network DN_2, for the setting of Example 7.1.

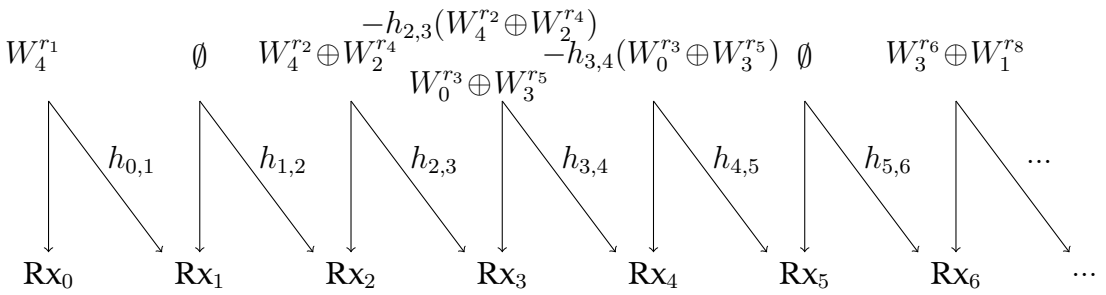

Figure 7.6: Slot 2 of Delivery Network DN_2, for the setting of Example 7.1.

haul load equal to $2m$ subfiles. Using this observation, we can calculate the overall required per-transmitter backhaul load, which is (note: $x = \frac{1-\gamma}{2\gamma}$)

$$M_T = \frac{1}{S} \cdot \sum_{m=1}^{x} 2m = \gamma \cdot 2 \cdot \frac{x(x+1)}{2} = \frac{1-\gamma^2}{4\gamma}.$$

7.1.3 Discussion and Concluding Remarks

From Corollary 7.1, we can deduce that receiver-side caching impacts the delivery time in three different ways.

Local Caching Gain Having stored fraction γ from each of the files, the system can have reductions in the delivery time since part of the desired content is already stored at the receivers and hence it is not required to be communicated.

Multicasting Gain Since messages contain XORed subfiles, in order to decode its desired subfile each receiver needs to make use of its cached but unwanted content. Thus, unwanted, cached content allows the transmission of more than one message simultaneously, which saves transmission slots.

Cooperative Transmission Gain As fraction γ of each file is cached at each receiver, the user will require only the smaller fraction $(1-\gamma)$ of the file. Now, for the same backhaul load as the no-caching case, this smaller request

(from 1 to $1-\gamma$) permits the transmitters to fetch more content, which can further boost the cooperation gains.

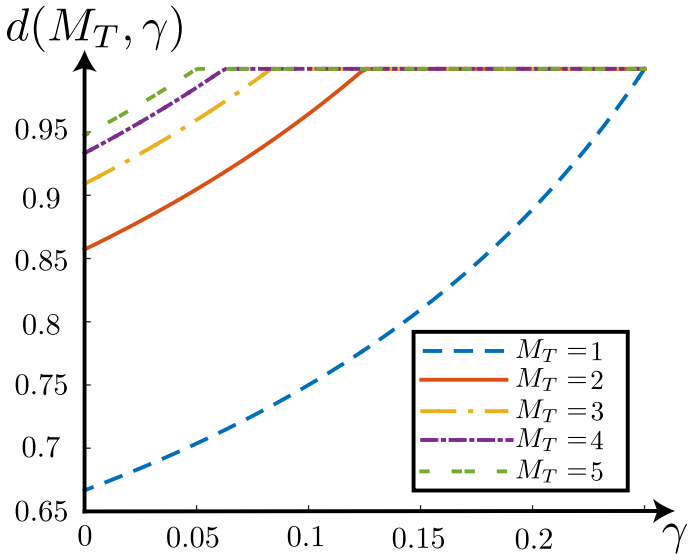

Figure 7.7: Per-user DoF as a function of cache size for different values of backhaul load.

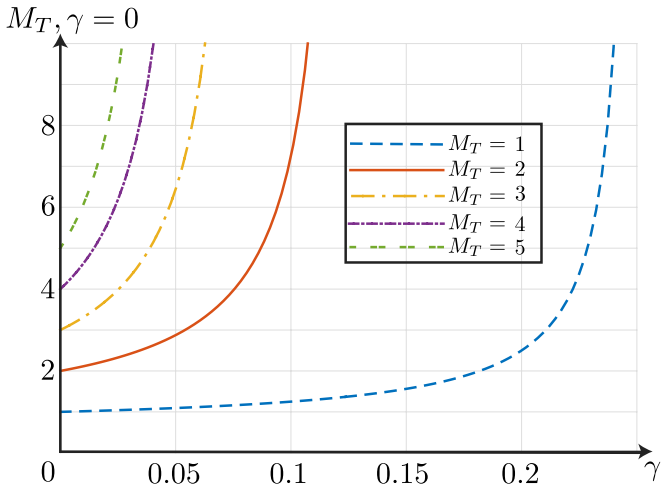

Figure 7.8: Required backhaul load without user caching that achieves the same per-user DoF as the cache-aided scheme with the pair (M_T, γ).

In Figure 7.7, we illustrate the above points b) and c) by plotting the puDoF that is achieved using different pairs (M_T, γ). It is interesting to note that a high backhaul load paired with a small cache can provide an intereference-free reception at every node.

Further, in Figure 7.8, we plot[3] the backhaul load that would have been

[3]The M_T values of the y-axis are calculated according to the results of Theorem 7.1. While not all of the points presented may be achievable, nevertheless their convex envelope is, and as a result present an even more optimistic case in favor of the no-caching schemes.

needed to achieve the same per-user DoF as does the pair (M_T, γ). We can note here that caching even fraction $\gamma = \frac{1}{20}$ with a backhaul load of $M_T = 3$ would have otherwise required a no caching backhaul load of $M_T = 6$. Moreover, caching fraction $\gamma = 0.1$ can reduce the load from $M_T \approx 7$ to $M_T = 2$. This further accentuates the role of coded transmissions and multicasting as relevant and impactful techniques that allow for fast delivery of content.

On the other hand, in the absence of caching, the cost of increasing the DoF even by a small fraction would have been extremely high. For example, if $M_T = \frac{16}{7}$ we know that we can achieve $d = \frac{7}{8}$, but in order to achieve $d' = \frac{15}{16}$, we would have to more than double the backhaul cost (see Eq. (7.4)). Contrarily, the same increase can be achieved by caching at each user an (approximate) fraction $\gamma = \frac{1}{16}$ of the library, and requiring a backhaul load of only $M_T = 2$.

7.1.4 No-Caching Schemes

In this section, we provide the achievable schemes, in the absence of caching, that prove Theorem 7.1. The presented schemes rely on applications of the schemes in [117] and [118] (See also [116] for a summary and high-level illustration) with time sharing, in order to meet the considered backhaul constraint.

Assume $M_T = \frac{4x^2}{4x-1}$, $x \in \mathbb{N}$.
Next(i) returns the smallest index of a subfile of W^{r_i} that has not been transmitted in a previous time slot.
Subpacketize each file into $S = 4x - 1$ subfiles.
for $t \in \{0, 1, ..., 4x-1\}$ *(Time Slots)* **do**
  Transmitter k : $0 \leq [k-t]_{4x} < 2x$ sends:

$$x_k = \sum_{i=0}^{[k-t]_{4x}} (-1)^i \left(\prod_{j=k-i}^{k-1} h_{j,j+1} \right) W^{r_{k-i}}_{\text{Next}(k-i)}.$$

  Transmitter k : $2x \leq [k-t]_{4x} < 4x-1$ sends:

$$x_k = \sum_{i=1}^{2x-L} (-1)^{i-1} \left(\prod_{j=k+1}^{k+i-1} \frac{1}{h_{j-1,j}} \right) W^{r_{k+i}}_{\text{Next}(k+i)},$$

  where $L = [k-t]_{4x} - 2x - 1$.
  Transmitter k : $[k-t]_{4x} = 4x - 1$ does not transmit.
end

Algorithm 7.2: Delivery Phase Under no Caching corresponding to the result of Eq. (7.4)

Proof of Theorem 7.1, Eq. (7.4)

The proposed scheme is completed in $4x$ blocks of communication. Each receiver gets $4x-1$ packets, and each packet is delivered through a one degree of freedom link. Each file is subpacketized into $4x-1$ subfiles, i.e. file $W^n \to \{W_0^n, W_1^n, ..., W_{4x-2}^n\}$. Each subfile is carried over one packet. In each block of communication, we divide the network into subnetworks, each consisting of $4x$ consecutive transmitter-receiver pairs, and use the scheme in [117] to deliver $4x-1$ packets in each subnetwork.

In what follows, we explain the scheme for the case when $x=1$ for simplicity, and demonstrate how it generalizes to larger values of x. For the first block of communication when $x=1$, the first transmitter downloads $W_0^{r_0}$, the second transmitter downloads $W_0^{r_0}$ and $W_0^{r_1}$, and the third downloads $W_0^{r_3}$. All three subfiles $W_0^{r_0}, W_0^{r_1}$ and $W_0^{r_3}$ can then be delivered through one DoF links using cooperative transmission, as illustrated in [117]. The fourth transmitter is inactive, thereby eliminating inter-subnetwork interference, and thus allowing the same scheme to be applied to each remaining subnetwork.

In the second block of communication, the first transmitter is inactive, while the same scheme is applied while we allocate to the second transmitter the role that the first transmitter had in the network, to the third transmitter the role that the second transmitter had and so on. More precisely, the first subnetwork would now consist of users with indices $\{1,2,3,4\}$. Subfile $W_1^{r_1}$ would then be downloaded by transmitters 1 and 2, $W_0^{r_2}$ would be downloaded by transmitter 2, and $W_1^{r_4}$ would be downloaded by transmitter 3, while the fifth transmitter is deactivated. Since transmitter 5 is inactive, then there is no inter-subnetwork interference, hence the same scheme can be applied to the subnetwork containing users $\{5,6,7,8\}$ to deliver subfiles $\{W_1^{r_5}, W_0^{r_6}, W_1^{r_8}\}$ without causing interference to the following subnetwork, and similarly, three subfiles can be delivered over one DoF links for every subsequent subnetwork. Proceeding in a similar fashion as above for the third block of communication for the case when $x=1$, we are able to deliver all $4x-1=3$ subfiles of each file W^i for almost all files[4] in the $4x=4$ communication blocks. The achieved puDoF would then be given by $\frac{4x-1}{4x}=\frac{3}{4}$. For the backhaul load, in each block of communication, each transmitter downloads an average of x subfiles. Over the $4x$ communication blocks, each transmitter downloads $4x^2$ subfiles, resulting in $M_T=\frac{4x^2}{4x-1}$ files.

7.1.5 Proof of Theorem 7.1, Eq. (7.5)

We explain in this section how the scheme presented in [118] can be modified to prove the result in Theorem 7.1, Eq. (7.5). The key idea is to employ time sharing for the scheme that achieves the same puDoF when the backhaul allows for distributing a message to a maximum of x transmitters.

[4] In fact, we deliver all files $W_i^{r_k}$, whose index $i \geq 4x-1$, but since the focus is on the asymptotic puDoF, then ignoring a small set of users would not affect the result.

Assume $M_T = \frac{x+1}{2},\ x \in \mathbb{N}$.
Next(i) returns the smallest index of a subfile of W^{r_i} that has not been transmitted in a previous time slot.
Subpacketize each file into $S = 2x$ subfiles.
for $t \in \{0, 1, ..., 2x\}$ *(Time Slots)* **do**
  Transmitter k : $0 \leq [k-t]_{2x+1} < x$ sends:

$$x_k = \sum_{i=0}^{[k-t]_{2x+1}} (-1)^i \left(\prod_{j=k-i}^{k-1} h_{j,j+1} \right) W^{r_{k-i}}_{\text{Next}(k-i)}.$$

  Transmitter k : $x \leq [k-t]_{2x+1} < 2x$ sends:

$$x_k = \sum_{i=1}^{x-L} (-1)^{i-1} \left(\prod_{j=k+1}^{k+i-1} \frac{1}{h_{j-1,j}} \right) W^{r_{k+i}}_{\text{Next}(k+i)},$$

  where $L = [k-t]_{2x+1} - x$.
  Transmitter k : $[k-t]_{2x+1} = 2x$ does not transmit.
end

Algorithm 7.3: Delivery Phase Under no Caching corresponding to the result of Eq. (7.5)

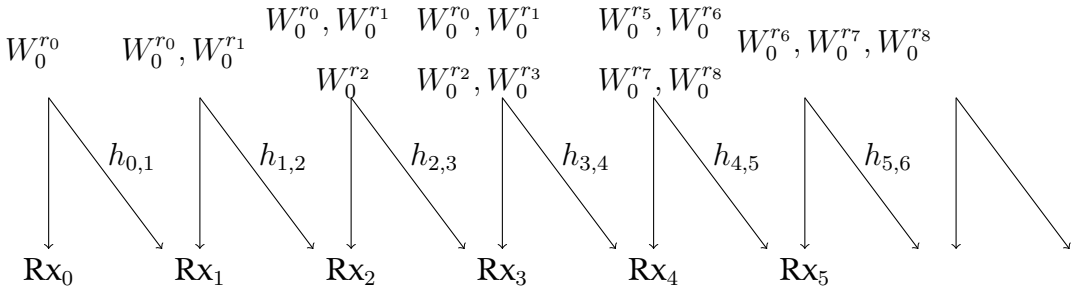

Figure 7.9: One transmission slot in the non-cache-aided network with backhaul constraint of $M_T = \frac{5}{2}$ corresponding to Eq. (7.5). This slot involves a subnetwork of 9 users and delivers 8 packets (to all users apart from the 4^{th}), thus achieving a pudDoF of $d\left(\frac{5}{2}, 0\right) = \frac{8}{9}$.

Similarly to the above proof, the proposed scheme completes in $2x+1$ communication blocks, and each file is subpacketized into $2x$ subfiles, and each subfile is delivered through a one DoF link. In each communication block, the network is split into subnetworks, where each has $2x+1$ consecutive transmitter-receiver pairs.

Consider the case where $x = 2$. In the proposed scheme, subfile $W_0^{r_0}$ is communicated between the first transmitter-receiver pair with no interference. The same subfile is also downloaded by the second transmitter (with index 1) to cancel its interference at receiver 1. Subfile $W_0^{r_1}$ is then delivered through transmitter 1 to receiver 1. Similarly, transmitter 3 delivers $W_0^{r_4}$ to the last

receiver in the first subnetwork, and transmitter 2 downloads the same subfile to cancel its interference at receiver 3. Finally, transmitter 2 delivers $W_0^{r_3}$ to receiver 3 with no interference. Note that $W_0^{r_3}$ is not transmitted in the first block of communication. Also, the last transmitter in the subnetwork is inactive to eliminate inter-subnetwork interference.

In each communication block, a total of $x(x+1)$ subfiles are downloaded from the backhaul for each subnetwork of $2x+1$ users. Upon the conclusion of all $2x+1$ communication blocks, each transmitter has downloaded an equal number of subfiles from the backhaul. Since each file has $2x$ subfiles, the per-transmitter backhaul load M_T is given by,

$$M_T = \frac{x(x+1)}{2x} = \frac{x+1}{2}. \tag{7.11}$$

7.1.6 Memory Sharing

In this section, we describe the memory sharing concept, which we first use to prove the result of Theorem 7.2, Eq. (7.7), i.e, the required backhaul load under the assumption of complete interference mitigation and fractional cache size $\gamma = \frac{1}{2k}$, and we further use this result to calculate the puDoF of any pair $\left(M_T \in \mathbb{N}, \gamma < \frac{1}{4M_T}\right)$.

The main idea of memory sharing is to split each file into two parts and cache from each part in an uneven manner. We begin by splitting each file W^n, $n \in [N]$ into parts i.e., $W^n \to \{W^{n,1}, W^{n,2}\}$ with respective sizes $|W^{n,1}| = p \cdot |W^n|$ and $|W^{n,2}| = (1-p) \cdot |W^n|$, where $p \in [0,1]$. We proceed to cache, at each user, fraction $\gamma_1 = \frac{1}{2x-1}$ from the first part of each file and fraction $\gamma_2 = \frac{1}{2x+1}$ from the second part of each file, which means that the cache constraint must satisfy

$$\begin{aligned} \gamma &= p \cdot \gamma_1 + (1-p) \cdot \gamma_2, \\ \frac{1}{2x} &= p\frac{1}{2x-1} + (1-p)\frac{1}{2x+1}, \\ p &= \frac{2x-1}{4x}. \end{aligned} \tag{7.12}$$

Then, using the result of Eq. (7.6), for each of the two parts, we can calculate the total required backhaul load as

$$\begin{aligned} M_T &= p\frac{1-\gamma_1^2}{4\gamma_1} + (1-p)\frac{1-\gamma_2^2}{4\gamma_2} \\ &= \frac{2x-1}{4x}\frac{4(2x)(2x-2)}{2x-1} + \frac{2x+1}{4x}\frac{4(2x)(2x+2)}{2x+1} \\ &= 8x = \frac{1}{4\gamma}. \end{aligned} \tag{7.13}$$

□

Further, in order to calculate the puDoF for an arbitrary $\gamma < \frac{1}{4M_T}$ that is paired with an integer-valued backhaul load, $M_T \in \mathbb{N}$, we follow the same

procedure of splitting the file into two parts, where now the fractional cache sizes chosen for each part take the values $\gamma_1 = \frac{1}{4M_T}$ and $\gamma_2 = 0$, respectively, thus p can be computed by solving

$$\gamma = p\gamma_1 + (1-p)\gamma_2 \Rightarrow p = 4\gamma M_T, \tag{7.14}$$

so that the memory cache constraint is respected. The puDoF when we transmit each part is going to be, respectively, $d(M_T, \gamma_1) = 1$ (cf. Eq. (7.7)) and $d(M_T, 0) = \frac{4M_T-2}{4M_T-1}$ (cf. Eq. (7.5)), thus the time required to serve all demands would be

$$T(M_T, \gamma) = p\frac{1-\gamma_1}{d(M_T, \gamma_1)} + (1-p)\frac{1-\gamma_2}{d(M_T, 0)},$$

from which we can calculate the achievable per-user DoF as

$$d\left(M_T, \gamma\right) = \frac{1-\gamma}{T(M_T, \gamma)}.$$

7.2 Transmitter Cooperation with No CSIT: Weak Interference meets Caching

In the second setting we consider a scenario with K transmitter/receiver pairs, where each receiver $k \in [K]$ is connected to transmitter k via a strong direct link with unit-normalized capacity, while the cross links from all other transmitters are weaker, with capacity $\alpha \leq 1$. In this setting, each transmitter and receiver are equipped with a cache of normalized sizes γ_T and γ_R, respectively, and it is assumed that transmitters do not have access to CSI.

The received signal at user $k \in [K]$, takes the form

$$y_k(t) = \sqrt{P}h_{k,k}(t)x_k(t) + \sum_{k' \in [K]\setminus\{k\}} \sqrt{P^\alpha}h_{k',k}x_{k'}(t) + w_k(t) \tag{7.15}$$

where an input signal $x_i(t)$ from transmitter i satisfies the power constraint $\mathbb{E}\{|x_i(t)|^2\} \leq 1$, where the channel fading coefficient from transmitter i to receiver k ($i, k \in [K]$) is denoted by $h_{i,k}(t)$, and where noise $w_k(t) \sim \mathbb{CN}(0,1)$. The average received signal-to-noise ratio (SNR) from any link to user k is, thus, calculated as

$$\mathbb{E}\left\{|\sqrt{P}h_{i,k}(t)x_i(t)|^2\right\} = \begin{cases} P, & i = k \\ P^\alpha, & i \neq k. \end{cases}$$

We note that the channel strength model follows the GDoF framework which we formally present in Chapter 6 (see also [45, 46, 49]).

7.2.1 Coding challenge

This setting brings about interesting challenges but also opportunities. The lack of CSIT is naturally non-beneficial, while the topological factor can be used to counter the lack of CSIT by (occasionally) reducing the interference power. At the same time though, this topological factor, together with the requirement that the transmitters do not receive any additional data after the caching phase, will inevitably force some of the receivers' requested data to reside only at far-away (weak) transmitters. Such 'cross' interference settings have notoriously bad performance, which will be here boosted by using receiver-side (coded) caching and partial cooperation among the transmitters. Hence our main challenge will be to place content in a way that best allows for topology, caching and transmitter cooperation to jointly remove unwanted interference. In the end, the schemes will combine Coded Caching with rate-splitting (Han-Kobayashi) approaches [119], and with interference enhancement techniques [120].

Some Intuition By carefully combining the above three techniques we want to ameliorate the performance loss due to the lack of CSIT. Specifically, the rate-splitting approach will allow for a partition of a message into "private" (low powered) and "common" (high powered) parts. The low powered part of the message will be "heard" only by one receiver (strongly connected) while the high powered part will be heard by all. Further, we will take advantage of content replicated at the transmitter side to bring forth interference enhancement, [120], by transmitting interference over the private part of the message, which is then used as side information to allow for interference removal at the receiver side. Finally, receiver side caching will be used as another source of side information that can help reduce interference.

7.2.2 Main Results

We first proceed with the GDoF performance of the case where each transmitter has access to the whole library, namely the topological MISO BC.

Theorem 7.3. *In the K-user topological MISO BC with parameter α and receiver-side caches of normalized size γ, the cache-aided GDoF here take the form*

$$D_\alpha(K, \gamma_T = 1, \gamma_R) = K(1-\alpha) + (K\gamma_R + 1)\alpha. \tag{7.16}$$

Theorem 7.4. *In the K-user topological MISO BC (parameter α) and receiver-side caches of normalized size γ, the cache-aided GDoF performance of*

$$D_\alpha(K, \gamma_T = 1, \gamma_R) = K(1-\alpha) + (K\gamma_R + 1)\alpha \tag{7.17}$$

is order optimal with gap of at most 12 from the optimal performance.

Proof. The order optimality is proven in [78]. □

Theorem 7.5. *In the K-transmitter K-receiver setting, with topological parameter α where each transmitter has cache of normalized size γ_T, while the receivers don't have caches, the achievable delay takes the form*

$$T_\alpha(K,\gamma_T) = \frac{K\gamma_T}{K(1-\tau)+\left(\min\{\frac{2\tau-1}{1-\gamma_T},\tau\}\right)^+} + \frac{x_s K(1-\gamma_T)}{\min\left\{\frac{\tau}{1-\gamma_T},1\right\}}.$$

where

$$x_s = \frac{K(1-\tau)}{K(1-\tau)+\left((K\gamma_R+1)\min\left\{\frac{2\tau-1}{1-\gamma_T},\tau\right\}\right)^+}$$

$$K_g = \frac{K(1-\gamma_T)\binom{K-1}{K\gamma_R}}{\binom{K}{K\gamma_R+1}-\gamma_T\binom{K-1}{K\gamma_R}-\binom{K\gamma_T}{K\gamma_R+1}\frac{K-K\gamma_R-1}{K}}.$$

Theorem 7.6. *In the K-transmitter K-receiver setting, with topological parameter α where each transmitter has cache of normalized size γ_T and each receiver has normalized cache-size γ_R, the achievable delay takes the form*

$$T_\alpha(K,\gamma_T \le \gamma_R) = \frac{K(1-\gamma_R)}{(K\gamma_R+1)\min\left\{\frac{\tau}{1-\gamma_T},1\right\}}$$

$$T_\alpha(K,\gamma_T > \gamma_R) = \frac{K\gamma_T(1-\gamma_R)}{K(1-\tau)+(K\gamma_R+1)\left(\min\left\{\frac{2\tau-1}{1-\gamma_T},\tau\right\}\right)^+} + \frac{K(1-\gamma_T)(1-\gamma_R)}{K_g\min\left\{\frac{\tau}{1-\gamma_T+},1\right\}}\cdot x_s$$

where x_s and K_g defined in Theorem 7.5.

7.2.3 Placement and Delivery Schemes

Cache Placement

Files are split into

$$S = \binom{K}{K\gamma_T}\binom{K}{K\gamma_R} \tag{7.18}$$

subfiles i.e.,

$$W^n \to \{W^n_{\tau_T,\tau_R}, \tau_T \subset [K], |\tau_T| = K\gamma_T, \tau_R \subset [K], |\tau_R| = K\gamma_R\}. \tag{7.19}$$

Placement at transmitter k_T and receiver k_R takes the form

$$\mathcal{Z}_{k_T} = \{W^n_{\tau_T,\tau_R} : k_T \in \tau_T,\ \forall\tau_R, \forall n \in [N]\}, \tag{7.20}$$

$$\mathcal{Z}_{k_R} = \{W^n_{\tau_T,\tau_R} : k_R \in \tau_R,\ \forall\tau_T, \forall n \in [N]\}. \tag{7.21}$$

Delivery Phase

The delivery scheme is dependent on parameters α and γ_T and is described here for the four distinct cases, namely

- The MISO BC case where each transmitter cache has access to the entire library,
- the interference scenario where $\gamma_R < \gamma_T < 1$, and $\alpha \leq \frac{1}{2}$
- the interference scenario where $\gamma_R < \gamma_T < 1$ and $\alpha \geq \frac{1}{2}$, and
- the interference scenario where $\gamma_T \leq \gamma_R$.

MISO BC with receiver side caching In this case, as discussed before, each transmitter has access to the whole library which means that the transmitter index takes the form $\tau_T = [K]$ thus, in order to simplify the notation we will abstain of using this index.

We begin by dividing a subfile into two parts,

$$W^n \to \{W^{n,1}_{\tau_R}, W^{n,2}_{\tau_R}\} \tag{7.22}$$

with respective relative sizes p and $1-p$, where

$$p = \frac{|W^{n,1}_{\tau_R}|}{|W^n_{\tau_R}|} = \frac{(K\gamma_R + 1)\alpha}{K(1-\alpha) + (K\gamma_R + 1)\alpha}. \tag{7.23}$$

The transmitted message from transmitter $k \in [K]$ will be of the form

$$x_k = \underbrace{c_k}_{\text{common}} + \underbrace{b_k}_{\text{private}}$$

where c has power $1 - P^{-\alpha}$ and rate α, while b_k, has power $P^{-\alpha}$ and rate $1-\alpha$. Each transmitter $k \in [K]$ encodes in b_k from any requested subfile its part 2, i.e.

$$b_k \leftarrow \{W^{d_k,2}_{\tau_R}, \forall \tau_R \ : \ k \notin \tau_R\}.$$

Moreover, the high powered, common, part of the message is encoded with all the XORs which are formed as a combination of $K\gamma_R + 1$ elements from the set of $W^{n,1}_{\tau_R}$ subfiles, i.e.

$$c_k \leftarrow \left\{ \bigoplus_{m \in \sigma_R} W^{d_m,1}_{\sigma_R \setminus \{m\}}, \ \forall \sigma_R \subseteq [K], |\sigma_R| = K\gamma_R + 1 \right\}.$$

Note 7.1. *We can notice that common message c_k is the same throughout all the transmitters, while private messages b_k are transmitter dependent.*

The above separation of transmitted messages into two parts (one with high power and one with low power) "re-shapes" the channel to act as the equivalent to a set of two separate channels, where the first channel conveys a common

message that can be overheard by all users with rate $(1-\alpha)$*, while the second acts as a set of* K *individual links each from a transmitter to its respective receiver and each carrying messages the rate of* α*.*

This observation along with the multicasting opportunities of Coded Caching allows to use the common part of the channel to serve many users at a time, by exploiting the user caches, while also continue to exploit the separation, and scalable rate, that the private part of the channel offers.

Decoding The received signal at user k takes the form

$$y_k = \underbrace{h_{k,k}c_k}_{P-P^{1-\alpha},\alpha} + \sum_{i\neq k} \underbrace{h_{i,k}c_i}_{P^{1-\alpha},\alpha} + \underbrace{h_{k,k}b_k}_{P^{\alpha},1-\alpha} + \sum_{i\neq k} \underbrace{h_{i,k}b_i}_{P^{0},1-\alpha}. \tag{7.24}$$

User $k \in [K]$ first decodes message c_k by treating interfering messages as noise (TIN), and does so with rate α. Then, proceeds to remove all common messages $c_i,\ i \in [K]\setminus\{k\}$ from Eq. (7.24). Further, by TIN the receiver can decode the respective private (lower powered) message with rate $1-\alpha$.

For the above, the delivery time achieved is

$$\begin{aligned} T_\alpha(K,\gamma_T,\gamma_R) &= \max\left\{(1-p)\frac{\binom{K}{K\gamma_T}\binom{K-1}{K\gamma_R}}{\binom{K}{K\gamma_T}\binom{K}{K\gamma_R}(1-\alpha)}, p\frac{\binom{K}{K\gamma_T}\binom{K}{K\gamma_R+1}}{\binom{K}{K\gamma_T}\binom{K}{K\gamma_R}\alpha}\right\} \\ &= \frac{K(1-\gamma_R)}{K(1-\alpha)+(K\gamma_R+1)\alpha}, \end{aligned}$$

which implies the cache-aided GDoF performance of

$$D_\alpha(K,\gamma_T=1,\gamma_R) = K(1-\alpha)+(K\gamma_R+1)\alpha. \tag{7.25}$$

Interference Channel with $\alpha \leq \frac{1}{2}$ and $\gamma_T > \gamma_R$

In this case, the delivery phase is split into two sub-phases. During the first sub-phase, each transmitter k will deliver all their cached subfiles that are requested by its respective (direct) receiver (receiver k). Each transmitter sends its message with power 1 and rate $1-\alpha$, while receivers decode the message by treating all other messages as noise, thus with rate $1-\alpha$. Specifically, a transmitted signal from transmitter k contains:

$$x_k \leftarrow \left\{W^{d_k}_{\tau_T,\tau_R} \ :\ k\in\tau_T \ \&\ k\notin\tau_R\right\}.$$

Users are able to decode their messages by treating interfering messages as noise.

The duration of this sub-phase is

$$T^1_\alpha(K,\gamma_T,\gamma_R) = \frac{\binom{K-1}{K\gamma_T-1}\binom{K-1}{K\gamma_R}}{(1-\alpha)\binom{K}{K\gamma_T}\binom{K}{K\gamma_R}} = \frac{\gamma_T(1-\gamma_R)}{1-\alpha}.$$

In essence, the above sub-phase takes advantage of the fact that interference from far-away transmitters is weak ($\alpha \leq \frac{1}{2}$) to deliver a part of the requested subfile using the direct links.

Further, the role of the second sub-phase is to deliver the remaining part of the files, which reside in far-away transmitters. Each transmitter forms all XORs of size $K\gamma_R + 1$ apart from those whose *all* $K\gamma_R + 1$ elements have already been transmitted. In the transmitted signal we encode the following subfiles

$$x_k \leftarrow \Bigg\{ \bigoplus_{m \in \sigma_R} W^{d_m}_{\tau_T, \sigma_R \setminus \{m\}}, \quad \forall\ \tau_T \subset [K], |\tau_T| = K\gamma_T, k \in \tau_T, \\ \forall \sigma_R \in \mathcal{R}_1 \setminus \mathcal{R}_2, \\ \mathcal{R}_1 = \{\rho_1 \subset [K],\ |\rho_1| = K\gamma_R + 1\} \\ \mathcal{R}_2 = \{\rho_2 \subseteq \tau_T,\ |\rho_2| = K\gamma_R + 1\} \Bigg\}.$$

i.e. in the transmitted signals we encode all possible XORs of size $K\gamma_R + 1$, apart from those whose all $(K\gamma_R + 1)$ elements have been transmitted in delivery subphase 1.

Moreover, we can see that some XORs have been partially transmitted in subphase 1, something that allows for interference removal opportunities.

Example 7.2. *As an example, let us assume that $K = 4$, that each transmitter caches fraction $\gamma_T = \frac{1}{2}$ of the library and that each receiver has a cache of normalized size $\gamma_R = \frac{1}{4}$. In this setting, we can see that receiver 1 would receive subfile $W^{d_1}_{12,4}$ in subphase 1.*

At the same time, this subfile would appear in subphase 2 encoded into XOR $W^{d_1}_{12,4} \oplus W^{d_4}_{12,1}$. While this XOR is useful for receiver 4, thus it needs to be transmitted since it conveys information that has not appeared in sub-phase 1, nevertheless it carries no useful information for receiver 1, and who can fully "cache-it-out" so that to reduce its experienced interference.

In total, each XOR will be transmitted from $K\gamma_T$ transmitters. Each message is transmitted with power 1 and encoded with *all* the above XORs, while the rate of each XOR is

$$r_{\text{XOR}} = \frac{\min\left\{\frac{\alpha}{1-\gamma_T}, 1\right\}}{\binom{K}{K\gamma_T}\left[\binom{K}{K\gamma_R+1} - \gamma_T\binom{K-1}{K\gamma_R} - \binom{K\gamma_T}{K\gamma_R+1}\frac{K-K\gamma_R-1}{K}\right]}$$

which accounts for all the possible XORs $\binom{K}{K\gamma_R+1}$, minus those that have been partially transmitted ($\gamma_T\binom{K-1}{K\gamma_R}$), (see Example 7.2), thus no longer causing interference, and minus those that have been fully transmitted in sub-phase 1 ($\binom{K\gamma_T}{K\gamma_R+1}$).

We can see that the total rate of each transmitted message is

$$r_{x_k} = \frac{\binom{K-1}{K\gamma_T-1}\left[\binom{K}{K\gamma_R+1} - \binom{K\gamma_T}{K\gamma_R+1}\right]\min\left\{\frac{\alpha}{1-\gamma_T},1\right\}}{\binom{K}{K\gamma_T}\left[\binom{K}{K\gamma_R+1} - \gamma_T\binom{K-1}{K\gamma_R} - \binom{K\gamma_T}{K\gamma_R+1}\frac{K-K\gamma_R-1}{K}\right]}$$
$$= \frac{\gamma_T\left[\binom{K}{K\gamma_R+1} - \binom{K\gamma_T}{K\gamma_R+1}\right]\min\left\{\frac{\alpha}{1-\gamma_T},1\right\}}{\left[\binom{K}{K\gamma_R+1} - \gamma_T\binom{K-1}{K\gamma_R} - \binom{K\gamma_T}{K\gamma_R+1}\frac{K-K\gamma_R-1}{K}\right]}.$$

By removing the partially transmitted XORs of subphase 1, from each x_i message, the remaining rate of any x_i is $r_{x_i} = \gamma_T \min\{\frac{\alpha}{1-\gamma_T}, 1\}$.

As evident, there appear two cases, depending on parameters γ_T and α. If $\alpha \leq 1-\gamma_T$, Rx_k decodes x_k by TIN, with rate

$$\gamma_T \frac{\alpha}{1-\gamma_T} \leq 1-\alpha \tag{7.26}$$

and proceeds to remove its contents from all the received signals. Since the XORs appearing in message x_k will also be transmitted from some other $K\gamma_T - 1$ transmitters the remaining signal has sum-rate

$$\sum_{i\in[K]} r_{x_i} - r_{x_k} = \frac{\alpha}{1-\gamma_T} - \gamma_T\frac{\alpha}{1-\gamma_T} = \alpha$$

which makes them decodable by a joint decoding process.

On the other hand, if $\alpha \geq 1-\gamma_T$, then a receiver proceeds to jointly decode all $x_i, i \in [K]$ messages.

The time required to complete this sub-phase is

$$T_\alpha^{(2)}(K,\gamma_T,\gamma_R) = \frac{\binom{K-1}{K\gamma_T}\left[\binom{K}{K\gamma_R+1} - \binom{K\gamma_T}{K\gamma_R+1}\right]}{r_{x_k}\binom{K}{K\gamma_T}\binom{K}{K\gamma_R}} = \frac{K(1-\gamma_T)(1-\gamma_R)}{K_g \min\left\{\frac{\alpha}{1-\gamma_{T^+}},1\right\}}.$$

Interference scenario with $\alpha \geq \frac{1}{2}$ and $\gamma_T > \gamma_R$ In this case, similarly to the previous one, delivery consists of two sub-phases.

First, each subfile is split into two parts $W^{n,1}_{\tau_T,\tau_R}$ and $W^{n,2}_{\tau_T,\tau_R}$ with relative sizes $|W^{n,1}_{\tau_T,\tau_R}| = x_s \cdot |W^n_{\tau_T,\tau_R}|$ and $|W^{n,2}_{\tau_T,\tau_R}| = (1-x_s)\cdot|W^n_{\tau_T,\tau_R}|$ where

$$x_s = \frac{(K\gamma_R+1)\min\left\{\frac{2\alpha-1}{1-\gamma_T},\alpha\right\}}{(K\gamma_R+1)\min\left\{\frac{2\alpha-1}{1-\gamma_T},\alpha\right\} + K(1-\alpha)}.$$

During delivery sub-phase 1, a transmitted message is partitioned into two parts, namely the common part and the private part, as follows

$$x_k = \underbrace{c_k}_{1-P^{-\alpha},\alpha} + \underbrace{b_k}_{P^{-\alpha},1-\alpha}, \ k \in [K]$$

and where the content encoded in each such message takes the form

$$c_k \leftarrow \Bigg\{ \bigoplus_{m\in\sigma_R} W^{d_m,1}_{\tau_T,\sigma_R\setminus\{m\}},\ \ \forall \tau_T \subset [K],\ |\tau_T| = K\gamma_T :\ k\in\tau_T \\ \forall \sigma_R \subseteq [K],\ |\sigma_R| = K\gamma_R + 1 \Bigg\} \tag{7.27}$$

$$b_k \leftarrow \Bigg\{ W^{d_k,2}_{\tau_T,\tau_R},\quad \forall \tau_T \subset [K],\ |\tau_T| = K\gamma_T,\ k\in\tau_T \\ \forall \tau_R \subset [K],\ |\tau_R| = K\gamma_R\ :\ k\notin\tau_R \Bigg\}. \tag{7.28}$$

Thus, c_k is designed to carry all XORs composed composed of subfiles from set $\left\{W^{n,1}_{\tau_T,\tau_R},\ k\in\tau_T\right\}$, i.e. from subfiles cached at transmitter k. Further, b_k carries all subfiles from set $\left\{W^{n,2}_{\tau_T,\tau_R},\ k\in\tau_T,\ k\notin\tau_R\right\}$, i.e. subfiles of the 2nd partition which are cached by transmitter k but not cached by receiver k.

The required time for this subphase is

$$\begin{aligned} T^1_\alpha(K,\gamma_T,\gamma_R) &= \frac{1}{S}\cdot \max\left\{ \frac{(1-x_s)\binom{K-1}{K\gamma_T-1}\binom{K-1}{K\gamma_R}}{(1-\alpha)}, \frac{x_s\binom{K}{K\gamma_T}\binom{K}{K\gamma_R+1}}{\min\{\frac{\alpha}{1-\gamma_T},\alpha\}} \right\} \\ &= \frac{K\gamma_T(1-\gamma_R)}{K(1-\alpha)+(K\gamma_R+1)\min\left\{\frac{2\alpha-1}{1-\gamma_T},\alpha\right\}}. \end{aligned}$$

At the end of the above subphase, all $W^{n,1}_{\tau_T,\tau_R}$ requested subfiles have been delivered, while the remaining of the requested $W^{n,2}_{\tau_T,\tau_R}$ subfiles are to be send via sub-phase 2. Similarly to the previous paragraph's approach, we will send the rest of the subfiles via the low-rate channel, since these subfiles are residing only on far-away transmitters. The achieved delivery time for this sub-phase takes the form

$$T^2_\alpha(K,\gamma_T,\gamma_R) = \frac{K(1-\gamma_T)(1-\gamma_R)}{K_g \min\left\{\frac{\alpha}{1-\gamma_T+},1\right\}}(1-x_s).$$

Interference scenario with $\gamma_T < \gamma_R$ In this final case, the benefits of Coded Caching outperform the gain of sending some subfiles using the fast, direct channel. As a result, we will only focus on transmitting XORs using uniquely the low-rate channel. Each transmitter sends its message with full power, i.e. $P=1$, and total rate $\gamma_T \min\{\frac{\alpha}{1-\gamma_T},1\}$. In this message we encode XORs that are comprized of $K\gamma_R+1$ subfiles and which are all found at the specific transmitter i.e.,

$$x_k \leftarrow \Bigg\{ \bigoplus_{k_R\in\sigma_R} W^{d_k}_{\tau_T,\sigma_R\setminus\{k_R\}}\ :\ k\in\tau_T, \sigma_R\subseteq[K],\ |\sigma_R| = K\gamma_R+1 \Bigg\} \tag{7.29}$$

with each XOR having rate

$$r_{\text{XOR}} = \frac{\min\left\{\frac{\alpha}{1-\gamma_T}, 1\right\}}{\binom{K}{K\gamma_T}\binom{K}{K\gamma_R+1}}.$$

We can discern two regions of interest, namely $\alpha \lessgtr 1-\gamma_T$. In either region, the received message takes the form

$$y_k = h_{k,k}\sqrt{P}x_k + \sqrt{P^{-\alpha}}\sum_{i\neq k} h_{i,k}x_i + w_k. \tag{7.30}$$

If $\alpha \leq 1-\gamma_T$, then receiver k would first decode x_k by treating any other message as noise, which can be achieved with rate α. Then, proceeds to remove message x_k from Eq. (7.30) and proceed to jointly decode all $K-1$ remaining messages.

The delay for this second sub-phase takes the form

$$\begin{aligned} T_\alpha^{(2)}(K,\gamma_T,\gamma_R) &= \frac{\binom{K}{K\gamma_T}\binom{K}{K\gamma_R+1}}{\binom{K}{K\gamma_T}\binom{K}{K\gamma_R}\min\left\{\frac{\alpha}{1-\gamma_T}, 1\right\}} \\ &= \frac{K(1-\gamma_R)}{(K\gamma_R+1)\min\left\{\frac{\alpha}{1-\gamma_T}, 1\right\}}. \end{aligned}$$

7.2.4 Example

Let us consider the $K=4$ pair network with topology parameter $\alpha = \frac{6}{10}$ and normalized cache-sizes $\gamma_T = \frac{1}{2}$ and $\gamma_R = \frac{1}{4}$, respectively which corresponds to the above presented case 3.

The initial subpacketization is $S = \binom{4}{2}\binom{4}{1} = 24$ subfiles, while the receiver side caches are filled as

$$\mathcal{Z}_{k_R} = \{12,13,14,23,24,34\} \otimes \{k_R\},\ \ k_R \in [4]$$

and the transmitter caches are filled as

$$\begin{aligned} \mathcal{Z}_{t_1} &= \{12,13,14\} \otimes \{1,2,3,4\}, \\ \mathcal{Z}_{t_2} &= \{12,23,24\} \otimes \{1,2,3,4\}, \\ \mathcal{Z}_{t_3} &= \{13,23,34\} \otimes \{1,2,3,4\}, \\ \mathcal{Z}_{t_4} &= \{14,24,34\} \otimes \{1,2,3,4\}. \end{aligned}$$

Delivery Sub-phase 1 In the first delivery sub-phase, subfiles are split into two parts $W^{n,1}_{\tau_T,\tau_R}, W^{n,2}_{\tau_T,\tau_R}$, according to Eq. (7.27).

The transmitted signals take the form

$$x_k = \underbrace{c_k}_{1-P^{-\alpha},2\alpha-1} + \underbrace{b_k}_{P^{-\alpha},1-\alpha}. \tag{7.31}$$

Common messages have individual rate of $2\alpha-1$ and contain the following XORs

$$c_1 \leftarrow \big\{A^1_{\tau_T,2} \oplus B^1_{\tau_T,1},\ A^1_{\tau_T,3} \oplus C^1_{\tau_T,1},\ A^1_{\tau_T,4} \oplus D^1_{\tau_T,1},\ C^1_{\tau_T,2} \oplus B^1_{\tau_T,3}, D^1_{\tau_T,2} \oplus B^1_{\tau_T,4},\ C^1_{\tau_T,4} \oplus D^1_{\tau_T,3}\big\}, \forall \tau_T \in \{12,13,14\}$$

$$c_2 \leftarrow \big\{A^1_{\tau_T,2} \oplus B^1_{\tau_T,1},\ A^1_{\tau_T,3} \oplus C^1_{\tau_T,1},\ A^1_{\tau_T,4} \oplus D^1_{\tau_T,1},\ C^1_{\tau_T,2} \oplus B^1_{\tau_T,3}, D^1_{\tau_T,2} \oplus B^1_{\tau_T,4},\ C^1_{\tau_T,4} \oplus D^1_{\tau_T,3}\big\}, \forall \tau_T \in \{12,23,24\}$$

$$c_3 \leftarrow \big\{A^1_{\tau_T,2} \oplus B^1_{\tau_T,1},\ A^1_{\tau_T,3} \oplus C^1_{\tau_T,1}, A^1_{\tau_T,4} \oplus D^1_{\tau_T,1},\ C^1_{\tau_T,2} \oplus B^1_{\tau_T,3}, D^1_{\tau_T,2} \oplus B^1_{\tau_T,4}, C^1_{\tau_T,4} \oplus D^1_{\tau_T,3}\big\}, \forall \tau_T \in \{13,23,34\}$$

$$c_4 \leftarrow \big\{A^1_{\tau_T,2} \oplus B^1_{\tau_T,1},\ A^1_{\tau_T,3} \oplus C^1_{\tau_T,1}, A^1_{\tau_T,4} \oplus D^1_{\tau_T,1},\ C^1_{\tau_T,2} \oplus B^1_{\tau_T,3}, D^1_{\tau_T,2} \oplus B^1_{\tau_T,4}, C^1_{\tau_T,4} \oplus D^1_{\tau_T,3}\big\}, \forall \tau_T \in \{14,24,34\}$$

while the b_i part of the message contains

$$\begin{aligned} b_1 &\leftarrow A^2_{\tau_T,\tau_R}, \forall \tau_T \in \{12,13,14\}, \forall \tau_R \in \{2,3,4\} \\ b_2 &\leftarrow B^2_{\tau_T,\tau_R}, \forall \tau_T \in \{12,23,24\}, \forall \tau_R \in \{1,3,4\} \\ b_3 &\leftarrow C^2_{\tau_T,\tau_R}, \forall \tau_T \in \{13,23,34\}, \forall \tau_R \in \{1,2,4\} \\ b_4 &\leftarrow D^2_{\tau_T,\tau_R}, \forall \tau_T \in \{14,24,34\}, \forall \tau_R \in \{1,2,3\}. \end{aligned}$$

In order to see the decoding process, let us focus on user 1, who receives

$$y_1 = \underbrace{h_{11}c_1}_{P-P^{1-\alpha}} + \sum_{i=2}^{4} \underbrace{h_{i1}c_i}_{P^{\alpha}} + \underbrace{h_{11}b_1}_{P^{1-\alpha}} + \sum_{i=2}^{4} \underbrace{h_{i1}b_i}_{P^0}. \tag{7.32}$$

First, user 1 proceeds to decode message c_1 by treating all other messages as noise. Since its rate is $2\alpha - 1 < 1 - \alpha$ user 1 can decode the message successfully. Then, proceeds to remove c_1 from Eq. (7.32).

From the remaining common messages, i.e. $c_2 + c_3 + c_4$ what is left is half of all the XORs, which in total have sum rate $2\alpha - 1$. User 1 can jointly decode these remaining common messages by treating the private messages as noise and proceed to remove them from the received signal. Finally, user 1 can decode b_1 with rate $1 - \alpha$, by treating b_2, b_3, b_4 as noise.

Delivery Sub-phase 2 This phase aims to deliver the rest of the messages, i.e. the part 2 of a subfile which is cached only at far away transmitters, since part 1 that resides in close transmitters has been transmitted in the previous sub-phase.

Transmitters send their messages $x_k, k \in [K]$ each with power 1 and rate $1/4$. For example transmitter 1's message content is

$$\begin{aligned} x_1 \leftarrow \big\{ & A^2_{12,3} \oplus C^2_{12,1}, A^2_{12,4} \oplus D^2_{12,1}, C^2_{12,2} \oplus B^2_{12,3}, D^2_{12,2} \oplus B^2_{12,4}, \\ & C_{12,4} \oplus D^2_{12,3}, A^2_{13,2} \oplus B^2_{13,1}, A^2_{13,4} \oplus D^2_{13,1}, C^2_{13,2} \oplus B^2_{13,3}, \\ & D^2_{13,2} \oplus B^2_{13,4}, C^2_{13,4} \oplus D^2_{13,3}, A^2_{14,2} \oplus B^2_{14,1}, A^2_{14,3} \oplus C^2_{14,1}, \\ & C^2_{14,2} \oplus B^2_{14,3}, D^2_{14,2} \oplus B^2_{14,4}, C^2_{14,4} \oplus D^2_{14,3} \big\}. \end{aligned}$$

From the above XORs, receiver 1 removes all those that contain subfiles of file A_{τ_T,τ_R}, since all these subfiles $A_{\tau_T,\tau_R},\ 1 \in \tau_T$ have been previously delivered, while their XORed counterpart is cached at $\mathcal{Z}_1$ and then proceeds to jointly decode $x_i, i \in [K]$.

Chapter 8

The high-dimensionality Aspect of Distributed Computing

8.1 Introduction

In this chapter we will change the focus from caching to distributed computing showing first, how the intuition from Coded Caching has been used to speed up the execution of MapReduce algorithms and further, how some of the previously discussed algorithms can *completely* remove any associated bottlenecks.

Chapter Overview We will begin by describing the MapReduce framework, which constitutes a way to transform serially executed programs to parallel ones. Further, we will discuss its extension to Coded MapReduce (CMR) [15], where the tools of Coded Caching, i.e. using side information to allow for message multicasting, have been used to provide significant communication reductions. Then, we will discuss two main bottlenecks of the CMR framework, the subpacketization bottleneck [121] and the co-existence of heterogeneous computing nodes [122]. Finally, we will describe algorithms that can *completely* resolve these two bottlenecks.

8.1.1 MapReduce

The MapReduce (MR) model [123] is a parallel computing framework that transforms a sequential problem into a parallel one. For a setting of K computing nodes connected through a bottleneck channel, the ultimate objective is the parallel computation of $Q \geq K$ functions on a dataset F comprized of f elements. To facilitate the execution of $\frac{Q}{K}$ functions at each node i.e., to allow a parallel execution of the final problem, the MR process takes place in three distinct phases. More specifically, the phases are:

1. the *mapping phase*, where each element of the dataset is assigned to one or more computing nodes and where the nodes perform an intermediate computation aiming to "prepare" for parallelization,

2. the *shuffling phase* (or communication phase), where nodes communicate between each other the preprocessed data that is needed to make the process parallel, and

3. the *reduce phase*, where nodes work in a completely parallel manner to provide the final output that each is responsible for.

Classes of tasks that can be parallelized under a MapReduce framework include Sorting [50], Data Analysis and Clustering [124,125] and Word Counting [126] among others [127].

The communication bottleneck of distributed computing While though MapReduce allows for parallelization, it also comes with different bottlenecks involving for example struggling nodes [126, 128] and non-fine-tuned algorithms [129]. The main bottleneck though that bounds the performance of MapReduce is the duration of the shuffling phase, especially as the dataset size becomes larger and larger. While having more nodes can speed up the computation time, the aforementioned information exchange often yields unchanged or even increased communication load and delays, leading to a serious bottleneck in the performance of distributed computing algorithms.

The required execution time of the MR model, assuming that the three phases happen in sequence, can be seen to be

$$T_{\text{tot}}^{\text{MR}} = T_{\text{map}}\left(\frac{f}{K}\right) + T_{\text{shuf}}\frac{K-1}{K} + \frac{Q}{K}T_{\text{red}} \tag{8.1}$$

where $T_{\text{map}}(d \cdot f)$ is the time spent, by one node, to pre-process a fraction d of the dataset, T_{shuf} is the required time to communicate the entire amount of intermediate values between any two nodes, and T_{red} denotes the time required, by a single node, to complete the reduction part of a single function.

As we can see from Eq. (8.1), the Mapping and Reduce phases are reducing as the number of nodes is increased. On the other hand, the problem lies with the communication delay T_{shuf} which, remains approximately the same thus, can provide the main bottleneck.

8.1.2 Emergence of Coded MapReduce: exploiting computing redundancy

For a general distributed computing problem fitting the aforementioned MapReduce model, a method of reducing the communication load was introduced in [15], which modified the mapping phase, in order to allow for the shuffling phase to employ coded communication. The main idea of the method — which is referred to as Coded MapReduce (CMR) — was to assign and then force each node to map fraction $\gamma > \frac{1}{K}$ of the whole dataset (such that each element of the dataset is mapped in $t = K\gamma$ computing nodes) and then — based on the fact that such a mapping would allow for common mapped

information at the different nodes — to eventually perform coded communication where during the shuffling phase, the packets were not sent one after the other, but were rather combined together into XORs and sent as one.

This allowed the total achieved execution time to take the form

$$T_{\text{tot}}^{\text{CMR}} = T_{\text{map}}(\gamma f) + T_{\text{shuf}} \frac{1-\gamma}{K\gamma} + \frac{Q}{K} T_{\text{red}} \tag{8.2}$$

where we can see that the mapping cost has increased, while the shuffling cost has decreased significantly, forming a trade-off between increasing the mapping cost and decreasing the shuffling cost.

The reason this speedup would work is because the recipients of these packets could use part of their (redundant) mapped packets as side information in order to remove the interfering packets from the received XOR, and acquire their own requested packet.

Coding for Straggler Mitigation

Another line of work (see [128,130–134]), has considered a different bottleneck of distributed computing (not falling under the MR framework) that is now caused by some nodes experiencing delays during the computation, thus causing significant delays in the overall execution time of the algorithm. The main idea behind these efforts to alleviate this so-called straggling effect, is to split the dataset into some $L < K$ parts and assign to nodes a function of one or more parts (usually in the form of coded linear combinations), thus needing only a subset of the results from the nodes (in particular the faster ones) in order to recover the final result.

While coding for straggling nodes can provide an increased performance, nevertheless it suffers from two main limitations: i) an increased load assigned to each node (fraction $\frac{1}{L}$ instead of fraction $\frac{1}{K}$ of the dataset), and ii) a restriction to linear problems (with the notable extension to polynomial problems [135]) thus excluding other tasks such as sorting.

8.1.3 Subpacketization bottleneck of distributed computing

Despite the fact that the aforementioned coded method promises, in theory, big delay reductions by a factor of $t = K\gamma$ compared to conventional uncoded schemes, these gains are heavily compromised by the fact that the method requires that the dataset be split into an unduly large number of packets[1]

$$S = \binom{K}{t} \geq \left(\frac{K}{t}\right)^t. \tag{8.3}$$

[1]The subpacketization S strictly refers to the number of chunks the dataset has to be split into in order to prepare the mapping phase. At the time when the shuffling phase takes place, CMR requires to further split each mapped chunk into $K\gamma$ equally-sized smaller parts. This extra subpacketization, which occurs at a bit level, is negligible compared to S and it is not taken into account in our analysis.

Specifically, the fact that the finite-sized dataset can only be divided into a finite number of packets, limits the values of parameter t that can be achieved, because the corresponding subpacketization S must be kept below some maximum allowable subpacketization $S_{\max}$ which, also, must be less than the number of elements f in the dataset. If this number $S = \binom{K}{t}$ exceeds the maximum allowable subpacketization $S_{\max}$, then coded communication is limited to include coding that spans only

$$\bar{K} = \arg\max_{K} \left\{ \binom{K}{t} \leq S_{\max} \right\} \tag{8.4}$$

nodes at a time, forcing us to repeat coded communication $K/\bar{K}$ times, thus resulting in a smaller, actual gain

$$\bar{t} = \bar{K}\gamma < K\gamma$$

which can be far below from the theoretical communication gain from coding. Such high subpacketization can naturally limit the coding gains t, but it can also further delay the shuffling phase because it implies more transmissions and thus higher packet overheads, as well as because smaller packets are more prone to have mapped outputs that are unevely sized, thus requiring more zero padding.

In what follows, we will solve the above problems with a novel group-based method of distributing the dataset across the computing nodes, and a novel method of cooperation/coordination between nodes in the transmission, which will jointly yield a much reduced subpacketization, allowing for a wider range of t values to be feasible, thus eventually allowing substantial reductions in the overall execution time for a large class of distributed computing algorithms.

8.1.4 Heterogeneous Nodes

In the second part, we assume that nodes have heterogeneous computing capabilities during the mapping phase. Specifically, K_1 nodes (set $\mathcal{K}_1$) can perform mapping faster than the rest $K_2 = K - K_1$ nodes (set $\mathcal{K}_2$), and as a result each node of the first type is tasked with mapping a fraction $\gamma_1 \in \left[\frac{1}{K_1}, 1\right]$ of the dataset, while each node of the second type is tasked with mapping a smaller fraction $\gamma_2 \in [0, \gamma_1)$.

Using the increased set-to-set dimensionality that can be found in wireless settings (cf. Section (8.1.5)) and the wired setting with intermediary nodes (cf. Section (8.1.6)), we exploit the redundancy at all nodes and reduce the shuffling delay by a factor of $K_1\gamma_1 + K_2\gamma_2$ that matches exactly the cumulative computed redundancy. This solution will show that even though the system is computationally heterogeneous, it actually performs like an equivalent homogeneous system where each node could map a uniform fraction $\gamma = \frac{K_1\gamma_1 + K_2\gamma_2}{K}$ of the dataset.

Data assignment uses a modification of the original CMR algorithm of [15]. The novel communication algorithm during the shuffling phase manages to consistently serve $K_1\gamma_1 + K_2\gamma_2$ nodes per transmission, a gain that appears either because a) nodes can transmit messages as if the two types were not interfering, or otherwise because b) nodes of set $\mathcal{K}_1$ can jointly transmit as if they were a single transmitter with $K_1\gamma_1$ antennas.

8.1.5 Channel model: Distributed computing in a D2D setting

We assume that the K computing nodes are all fully connected via a wireless shared channel as in the classical fully-connected D2D wireless network. At each point there will be a set of active receivers, and active transmitters. Assuming a set of L active transmitters jointly transmitting vector $\mathbf{x} \in \mathbb{C}^{L\times 1}$, then the received signal at a receiving node k takes the form

$$y_k = \mathbf{h}_k^T \mathbf{x} + w_k, \tag{8.5}$$

where as always $\mathbf{x}$ satisfies a power constraint $\mathbb{E}(||\mathbf{x}||^2) < P$, where $\mathbf{h}_k \in \mathbb{C}^{L\times 1}$ is the (potentially random) fading channel between the transmitting set of nodes and the receiving node k, and where w_k denotes the unit-power AWGN noise at receiver k. We assume the system to operate in the high SNR regime (high P), and we assume perfect channel state information (CSI) (and for the wired case, perfect network coding coefficients) at the active receivers and transmitters.

8.1.6 High-Dimensional CRM in the wired medium

We note that the same procedure that takes place in the wireless channel, can be directly applied in the wired setting where the intermediate nodes (routers, switches, etc.) in the links, can perform pseudo-random network coding operations on the received data (cf. [10]). This would then automatically yield a linear invertible relationship between the input vectors and the received signals, thus allowing for the design of the precoders that cancel intra-group interference. This is also depicted in Figure 8.1.

8.1.7 Related work

The CMR framework has sparked significant interest due to its ability to allow the shuffling phase of MapReduce to scale with the number of nodes. The work in [136] considered the wireless channel with full-duplex links and proposed new assignment and shuffling phases that could allow the doubling of the rate. Further, the work in [137] considered using an MDS outer code in order to address straggling nodes during the mapping phase. Further, the work in [138] managed to reduce even further the execution time of the mapping phase without effecting the associated shuffling gains.

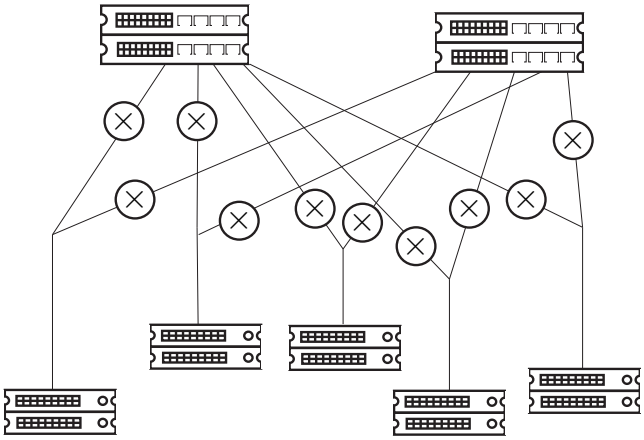

Figure 8.1: Illustration of the wired setting. $\times$ denotes a network coding operation.

8.1.8 Schemes Preliminaries

In the section we will introduce some preliminaries that cover both presented schemes. Specifically, we will introduce the mapping phase and reduce phases, while the assignment of segments to the nodes, as well as the respective shuffling phases are treated individually in the respective sections (see Section 8.2 and Section 8.3).

The process commences with the *mapping phase*, where the dataset, F, is divided into S chunks, $F_1, ..., F_S$, which are distributed to the nodes, such that node k is assigned chunks

$$\mathcal{M}_k = \left\{F_{s_k},\ s_k \in [S]\right\} \tag{8.6}$$

and where

$$\frac{|\mathcal{M}_k|}{S} = \gamma, \quad \forall k \in [K]. \tag{8.7}$$

Each node is responsible for executing a set of mapping functions $\{m_q,\ \forall q \in [Q]\}$ on each of its assigned dataset chunks $s_k \in \mathcal{M}_k$. The result of the mapping phase is the set of intermediate values $\{m_q(F_s), q \in [Q],\ s \in [S]\}$.

In the reduce phase, each node k will be assigned the set of functions $Q_k \subset [Q]$ in order to perform the reduction. Thus, at the end of the mapping phase, all nodes need to communicate intermediate values guaranteeing that node $k \in [K]$ will gather

$$\mathcal{I}(q_k) = \{m_q(F_s) \triangleq F_s^{(q)},\ \forall s \in [S]\},\ \forall q \in Q_k. \tag{8.8}$$

In what follows, we will assume, for simplicity, that $Q = K$ and that node k will be assigned function $q = k$.

We define as Category $q \in [Q]$ the collection of all intermediate values that, after reduction, can produce result $q \in [Q]$.

During the reduce phase the objective is for each node to perform functions $r_q,\ \forall q \in Q_k$ on the set $\mathcal{I}_q$ of intermediate values that it has gathered.

8.2 Node Cooperation to reduce Subpacketization in Coded MapReduce

Main result

We proceed to describe the performance of the new proposed algorithm. Key to this algorithm — which we will refer to as the Group-based Coded MapReduce (GCMR) algorithm — is the concept of user grouping, similar to work [56]. We will group the K nodes into $\frac{K}{L}$ groups of L nodes each, and then every node in a group will be assigned the same subset of the dataset and will produce the same mapped output. By properly doing so, this will allow us to use in the shuffling phase a new D2D coded caching communication algorithm which assigns the D2D nodes an adaptive amount of content. This will in turn substantially reduce the required subpacketization, thus substantially boosting the speedup in communication and overall execution time.

For the sake of comparison, let us first recall that under the subpacketization constraint $S_{\max}$, the original Coded MapReduce approach achieves communication delay

$$T_{\text{shuf}}^{\text{CMR}} = \frac{1-\gamma}{\bar{t}} T_c \tag{8.9}$$

where

$$\bar{t} = \gamma \cdot \arg\max_{\bar{K} \leq K} \left\{ \binom{\bar{K}}{\bar{K}\gamma} \leq S_{\max} \right\} \tag{8.10}$$

is the maximum achievable effective speedup (due to coding) in the shuffling phase. In the above and in what follows, we assume for simplicity that $Q = K$ such that each node has one final task.

We proceed with the main result.

Theorem 8.1. *In the K-node distributed computing setting where the dataset can only be split into at most $S_{\max}$ identically sized packets, the proposed Group-based Coded MapReduce algorithm with groups of L users, can achieve communication delay*

$$T_{shuf}^{GCMR} = \frac{1-\gamma}{\bar{t}_L} T_c \tag{8.11}$$

for

$$\bar{t}_L = \gamma \cdot \arg\max_{\bar{K} \leq K} \left\{ \binom{\bar{K/L}}{\bar{K/L}\gamma} \leq S_{\max} \right\} \tag{8.12}$$

Proof. The proof follows directly from the scheme description. □

The above implies the following corollary, which reveals that in the presence of subpacketization constraints, simple node grouping can further speedup the shuffling phase by a factor of up to L.

Corollary 8.1. *In the subpacketization-constrained regime where* $S_{\max} \leq \binom{K/L}{K\gamma/L}$, *the new algorithm here allows for shuffling delay*

$$T_{shuf}^{GCMR} = \frac{1-\gamma}{\bar{t}_L} T_c = \frac{T_{shuf}^{CMR}}{L} \tag{8.13}$$

which is L *times smaller than the delay without grouping for the same choice of the parameter* γ.

Proof. The proof is direct from the theorem. □

Finally the following also holds.

Corollary 8.2. *When* $S_{\max} \geq \binom{K/L}{K\gamma/L}$, *the new algorithm allows for the unconstrained theoretical execution time*

$$T_{tot}^{GCMR} = T_{map}(\gamma F) + \frac{(1-\gamma)}{K\gamma} T_c + T_{red}\left(\frac{F}{K}\right). \tag{8.14}$$

Proof. The proof is direct from Theorem 8.1. □

Dataset assignment phase

We split the K nodes into $K' \triangleq \frac{K}{L}$ groups

$$\mathcal{G}_i = \{i, i+K', ..., i+(L-1)K'\}, \quad i \in [K'] \tag{8.15}$$

of L nodes per group, and we split the dataset into

$$S_{\text{GCMR}} = \binom{K'}{K'\gamma} \tag{8.16}$$

segments, where $\gamma \in \{\frac{1}{K'}, \frac{2}{K'}, \cdots, 1\}$ is a parameter of choice defining the redundancy factor of the mapping phase later on.

Thus, the dataset is divided into

$$F \to \left\{F_\tau, \ \tau \subset \left[\frac{K}{L}\right], \ |\tau| = \frac{K\gamma}{L}\right\}. \tag{8.17}$$

Each node $g_k \in \mathcal{G}_k$ is assigned the set of segments

$$\mathcal{M}_{\mathcal{G}_k} = \{F_\tau : \ k \in \tau\}. \tag{8.18}$$

Shuffle Phase

Each node g_k of group $\mathcal{G}_k$, must retrieve from the nodes of the other groups[2] the set of intermediate values

$$\{F_\tau^{(q)} : \ k \notin \tau, \ \forall q \in Q_{g_k}\} \tag{8.19}$$

[2]Since the nodes of the same group have computed exactly the same results.

i.e., all the intermediate values that correspond to function set Q_{g_k} that have not been computed locally ($k \notin \tau$).

The shuffling process is described in Algorithm 8.1, which we proceed to analyze.

Initially, the set of active groups are selected (Step 1), such that a total of $\frac{K\gamma}{L}+1$ groups are selected. From these groups, one-by-one a group is selected to be the transmitting group (Step 2). The nodes of the transmitting group cooperate and form a distributed MIMO system that transmits the $L\times 1$ vector

$$\mathbf{x}_{s,\sigma} = \sum_{k\in\sigma\setminus\{s\}} \mathcal{H}^{-1}_{\mathcal{G}_s,\mathcal{G}_k} \begin{bmatrix} F^{(g_k(1))}_{\sigma\setminus\{k\}} \\ \vdots \\ F^{(g_k(L))}_{\sigma\setminus\{k\}} \end{bmatrix}. \tag{8.20}$$

Examining Eq. (8.20) we can see that the transmitted vector is a linear combination of $\frac{K\gamma}{L}$ vectors, where each vector is conveying information to the nodes of a specific group.

1 **for** *all* $\sigma \subseteq \left[\frac{K}{L}\right]$, $|\sigma| = \frac{K\gamma}{L}$ *(pick active groups)* **do**
2 **for** *all* $s \in \sigma$ *(pick transmitting group)* **do**
3 Group $\mathcal{G}_s$ transmits:

$$\mathbf{x}_{s,\sigma} = \sum_{k\in\sigma\setminus\{s\}} \mathcal{H}^{-1}_{\mathcal{G}_s,\mathcal{G}_k} \begin{bmatrix} F^{(g_k(1))}_{\sigma\setminus\{k\}} \\ \vdots \\ F^{(g_k(L))}_{\sigma\setminus\{k\}} \end{bmatrix}. \tag{8.21}$$

4 **end**
5 **end**

Algorithm 8.1: Shuffling Phase with Node Cooperation

Decoding Let us assume that the selected set of groups is σ and the transmitting group is $s \in \sigma$. Then, for some node $p \in \mathcal{G}_p$, $p \in \sigma \setminus \{s\}$, the received message takes the form (Noise is suppressed for simplicity)

$$y_p = \mathbf{h}^T_{\mathcal{G}_s,p} \cdot \sum_{k\in\sigma\setminus\{s\}} \mathcal{H}^{-1}_{\mathcal{G}_s,\mathcal{G}_k} \begin{bmatrix} F^{(g_k(1))}_{\sigma\setminus\{k\}} \\ \vdots \\ F^{(g_k(L))}_{\sigma\setminus\{k\}} \end{bmatrix}. \tag{8.22}$$

First, we notice that from all the $\frac{K\gamma}{L}$ vectors that are summed, the $\frac{K\gamma}{L}-1$ are intermediary values that have been previously been calculated by node p can be removed, leaving the resulting message to be

$$y^{rem}_p = \mathbf{h}^T_{\mathcal{G}_s,p} \cdot \mathcal{H}^{-1}_{\mathcal{G}_s,\mathcal{G}_p} \begin{bmatrix} F^{(g_p(1))}_{\sigma\setminus\{p\}} \\ \vdots \\ F^{(g_p(L))}_{\sigma\setminus\{p\}} \end{bmatrix}. \tag{8.23}$$

Due to the design of the precoder matrix $\mathcal{H}^{-1}_{\mathcal{G}_s,\mathcal{G}_p}$ all nodes belonging in group $\mathcal{G}_p$ are spatially separated thus will receive their messages interference-free.

8.2.1 Node cooperation example

Let us consider a setting with $K = 32$ computing nodes, a chosen redundancy of $K\gamma = 16$, and a cooperation parameter $L = 8$. The nodes are split into $\frac{K}{L} = 4$ groups

$$\begin{aligned}\mathcal{G}_1 =&\{1, 5, 9, 13, 17, 21, 25, 29\},\\ \mathcal{G}_2 =&\{2, 6, 10, 14, 18, 22, 26, 30\},\\ \mathcal{G}_3 =&\{3, 7, 11, 15, 19, 23, 27, 31\},\\ \mathcal{G}_4 =&\{4, 8, 12, 16, 20, 24, 28, 32\}\end{aligned}$$

and the dataset is split into $\binom{K/L}{K\gamma/L} = 6$ segments as

$$F \to \{F_{12}, F_{13}, F_{14}, F_{23}, F_{24}, F_{34}\} \tag{8.24}$$

Segments are distributed to the nodes as follows

$$\begin{aligned}\mathcal{M}_{\mathcal{G}_1} =&\{F_{12}, F_{13}, F_{14}\},\\ \mathcal{M}_{\mathcal{G}_2} =&\{F_{12}, F_{23}, F_{24}\},\\ \mathcal{M}_{\mathcal{G}_3} =&\{F_{13}, F_{23}, F_{34}\},\\ \mathcal{M}_{\mathcal{G}_4} =&\{F_{14}, F_{24}, F_{34}\}.\end{aligned}$$

In the mapping phase, each segment F_τ is mapped into the set of intermediate values $\{F^q_\tau\}^K_{q=1}$. The transmissions that will deliver all needed intermediate values are (for simplicity we use $\mathcal{H}^{-1}_{i,j} \triangleq \mathcal{H}^{-1}_{\mathcal{G}_i,\mathcal{G}_j}$).

$$\begin{aligned}\mathbf{x}_{1,123} =&\mathcal{H}^{-1}_{1,2}\mathbf{F}^{\mathcal{G}_2}_{13,1} + \mathcal{H}^{-1}_{1,3}\mathbf{F}^{\mathcal{G}_3}_{12,1}\\ \mathbf{x}_{1,124} =&\mathcal{H}^{-1}_{1,2}\mathbf{F}^{\mathcal{G}_2}_{14,1} + \mathcal{H}^{-1}_{1,4}\mathbf{F}^{\mathcal{G}_4}_{12,1}\\ \mathbf{x}_{1,134} =&\mathcal{H}^{-1}_{1,3}\mathbf{F}^{\mathcal{G}_3}_{14,1} + \mathcal{H}^{-1}_{1,4}\mathbf{F}^{\mathcal{G}_4}_{13,1}\\ \mathbf{x}_{2,123} =&\mathcal{H}^{-1}_{2,1}\mathbf{F}^{\mathcal{G}_1}_{23,2} + \mathcal{H}^{-1}_{2,3}\mathbf{F}^{\mathcal{G}_3}_{12,2}\\ \\ \mathbf{x}_{2,124} =&\mathcal{H}^{-1}_{2,1}\mathbf{F}^{\mathcal{G}_1}_{24,2} + \mathcal{H}^{-1}_{2,4}\mathbf{F}^{\mathcal{G}_4}_{12,2}\\ \mathbf{x}_{2,234} =&\mathcal{H}^{-1}_{2,3}\mathbf{F}^{\mathcal{G}_3}_{24,2} + \mathcal{H}^{-1}_{2,4}\mathbf{F}^{\mathcal{G}_4}_{23,2}\\ \mathbf{x}_{3,123} =&\mathcal{H}^{-1}_{3,1}\mathbf{F}^{\mathcal{G}_1}_{13,3} + \mathcal{H}^{-1}_{3,2}\mathbf{F}^{\mathcal{G}_2}_{13,3}\\ \mathbf{x}_{3,134} =&\mathcal{H}^{-1}_{3,1}\mathbf{F}^{\mathcal{G}_1}_{34,3} + \mathcal{H}^{-1}_{3,4}\mathbf{F}^{\mathcal{G}_4}_{13,3}\\ \\ \mathbf{x}_{3,234} =&\mathcal{H}^{-1}_{3,2}\mathbf{F}^{\mathcal{G}_2}_{34,3} + \mathcal{H}^{-1}_{3,4}\mathbf{F}^{\mathcal{G}_4}_{23,3}\\ \mathbf{x}_{4,124} =&\mathcal{H}^{-1}_{4,1}\mathbf{F}^{\mathcal{G}_1}_{24,4} + \mathcal{H}^{-1}_{4,2}\mathbf{F}^{\mathcal{G}_2}_{14,4}\\ \mathbf{x}_{4,134} =&\mathcal{H}^{-1}_{4,1}\mathbf{F}^{\mathcal{G}_1}_{34,4} + \mathcal{H}^{-1}_{4,3}\mathbf{F}^{\mathcal{G}_3}_{14,4}\\ \mathbf{x}_{4,234} =&\mathcal{H}^{-1}_{4,2}\mathbf{F}^{\mathcal{G}_2}_{34,4} + \mathcal{H}^{-1}_{4,3}\mathbf{F}^{\mathcal{G}_3}_{24,4},\end{aligned}$$

where we have used

$$\mathbf{F}_{\tau}^{\mathcal{G}_g} \triangleq \begin{bmatrix} F_{\tau}^{g(1)} \\ \vdots \\ F_{\tau}^{g(L)} \end{bmatrix} \tag{8.25}$$

to denote the vector of $L = 8$ elements consisting of the intermediate results intended for nodes in group $\mathcal{G}_g$.

Observing for example the first transmission, we see that the nodes in group $\mathcal{G}_2$ can remove any interference caused by the intermediate results intended for group $\mathcal{G}_3$ since these intermediate values have been calculated by each node in $\mathcal{G}_2$ during the mapping phase. After noting that the precoding matrix $\mathcal{H}_{1,2}^{-1}$ removes intra-group interference, we can conclude that each transmission serves each of the 16 users with one of their desired intermediate values, which in turn implies a 16-fold speedup over the uncoded case.

8.3 CMR with Heterogeneous nodes

To reflect the heterogeneous capabilities of the computing nodes during the mapping phase – as stated before – we consider a set of K_1 computing nodes, $\{1, 2, ..., K_1\} \triangleq \mathcal{K}_1$, being able of mapping on time a fraction $\gamma_1 \in \left[\frac{1}{K}, 1\right]$ of the dataset, while the rest $K_2 = K - K_1$ nodes, $\{K_1+1, ..., K\} \triangleq \mathcal{K}_2$, are able to map a smaller fraction $\gamma_2 \in [0, \gamma_1)$ of the dataset.

Main Results

The main contribution of this work is the design of a new algorithm for dataset assignment, and a new transmission policy during the shuffling phase. The achievable performance for any values of γ_1, γ_2 yields

$$\begin{aligned} T_{\text{tot}}^{\text{hCMR}}(\gamma_1,\gamma_2) = &\max\left\{T_{\text{map}}^{(1)}(\gamma_1 f), T_{\text{map}}^{(2)}(\gamma_2 f)\right\} + \frac{T_{\text{shuf}}}{K}\frac{1-\gamma_1}{\gamma_1} \\ &+ \frac{T_{\text{shuf}}}{K}\frac{K_2\left(1-\frac{\gamma_2}{\gamma_1}\right)}{\min\{K_2, K_1\gamma_1 + K_2\gamma_2\}} + \frac{Q}{K}T_{\text{red}} \end{aligned} \tag{8.26}$$

where $T_{\text{map}}^{(j)}(\gamma_j F),\ j = 1, 2$ corresponds to the time required for each node in group j to map fraction γ_j of the dataset, and where the delay reflects the fact that the shuffling phase will conclude when the mapping at any node has finished. In order to minimize Eq. (8.26), we choose values γ_1, γ_2 in such a way to reflect the relative computing capabilities[3] i.e., $T_{\text{map}}^{(1)}(\gamma_1 F) = T_{\text{map}}^{(2)}(\gamma_2 F)$. This is summarized in the following theorem.

[3]Thus, if nodes in $\mathcal{K}_1$ are twice as fast as nodes in $\mathcal{K}_2$, then $\gamma_1 = 2\gamma_2$.

Theorem 8.2. *The achievable time of a MapReduce algorithm in a heterogeneous distributed computing system with K_1 nodes each able to map fraction γ_1 of the total dataset, while K_2 nodes can map fraction γ_2 of the dataset each, takes the form*

$$T_{tot}^{hCMR}(\gamma_1,\gamma_2)=T_{map}^{(1)}(\gamma_1 f)+\frac{T_{shuf}}{K}\frac{1-\gamma_1}{\gamma_1}+ \\ +\frac{T_{shuf}}{K}\frac{K_2\left(1-\frac{\gamma_2}{\gamma_1}\right)}{\min\{K_2,K_1\gamma_1+K_2\gamma_2\}}+\frac{Q}{K}T_{red} \tag{8.27}$$

where the communication (shuffling) cost can be simplified, in the case of $K_2 \geq K_1\gamma_1+K_2\gamma_2$, to

$$T_{comm}^{hCMR}(\gamma_1,\gamma_2)=\frac{T_{shuf}}{K}\frac{K_1(1-\gamma_1)+K_2(1-\gamma_2)}{K_1\gamma_1+K_2\gamma_2} \tag{8.28}$$

Proof. The proof is constructive and presented in Sec. 8.3.1. □

Remark 8.1. *The effect of heterogeneity can be completely removed.*

The above holds directly from Eq. (8.28) where we see that the delay matches that of a homogeneous system with K nodes and a uniform $\gamma = \frac{K_1\gamma_1+K_2\gamma_2}{K}$.

Remark 8.2. *Heterogeneous systems with interference are faster than multiple homogeneous systems even when the latter multiple systems experience no inter-system interference.*

This is because, by comparing the proposed solution with a second system where now the two sets $\mathcal{K}_1$ and $\mathcal{K}_2$ are 'magically' non-interfering (isolated), we conclude that while the mapping delay of the two systems would be the same, the shuffling phase delay $T_{\text{com}} = \frac{T_{\text{shuf}}}{K}\max\left\{\frac{1-\gamma_1}{\gamma_1},\frac{1-\gamma_2}{\gamma_2}\right\} = \frac{T_{\text{shuf}}}{K}\frac{1-\gamma_2}{\gamma_2}$ of the second system would exceed that of our system. This reveals an important advantage of larger heterogeneous systems over multiple non-interfering homogeneous ones, and it suggests that the collaborative effects from mapping data to more nodes, despite heterogeneity, exceeds any effect of having parallel (smaller, homogeneous) systems. This surprising conclusion is mainly due to the fact that "strong" nodes have the potential to assist the transmission of intermediate values to the "weak" nodes, thus boosting the overall performance.

8.3.1 Scheme Description

Data Assignment

We begin by dividing the dataset into

$$S^{\text{het}} = \binom{K_1}{K_1\gamma_1}\cdot\binom{K_2}{K_2\gamma_2} \tag{8.29}$$

segments i.e., F_{τ_1,τ_2}, $\tau_1 \subset \mathcal{K}_1$, $|\tau_1| = K_1\gamma_1$, $\tau_2 \subset \mathcal{K}_2$, $|\tau_2| = K_2\gamma_2$.

8.3.2 Mapping Phase

Nodes $k_1 \in \mathcal{K}_1$ and $k_2 \in \mathcal{K}_2$ are assigned to respectively map[4]

$$\mathcal{M}_{k_1} = \{F_{\tau_1,\tau_2} : k_1 \in \tau_1,\ \forall \tau_2 \subset \mathcal{K}_2, |\tau_2| = K_2\gamma_2\}$$
$$\mathcal{M}_{k_2} = \{F_{\tau_1,\tau_2} : k_2 \in \tau_2,\ \forall \tau_1 \subset \mathcal{K}_1, |\tau_1| = K_1\gamma_1\}.$$

8.3.3 Shuffling Phase

After the mapping phase, the nodes exchange information so that each node $k \in [K]$ can collect the set

$$\mathcal{I}_k = \left\{F^{(q)}_{\tau_1,\tau_2},\ \tau_j \subseteq \mathcal{K}_j, |\tau_j| = K_j\gamma_j,\ j = \{1,2\},\ q \in Q_k\right\}$$

where we remind that $F^{(q)}_{\tau_1,\tau_2} \triangleq m_q(F_{\tau_1,\tau_2})$.

We will employ a novel way of transmission, comprized of two sub-phases. The first sub-phase is reminiscent of the original CMR approach [15], where though now we will allow two nodes to transmit simultaneously, one node from each group; this can happen because we choose the transmitted messages to be completely known to the receiving nodes of the other group. During the second shuffling sub-phase, nodes from $\mathcal{K}_1$ will coordinate to act as a distributed multi-antenna system and transmit the remaining data to nodes of set $\mathcal{K}_2$.

for *all* $\tau_1 \subset \mathcal{K}_1$, $|\tau_1| = K_1\gamma_1$ *(pick Rx nodes from* $\mathcal{K}_1$*)* **do**
  for *all* $\tau_2 \subset \mathcal{K}_2$, $|\tau_2| = K_2\gamma_2$ *(pick Rx nodes from* $\mathcal{K}_2$*)* **do**
    for *all* $k_1 \in \mathcal{K}_1 \setminus \tau_1$ *(pick transmitter 1)* **do**
      for *all* $k_2 \in \mathcal{K}_2 \setminus \tau_2$ *(pick transmitter 2)* **do**
        Transmit concurently:

$$\text{Tx } k_1\text{: } x^{k_2,\tau_2}_{k_1,\tau_1} = \bigoplus_{m\in\tau_1} F^{(q_m),k_1,k_2}_{\{k_1\}\cup\tau_1\setminus\{m\},\tau_2}$$
$$\text{Tx } k_2\text{: } x^{k_1,\tau_1}_{k_2,\tau_2} = \bigoplus_{n\in\tau_2} F^{(q_n),k_1,k_2}_{\tau_1,\{k_2\}\cup\tau_2\setminus\{n\}}$$

      end
    end
  end
end

Algorithm 8.2: Shuffling Sub-Phase 1 of the Heterogeneous Nodes Scheme

[4]The dataset fraction assigned to a node in each group yields

$$\frac{|\mathcal{M}_{k_i}|}{F} = \frac{\binom{K_i-1}{K_i\gamma_i-1}\cdot\binom{K_j}{K_j\gamma_j}}{\binom{K_i}{K_i\gamma_i}\cdot\binom{K_j}{K_j\gamma_j}} = \gamma_j,\ i,j \in \{1,2\}, i \neq j.$$

First Shuffling Sub-Phase (Transmission)

At the beginning of the first sub-phase, an intermediate value $F_{\tau_1,\tau_2}^{(q_k)},\ q_k \in Q_k$ required by node $k \in [K]$, is further subpacketized as

$$F_{\tau_1,\tau_2}^{(q_k)} \to \begin{cases} \left\{F_{\tau_1,\tau_2}^{(q_k),k_1,k_2}, k_1 \in \tau_1,\ \ k_2 \in \mathcal{K}_2 \setminus \tau_2\right\} & k \in \mathcal{K}_1 \\ \left\{F_{\tau_1,\tau_2}^{(q_k),k_1,k_2}, k_1 \in \mathcal{K}_1 \setminus \tau_1,\ \ k_2 \in \tau_2\right\} & k \in \mathcal{K}_2 \end{cases} \tag{8.30}$$

yielding the relative sub-packet sizes

$$\frac{\left|F_{\tau_1,\tau_2}^{(q_k),k_1,k_2}\right|}{\left|F_{\tau_1,\tau_2}^{(q_k)}\right|} = \begin{cases} \dfrac{1}{K_1\gamma_1 K_2(1-\gamma_2)}, & \text{if } k \in \mathcal{K}_1 \\ \dfrac{1}{K_1(1-\gamma_1)K_2\gamma_2}, & \text{if } k \in \mathcal{K}_2 \end{cases} \tag{8.31}$$

meaning that sub-packets for nodes in set $\mathcal{K}_2$ are larger than sub-packets for nodes in set $\mathcal{K}_1$. To rectify this uneveness and allow for the simultaneous transmissions required by our algorithm, we will equalize the packet sizes transmitted from both $\mathcal{K}_1$ and $\mathcal{K}_2$ by further splitting the sub-packets intended for node $k \in \mathcal{K}_2$ into two parts, with respective sizes

$$\left|F_{\tau_1,\tau_2}^{\prime(q_k),k_1,k_2}\right| = \frac{\left|F_{\tau_1,\tau_2}^{(q_k)}\right|}{K_1\gamma_1 K_2(1-\gamma_2)} \tag{8.32}$$

$$\left|F_{\tau_1,\tau_2}^{\prime\prime(q_k),k_1,k_2}\right| = \left|F_{\tau_1,\tau_2}^{(q_k)}\right| \frac{1-\gamma_2/\gamma_1}{K_1(1-\gamma_1)K_2(1-\gamma_2)\gamma_2}. \tag{8.33}$$

The process of transmitting packets in the first shuffling sub-phase is described in Algorithm 8.2, which successfully communicates a single category to each of the nodes.

In Algorithm 8.2, Steps 1 and 2 select, respectively, a subset of receiving nodes from set $\mathcal{K}_1$ and a subset of receiving nodes from set $\mathcal{K}_2$. Then, Steps 3 and 4 pick transmitting nodes from the sets $\mathcal{K}_1$ and $\mathcal{K}_2$, respectively such that these nodes are not in the already selected receiver sets from Steps 1 and 2. Then, in Step 5, the two selected transmitting nodes create and simultaneously transmit their packets.

Decoding in First Shuffling Sub-phase In Algorithm 8.2, transmitted sub-packets for nodes of set $\mathcal{K}_1$ are picked to be known at all nodes of set $\mathcal{K}_2$, and vice versa. This allows nodes of a set to communicate as if not being interfered by nodes of the other set.

First Shuffling Sub-phase Performance The above-described steps are arranged in nested loops that will, eventually, pick all possible combinations of $(k_1, k_2, \tau_1, \tau_2)$, thus delivering all needed sub-packets to nodes in $\mathcal{K}_1$. The normalized completion time of this first sub-phase is

$$T_{\text{comm},1} = \frac{Q}{K}\frac{K_1(1-\gamma_1)\binom{K_1}{K_1\gamma_1}K_2(1-\gamma_2)\binom{K_2}{K_2\gamma_2}}{K_1\gamma_1 K_2(1-\gamma_2)\binom{K_1}{K_1\gamma_1}\binom{K_2}{K_2\gamma_2}} = \frac{Q}{K}\frac{1-\gamma_1}{\gamma_1}$$

where factor $\frac{Q}{K}$ reflects a repetition of Algorithm 8.2 needed for delivering all categories, where the numerator reflects the four loops of the algorithm, and where the denominator uses the subpacketization level S and Eq. (8.31) to reflect the size of each transmitted packet.

Second Shuffling Sub-Phase

The second part of the shuffling phase initiates when the data required by the nodes in $\mathcal{K}_1$ have all been delivered. The transmission during this second sub-phase follows the ideas of the distributed antenna cache-aided literature (cf. [2,10,11,47]). Specifically, nodes of set $\mathcal{K}_1$ act as a distributed $K_1\gamma_1$-multi-antenna[5] system, while nodes of set $\mathcal{K}_2$ act as cache-aided receivers, cumulatively "storing" $K_2\gamma_2$ of the total "library". Directly from [2,10,11,47], we can conclude that in our setting, this setup allows a total of $\min\{K_2,\ K_1\gamma_1+K_2\gamma_2\}$ nodes to decode successfully a desired intermediate value per transmission. The remaining messages to be sent during this sub-phase are

$$\left\{F^{\prime\prime(q_k),k_1,k_2}_{\tau_1,\tau_2},\ k_1\in\mathcal{K}_1\setminus\tau_1,\ k_2\in\tau_2,\forall\tau_1,\forall\tau_2\right\}.$$

By combining the remaining messages of category q_k i.e.,

$$\{F^{\prime\prime(q_k),k_1,k_2}_{\tau_1,\tau_2}\ \forall k_1\in\mathcal{K}_1\setminus\tau_1,\ \forall k_2\in\mathcal{K}_2\}\to F^{\prime\prime(q_k)}_{\tau_1,\tau_2}, \tag{8.34}$$

we have

$$\left|F^{\prime\prime(q_k)}_{\tau_1,\tau_2}\right|=\frac{1-\gamma_2/\gamma_1}{1-\gamma_2}\left|F^{\prime\prime(q_k),k_1,k_2}_{\tau_1,\tau_2}\right|,\quad\forall\tau_1,\tau_2. \tag{8.35}$$

With this in place, the delay of the second subphase is

$$\begin{aligned}T_{\text{comm},2}&=\overbrace{\frac{Q}{K}}^{t_2}\overbrace{\frac{1-\gamma_2/\gamma_1}{(1-\gamma_2)}}^{t_3}\underbrace{\frac{\overbrace{K_2}^{t_4}\overbrace{(1-\gamma_2)}^{t_5}}{\min\{K_2,\ K_2\gamma_2+K_1\gamma_1\}}}_{t_1}\\&=\frac{Q}{K}\frac{K_2(1-\gamma_2/\gamma_1)}{\min\{K_2,\ K_2\gamma_2+K_1\gamma_1\}}\end{aligned} \tag{8.36}$$

where term t_1 corresponds to the aforementioned performance (in number of nodes served per transmission) that is achieved by multi-antenna Coded Caching algorithms [2,10,11,47]. In the above, the term t_2 corresponds to a repetition of the scheme to deliver multiple categories to each node. Finally terms t_3,t_4,t_5 represent the total amount of information that needs to be transmitted, where t_3 represents the size of each category, t_4 the number of nodes and t_5 corresponding to the (remaining) part that each node had not computed and needed to receive.

[5] We note to the reader that at this point is where the notion of the high-rank matrix is required in the equivalent wired system.

Example

Let us consider a setting where the two sets are comprised of $K_1 = 3$ and $K_2 = 4$ nodes, and where any node in each group respectively maps fractions $\gamma_1 = \frac{2}{3}$ and $\gamma_2 = \frac{1}{2}$ of the dataset. The goal is the calculation of a total of $Q = 7$ functions. Initially, dataset F is divided into $S = \binom{3}{2}\binom{4}{2} = 18$ chunks, each defined by two indices[6], $\tau_1 \in \{12, 13, 23\}$ and $\tau_2 \in \{45, 46, 47, 56, 57, 67\}$. Node $k \in [7]$ is tasked with mapping F_{τ_1,τ_2} iff $k \in \tau_1 \cup \tau_2$.

Following the mapping phase, each chunk, F_{τ_1,τ_2}, has $Q = 7$ associated intermediate values $F^{(1)}_{\tau_1,\tau_2}, ..., F^{(7)}_{\tau_1,\tau_2}$, where each intermediate value $F^{(q_k)}_{\tau_1,\tau_2}$, is further divided into sub-packets according to Eq. (8.30), while those sub-packets meant for $\mathcal{K}_2$, are further divided according to Eq. (8.31)-(8.33). For example chunks $C_{12,45}$, $D_{12,56}$ are split as

$$C_{12,45} \rightarrow \{C^{1,6}_{12,45}, C^{1,7}_{12,45}, C^{2,6}_{12,45}, C^{2,7}_{12,45}\}$$
$$D_{12,56} \rightarrow \{D^{3,5}_{12,56}, D^{3,6}_{12,56}\} \rightarrow \{D'^{3,5}_{12,56}, D''^{3,5}_{12,56}, D'^{3,6}_{12,56}, D''^{3,6}_{12,56}\}.$$

Directly from Algorithm 8.2, the sets of transmissions that correspond to shuffling sub-phase 1 are

$$\begin{aligned}
x^{4,56}_{1,23} &= B^{1,4}_{13,56} \oplus C^{1,4}_{12,56}, & x^{1,23}_{4,56} &= E^{1,4}_{23,46} \oplus F^{1,4}_{23,45} \\
x^{4,57}_{1,23} &= B^{1,4}_{13,57} \oplus C^{1,4}_{12,57}, & x^{1,23}_{4,57} &= E^{1,4}_{23,47} \oplus G^{1,4}_{23,45} \\
x^{4,67}_{1,23} &= B^{1,4}_{13,67} \oplus C^{1,4}_{12,67}, & x^{1,23}_{4,67} &= F^{1,4}_{23,47} \oplus G^{1,4}_{23,46} \\
x^{5,46}_{1,23} &= B^{1,5}_{13,46} \oplus C^{1,5}_{12,46}, & x^{1,23}_{5,46} &= D^{1,5}_{23,56} \oplus F^{1,5}_{23,45} \\
x^{5,47}_{1,23} &= B^{1,5}_{13,47} \oplus C^{1,5}_{12,47}, & x^{1,23}_{5,47} &= D^{1,5}_{23,57} \oplus G^{1,5}_{23,45}. \\
x^{5,67}_{1,23} &= B^{1,5}_{13,67} \oplus C^{1,5}_{12,67}, & x^{1,23}_{5,67} &= F^{1,5}_{23,57} \oplus G^{1,5}_{23,56}.
\end{aligned}$$

$$\begin{aligned}
x^{4,56}_{2,13} &= A^{2,4}_{23,56} \oplus C^{2,4}_{12,56}, & x^{2,13}_{4,56} &= E^{2,4}_{13,46} \oplus F^{2,4}_{13,45} \\
x^{4,57}_{2,13} &= A^{2,4}_{23,57} \oplus C^{2,4}_{12,57}, & x^{2,13}_{4,57} &= E^{2,4}_{13,47} \oplus G^{2,4}_{13,45} \\
x^{4,67}_{2,13} &= A^{2,4}_{23,67} \oplus C^{2,4}_{12,67}, & x^{2,13}_{4,67} &= F^{2,4}_{13,47} \oplus G^{2,4}_{13,46} \\
x^{5,46}_{2,13} &= A^{2,5}_{23,46} \oplus C^{2,5}_{12,46}, & x^{2,13}_{5,46} &= D^{2,5}_{13,56} \oplus F^{2,5}_{13,45} \\
x^{5,47}_{2,13} &= A^{2,5}_{13,47} \oplus C^{2,5}_{12,47}, & x^{2,13}_{5,47} &= D^{2,5}_{13,57} \oplus G^{2,5}_{13,45} \\
x^{5,67}_{2,13} &= A^{2,5}_{23,67} \oplus C^{2,5}_{12,67}, & x^{2,13}_{5,67} &= F^{2,5}_{13,57} \oplus G^{2,5}_{13,56}
\end{aligned}$$

[6]For notational simplicity, for sets we use, for example $\{12\}$ instead of $\{1, 2\}$. We also rename as $A_{\tau_1,\tau_2} \triangleq F^{(1)}_{\tau_1,\tau_2}$, $B_{\tau_1,\tau_2} \triangleq F^{(2)}_{\tau_1,\tau_2}, ..., G_{\tau_1,\tau_2} \triangleq F^{(7)}_{\tau_1,\tau_2}$ and also $F'^{(q_k)}_{\tau_1,\tau_2}$ is denoted with $F^{(q_k)}_{\tau_1,\tau_2}$, since the corresponding transmission (either in the first or second sub-phase) is clearly associated with each of the two parts.

$$
\begin{aligned}
x_{3,12}^{4,56} &= A_{23,56}^{3,4} \oplus B_{13,56}^{3,4}, & x_{4,56}^{3,12} &= E_{12,46}^{3,4} \oplus F_{12,45}^{3,4} \\
x_{3,12}^{4,57} &= A_{23,57}^{3,4} \oplus B_{13,57}^{3,4}, & x_{4,57}^{3,12} &= E_{12,47}^{3,4} \oplus G_{12,45}^{3,4} \\
x_{3,12}^{4,67} &= A_{23,67}^{3,4} \oplus B_{13,67}^{3,4}, & x_{4,67}^{3,12} &= F_{12,47}^{3,4} \oplus G_{12,46}^{3,4} \\
x_{3,12}^{5,46} &= A_{23,46}^{3,5} \oplus B_{13,46}^{3,5}, & x_{5,46}^{3,12} &= D_{12,56}^{3,5} \oplus F_{12,45}^{3,5} \\
x_{3,12}^{5,47} &= A_{23,47}^{3,5} \oplus B_{12,47}^{3,5}, & x_{5,47}^{3,12} &= D_{12,57}^{3,5} \oplus G_{12,45}^{3,5} \\
x_{3,12}^{5,67} &= A_{23,67}^{3,5} \oplus B_{13,67}^{3,5}, & x_{5,67}^{3,12} &= F_{12,57}^{3,5} \oplus G_{12,56}^{3,5}.
\end{aligned}
$$

During shuffling sub-phase 2, where now all information for the first set of nodes has been delivered, the first set of nodes will use their computed intermediate values so as to speed up the transmission of the remaining intermediate values toward the second set of nodes. To simplify the second shuffling sub-phase, we will combine together sub-parts of the same upper indices i.e., $\{F''^{k_1,k_2}_{\tau_1,\tau_2}, \forall k_1 \in \mathcal{K}_1 \backslash \tau_1, \forall k_2 \in \tau_2\} \rightarrow F_{\tau_1,\tau_2}$.

Using a scheme similar to that of [56] and denoting with $\mathcal{H}_{\tau,\mu}^{-1}$ the normalized inverse of the channel matrix between transmitting nodes in set τ and receiving nodes in set μ, and with $\mathbf{h}_{\tau,k}$ the channel vector from nodes in set τ to node k, then the 9 transmissions that deliver all remaining data are

$$
\begin{aligned}
x_{45,67}^{\tau} &= \mathcal{H}_{\tau,45}^{-1} \begin{bmatrix} D_{\tau,67} \\ E_{\tau,67} \end{bmatrix} + \mathcal{H}_{\tau,67}^{-1} \begin{bmatrix} F_{\tau,45} \\ G_{\tau,45} \end{bmatrix} \quad \tau \in \{12, 13, 23\} \\
x_{46,57}^{\tau} &= \mathcal{H}_{\tau,46}^{-1} \begin{bmatrix} D_{\tau,57} \\ F_{\tau,57} \end{bmatrix} + \mathcal{H}_{\tau,57}^{-1} \begin{bmatrix} E_{\tau,46} \\ G_{\tau,46} \end{bmatrix} \quad \tau \in \{12, 13, 23\} \\
x_{47,56}^{\tau} &= \mathcal{H}_{\tau,47}^{-1} \begin{bmatrix} D_{\tau,56} \\ G_{\tau,56} \end{bmatrix} + \mathcal{H}_{\tau,56}^{-1} \begin{bmatrix} E_{\tau,47} \\ F_{\tau,47} \end{bmatrix} \quad \tau \in \{12, 13, 23\}.
\end{aligned}
$$

The message arriving at each receiving node takes the form of a linear combination of intermediate values. In order for a node to decode its desired intermediate value, first we can see that matrix $\mathcal{H}_{\tau,\mu}^{-1}$ is designed in such a way so as to separate the transmitted messages at the two nodes, i.e.,

$$
\mathbf{h}_{\tau,k\in\mu}^{T} \cdot \mathcal{H}_{\tau,\mu}^{-1} \cdot \begin{bmatrix} F^1, \\ \vdots \\ F^k, \\ \vdots \\ F^n \end{bmatrix} = W^k.
$$

For example, Node 4 will receive, in the first transmission,

$$
y_{45,67}^{12}(k=4) = D_{12,67} + \mathbf{h}_{12,6}^{T} \cdot \mathcal{H}_{12,67}^{-1} \begin{bmatrix} F_{12,45} \\ G_{12,45} \end{bmatrix}
$$

and then, using CSIR and its computed intermediate values $F_{12,45}$ and $G_{12,45}$, Node 4 will decode its desired intermediate value $D_{12,67}$.

Chapter 9

Conclusion and Open Problems

In this final chapter we will explore some of the ramifications of the presented algorithms along with some open problems.

9.1 Caching gains under Subpacketization

We begin by discussing some interesting examples related to subpacketization. The following note relates to content replication at the transmitter side.

Note 9.1. *As seen in the placement algorithm of Note 3.2 (see also Section 3.3.5), content replication at the transmitter side need not impose a fundamental increase in the required subpacketization. This observation allows the caching gains to be complemented by the multiplexing gains* ***free*** *of any additional subpacketization that would have imposed a further constriction on the Coded Caching gains.*

As we have seen, previous multi-transmitter algorithms required subpacketization that not only was exponential to the caching gains, but was also exponential to the number of transmitters.

Example 9.1 (Multiplicative boost of effective DoF). *Let us assume $K = 1280$ users each with cache of normalized size $\gamma = \frac{1}{20}$, served by an L-antenna base station. In this setting, the theoretical caching gain is $G = K\gamma = 64$ and the theoretical DoF $D_L(K, \gamma) = L + G = L + 64$.*

If the subpacketization limit $S_{\max}$ was infinite, then the effective and theoretical caching gains would match, as we could get $\bar{G}_1 = \bar{G}_L = G = K\gamma = 64$ even when $L = 1$, which would imply only an additive DoF increase from having many antennas.

If, instead, the subpacketization limit was the lesser, but still practically infeasible

$$S_{\max} = \binom{K/2}{K\gamma/2} = \binom{640}{32} = 10^{54} \tag{9.1}$$

then, in the presence of a single antenna, we would encode over $\bar{K} = 640$ users to get a constrained gain of $\bar{G}_1 = \bar{K}\gamma = 640\frac{1}{20} = 32$ which means that,

irrespective of the number of antennas $L \geq 2$, the caching boost offered by the proposed method would yield the multiplicative boost of $\frac{\bar{G}_L}{\bar{G}_1} = 2$.

Let us now consider the more reasonable subpacketization

$$S_{\max} = \binom{80}{4} \approx 1.5 \cdot 10^6 \tag{9.2}$$

which implies that we encode over only $\bar{K} = 80$ users, to get, in the single-antenna case, an effective caching gain $\bar{G}_1 = \bar{K}\gamma = 4$ treating a total of $D_1(\bar{K}, \gamma) = 1 + \bar{G}_1 = 5$ users at a time.

Assume now that we increased the number of transmitting antennas to $L = 2$. Then, using the exact same subpacketization of $S_{\max} = \binom{80}{4}$ and making use of the Algorithm 3.1 would allow to treat

$$D_{L=2}\left(\bar{K} = 80, \gamma = \frac{1}{20}\right) = 2 \cdot D_1\left(\bar{K} = 80, \gamma = \frac{1}{20}\right) = 10 \tag{9.3}$$

users at a time thus, $\bar{G}_L = 8$ and thus providing a $\frac{\bar{G}_L}{\bar{G}_1} = 2$ DoF-boost.

Similarly for $L = 4$ and $L = 16$, which yield a DoF boost of 4 and 16, respectively.

The following example highlights the utility of matching $K\gamma$ with L, and focuses on smaller cache sizes.

Example 9.2. *In a BC with $\gamma = 1/100$ and $L = 1$, allowing for caching gains of $G = K\gamma = 10$ (additional users due to caching), would require*

$$S_1 = \binom{1000}{10} > 10^{23} \tag{9.4}$$

so in practice coded caching could not offer such gains. In the $L = 10$ antenna case, this caching gain comes with subpacketization of only $S_{L=10} = K/L = 100$.

Example 9.3 (Base-station cooperation). *Let us consider a scenario where in a dense urban setting, a single base-station ($K_T = 1$) serves $K = 10^4$ cell-phone users, where each is willing to dedicate 20 GB of its phone's memory for caching parts from a Netflix library of $N = 10^4$ low-definition movies. Each movie is 1 GB in size, and the base-station can store 10 TB. This corresponds to having $M = 20$, $\gamma = M/N = 1/500$, and $\gamma_T = 1$. If $L_T = 1$ (single transmitting antenna), a caching gain of $G = 20$ would have required (given the MN algorithm) subpacketization of*

$$S_1 = \binom{K}{K\gamma} = \binom{10000}{20} > 10^{61}. \tag{9.5}$$

If instead we had two base-stations ($K_T = 2$) with $L_T = 5$ transmitting antennas each, this gain would require subpacketization

$$S_{10} = \binom{\frac{K}{K_T L_T}}{\frac{K\gamma}{K_T L_T}} = \binom{10000/10}{20/10} = \binom{1000}{2} \approx 5 \cdot 10^5 \tag{9.6}$$

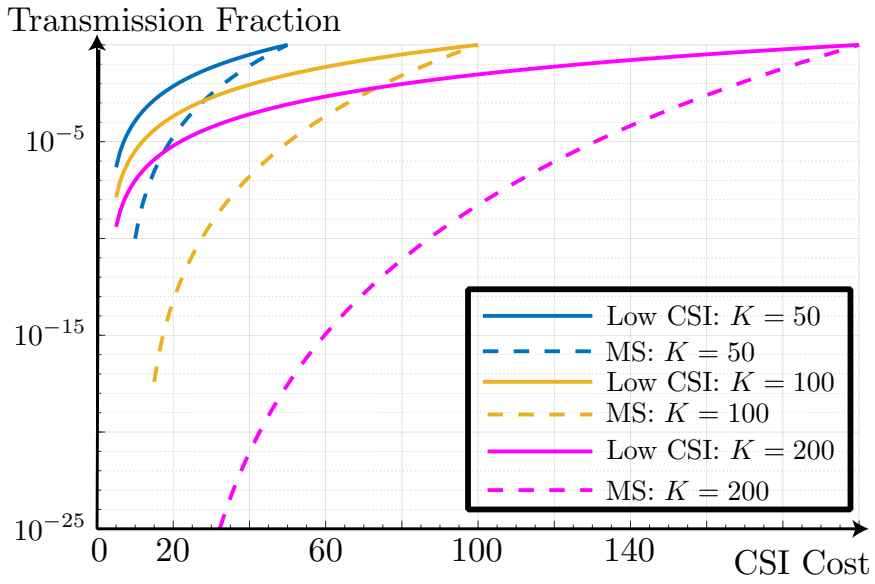

Figure 9.1: The CSI cost (x-axis) that is required to successfully communicate a fraction (y-axis) of the delivery phase using the algorithm in [2] (Low CSI) and the multi-server (MS) algorithm in [10]. The CSI cost represents the number of users that need to communicate feedback (CSIT). Parameter γ is fixed at value $\gamma = \frac{1}{10}$ and number of antennas are $L = 5$ for every plotted scenario. We can see that by simply increasing the number of users, the resulting increase in the CSI cost, for the MS algorithm, to allow for the same fraction to be transmitted, increases in an accelerated manner.

while with $K_T = 4$ such cooperating base-stations, this gain could be achieved with subpacketization $S_{20} = \binom{10000/20}{20/20} = 500$.

If the library is now reduced to the most popular $N = 1000$ movies (and without taking into consideration the cost of cache-misses due to not caching the tail of unpopular files), then the same 20 GB memory at the receivers would correspond to $\gamma = 1/50$ and to a theoretical caching gain of $G = K\gamma = 200$ additional users served per second per hertz. In this case, having a single large-MIMO array with $L_T = 100$ antennas, or having $K_T = 5$ cooperating base-stations with $L_T = 20$ antennas each, would yield a DoF $D_{100}(K, \gamma) = 300$ (caching would allow us to serve 200 additional users at a time), at subpacketization

$$S_{100} = \binom{10000/100}{200/100} = \binom{100}{2} \approx 5000. \tag{9.7}$$

9.2 The CSI curse

Example 9.4. *Let us assume a transmitter with $L = 5$ antennas, serving K users, with each user being equipped with a cache of normalized size $\gamma = \frac{1}{10}$. The goal is to understand the CSI requirements of each algorithm, as well as to compare their ability to reuse existing feedback, and we will do so by*

comparing the fraction of the overall delivery that can be completed, given a certain amount of acquired CSI. In particular, let us assume that the allowable feedback cost[1] *is* C*, where for example in TDD this can imply receiving feedback from only* C *users, corresponding to* C *uplink and* C *downlink feedback acquisition slots. The comparison will be between the proposed algorithm of [2] (where in Figure 9.1 is denoted as "Low CSI") and the state-of-art multi-antenna coded caching algorithms [10, 11] (where in Figure 9.1 are denoted as "MS") . We first note that for the proposed algorithm,* C *needs to satisfy* $C \geq L = 5$*, while for the state-of-art algorithms,* C *needs to satisfy* $C \geq L + K\gamma = 5 + \frac{K}{10}$*.*

To understand the ability of the state of art algorithms to reuse feedback, let us recall from [10, 11] that once feedback is acquired for a set of $C \geq L + K\gamma$ *users, then there are*

$$(K\gamma + L)\binom{C}{L + K\gamma}\binom{L + K\gamma - 1}{K\gamma} \tag{9.8}$$

transmissions[2] *that can take place without the need for additional feedback.*

On the other hand, by observing the proposed algorithm (see Algorithm 4.2), we can see that having feedback for some subset of $C \geq L$ *users allows for*

$$L \cdot \binom{C}{L}\binom{K - L}{K\gamma} \tag{9.9}$$

transmissions. Now comparing Eq. (9.8) with Eq. (9.9), and given that each transmission in both cases carries the same amount of information, we can conclude that the new algorithm can serve a much larger portion of the delivery. Specifically the new algorithm can serve, for a given CSI cost C*,*

$$\frac{L \cdot \binom{C}{L}\binom{K-L}{K\gamma}}{(K\gamma + L)\binom{C}{L+K\gamma}\binom{L+K\gamma-1}{K\gamma}} \overset{(\star)}{\approx} \left(\frac{L + K\gamma}{L}\right)^{L} \left(\frac{K - L}{C}\right)^{K\gamma}$$

times more content than the existing algorithms, where in $(\star)$ *we used the Sterling approximation*

$$\binom{n}{k} \approx \left(\frac{n}{k}\right)^{k}. \tag{9.10}$$

The comparison of the algorithms is illustrated in Figure 9.1, where we show the CSI cost needed to complete any fraction of the entire delivery.

For example, delivering the fraction of 10^{-5} *in the case of* $K = 50$*, we see that the proposed algorithm requires feedback cost* $C_{low} = 7$*, while the state-of-art (MS) requires a cost of approximately* $C_{MS} = 18$*. This effect is further*

[1] For simplicity we assume that the coherence block is long enough to fit the transmission of this particular portion that we aim to complete. In other words, the time frame of this comparison here is such that we do not have to worry about users having to renew their CSI because the coherence period has elapsed.

[2] Where we added the first term in order to match the subpacketization of the proposed algorithm. As a result, in either of the compared algorithms, one transmission carries the same amount of information.

amplified when we visit cases with higher number of users. Specifically, for the same fraction 10^{-5}*, when the number of users is* $K = 100$ *the respective costs are* $C_{low} = 12$ *and* $C_{MS} = 52$*, while in the case of* $K = 200$ *receivers, delivering fraction* 10^{-5} *would rise further the aforementioned feedback costs to* $C_{low} = 22$ *and* $C_{MS} = 131$*, respectively.*

9.3 Subpacketization savings and low CSI

The work in [80] combined, for the first time, a subpacketization requirement exponentially smaller than the single-stream setting[3], with CSI cost that is untangled from the number of users and only depends on the number of antennas. From Remark 4.1 we can see that the number of antennas plays a pivotal role in the subpacketization reduction. While on the one hand the CSI cost increases linearly with the number of antennas, on the other hand, the subpacketization decreases exponentially.

As it clear, when $L = 1$ there is no subpacketization reductions, but further increasing the number of antennas provides approximately a factor e^L decrease on the subpacketization (see also Remark 4.1). The subpacketization reductions are also illustrated in Figure 9.2.

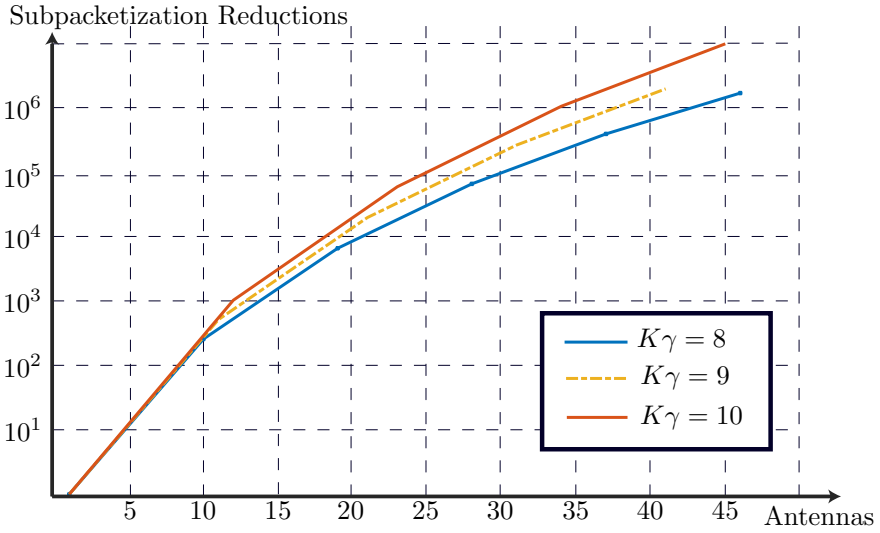

Figure 9.2: The CSI cost that is required to complete a fraction of the delivery phase. The cost represents the number of users that need to send feedback. Parameter γ is fixed at value $\gamma = \frac{1}{10}$.

[3]This reduction provides even further exponential subpacketization savings compared to other multi-antenna algorithms, with the notable exception of the algorithms in [56] and Section 4.2.

L	CSI		Subpacketization		DoF	
	MN [1]	JCS [80]	MN [1]	JCS [80]	MN [1]	JCS [80]
12	0	12	10^{14}	$8 \cdot 10^{10}$	11	20
23	0	23	10^{14}	$8 \cdot 10^{8}$	11	30
34	0	34	10^{14}	$3 \cdot 10^{7}$	11	40

Table 9.1: Comparison of the single-antenna scheme of [1] and the JCS scheme of [80] in terms of CSI cost, achieved DoF and Subpacketization requirements for various antenna values for the case of $K = 120$ users equipped with caches of fractional size $\gamma = \frac{1}{10}$.

Subpacketization, CSI cost and DoF

It is interesting to note, that the CSI and subpacketization benefits are provided with a small DoF reduction which, in practical scenarios where the cache size is small, will amount to an insignificant performance degradation.

Example 9.5. *Let us assume the L-antenna MISO BC with $K = 120$ single antenna, cache-aided users, each equipped with a cache of normalized size $\gamma = \frac{1}{10}$. For this setting of interest, the algorithm of [1] would require subpacketization of $S_1 = \binom{120}{10} = 10^{14}$ and would achieve DoF $D_1(K, \gamma) = 11$.*

Through the use of the algorithm in [80] we can both harness the DoF benefits of multiple-antennas and at the same time reduce the subpacketization. For example, adding 11 antennas i.e. $L = 12$, would provide the DoF of $D_{L,JCS}(K, \gamma) = 20$, and subpacketization $S_{L,JCS} = 8 \cdot 10^{10}$, i.e. a reduction of more than 3 orders of magnitude.

We can see that further increase of the number of the antennas leads to an increase of the DoF and also reduces the subpacketization. Some further values are illustrated in Table 9.1.

9.4 Uneven caches and multiple antennas

As we have seen, removing the cache-size unevenness bottleneck in the single antenna setting is fundamentally impossible when cache-less users are in the same channel as cache-aided ones, while if all users are cache-aided, but with uneven cache-sizes, any known single antenna algorithm performs worse than the homogeneous case.

Examining, though, the multiple antenna channel shows that this performance degradation can be completely resolved for a wide range of parameters, even if cache-less users appear in the channel. This shows the vital role of multiple antenna systems, especially in the transitional period of implementing cache-aided communications, where some users will be able to take advantage of coded caching, while others won't.

It is also interesting to note, that when cache-aided users coexist with cache-less users, even if the homogeneous performance can not be achieved,

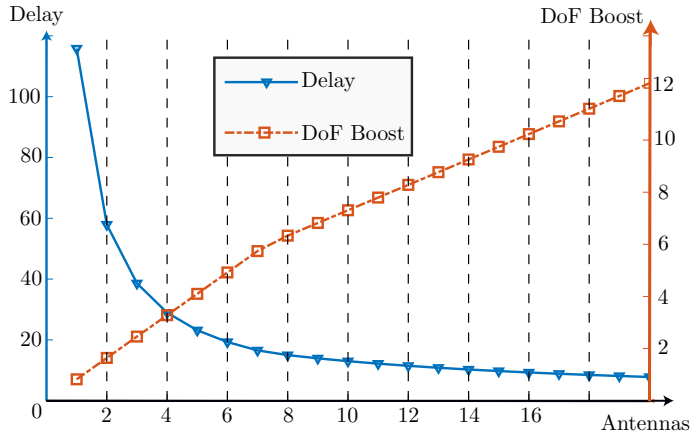

Figure 9.3: The improvement in the delay and the DoF boost that are achieved as a function of the number of antennas for a scenario with $K_1 = 100$ cache-aided users with normalized cache-size $\gamma_1 = \frac{1}{20}$ and $K_2 = 120$ cache-less users. We can see that the DoF boost slope becomes smaller after $L = 7$ antennas, showing that any cache-size unevenness effect has been completely removed, thus the DoF continues to increase additively instead of multiplicatively.

nonetheless by increasing the number of antennas one can achieve a multiplicative boost in the performance equal to the number of antennas. This performance improvement is depicted in Figure 9.3, showing that adding one more antenna can double the DoF or, else, half the delivery time.

9.5 How to resolve the CSI bottleneck

As discussed in Chapter 4, the cost of feedback acquisition is not only described by the cost of transmitting a single vector, but can also be seen to accumulate because of the rapid introduction of new users that will need to send CSI. In particular, if we consider a single coherence period and any multi-antenna Coded Caching algorithm, then in each iteration a new subset of users is selected, which new user subset will overlap only partially with the previously selected subset, thus more CSI will be required to accommodate the transmission.

This effect is a direct consequence of Coded Caching algorithms that require in each iteration a new user subset to be selected. Furthermore, we can discern another effect that amplifies the CSI bottleneck i.e., the high subpacketization.

It is clear that if subfiles were long enough, approximately in the order of magnitude of the length of the coherence period, then the CSI cost would be approximately $L + K\gamma$ and thus pose a lesser bottleneck. On the other hand, when each subfile is small then, there are many iterations of the algorithm, which requires encoding over many user subsets and thus a great amount of CSI acquisition needs to take place.

From the above, it remains clear that the road to reduced CSI algorithms entails two important elements, namely

- High CSI reuse, and
- Low subpacketization.

9.5.1 Subpacketization - CSI tradeoff

An interesting direction of research lies in the design of algorithms that can simultaneously achieve low CSI requirements and low Subpacketization. While this is the ultimate goal nevertheless, it is not clear whether this can be achieved, thus focus should also be placed in the design of algorithms that can trade-off the performance of each of these two quantities.

9.5.2 Distributed Computing Bottlenecks

Distributed computing

The distributed computing algorithm of [14, 15], as discussed above, uses the increased redundancy from mapping (bringing a delay increase in the mapping phase) to counteract the effect from a high shuffling phase. The higher the time spent mapping means a higher redundancy at the nodes, which amounts to a higher multicasting gain (decrease in shuffling) in the shuffling phase. This creates a trade-off between increasing the redundancy and decreasing the shuffling cost, which minimizes the total delay when $\gamma = \gamma^\star$.

It is the case though that when subpacketization becomes too large, the CMR algorithm cannot increase the shuffling gains, thus the total execution time will remain almost unaffected, as the redundancy parameter γ increases. This is depicted in Figure 9.4.

We can see that for small values of γ, the CMR algorithm can exactly achieve the theoretical delay $T_{\text{tot}}(\gamma)$, and can also achieve the exact delay for large values of γ, simply because

$$S_{\max} \leq \binom{K}{K\gamma} \quad \Leftrightarrow \quad S_{\max} \leq \binom{K - K\gamma}{K\gamma}. \tag{9.11}$$

The reality, though, is that the optimal value $\gamma^\star$ lies in some "moderate" (nor too small, neither too big) region. When this optimal tradeoff point $\gamma^\star$ is unreachable by the CMR algorithm, due to subpacketization constraints, then the algorithm cannot be fine-tuned to achieve the optimal performance. On the other hand, making use of the GCMR algorithm can achieve this point, when the cooperation parameter can be large enough to allow for small subpacketization costs.

As we have seen, in the broadcast scenario of Coded Caching it was required an increase in the dimensionality in order to achieve a decrease in the

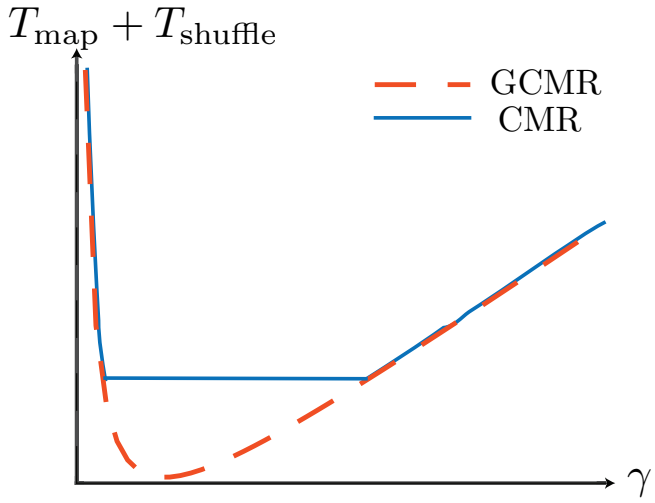

Figure 9.4: The required combined mapping and shuffle delay as a function of redundancy parameter γ for the CMR algorithm and the GCMR algorithms.

subpacketization nevertheless, as evident from Section 8.2, this dimensionality is inherent in the wireless distributed computing (D2D) setting, while also it can be achieved easily in the wired distributed computing setting.

Category Unevenness Effect In the following example we will see yet another aspect — referred to here as the uneven-category effect — which again has the potential to reduce the CMR gains.

Example 9.6. *Let us assume a sorting job that will be completed using the CMR algorithm by $K = 3$ computing nodes. For this job we will assume the redundancy $\gamma = \frac{2}{3}$, thus the whole dataset will be mapped a total of 2 times across the nodes, yielding the theoretical gain for the shuffling phase of 2. Let us, further, assume that the dataset F is comprized of the following elements*

$$F = \{24, 27, 3, 27, 18, 2, 8, 16, 28, 28, 4, 29, 28, 14, 24, 4, 12, 27\}. \tag{9.12}$$

The CMR algorithm would need to break this dataset into $S = 3$ segments F_{12}, F_{13}, F_{23} as follows

$$\begin{aligned} F_{12} &= \{24, 27, 3, 27, 18, 2\} \\ F_{13} &= \{8, 16, 28, 28, 4, 29\} \\ F_{23} &= \{28, 14, 24, 4, 12, 27\}. \end{aligned}$$

and assign them to the nodes.

Mapping Phase *The mapping phase will produce the following intermediate values*

$$
\begin{aligned}
F_{12}^{1} &= \{3, 2\} \\
F_{12}^{2} &= \{18\} \\
F_{12}^{3} &= \{24, 27, 27\} \\
F_{13}^{1} &= \{8, 4\} \\
F_{13}^{2} &= \{16\} \\
F_{13}^{3} &= \{28, 28, 29\} \\
F_{23}^{1} &= \{4\} \\
F_{23}^{2} &= \{14, 12\} \\
F_{23}^{3} &= \{28, 24, 27\}
\end{aligned}
$$

where we can see the unevenness, in terms of elements which would also correspond to uneven bit-sizes, of some of the intermediate values.

Shuffle *In the shuffling phase, the transmitted messages are the following*[4]

$$
\begin{aligned}
&\textit{Node } 1\textit{:} \; F_{12}^{3} \oplus F_{13}^{2} \rightarrow \quad |F_{12}^{3} \oplus F_{13}^{2}| = 3 \\
&\textit{Node } 2\textit{:} \; F_{12}^{3} \oplus F_{23}^{1} \rightarrow \quad |F_{12}^{3} \oplus F_{23}^{1}| = 2 \\
&\textit{Node } 3\textit{:} \; F_{13}^{2} \oplus F_{23}^{1} \rightarrow \quad |F_{13}^{2} \oplus F_{23}^{1}| = 3.
\end{aligned}
$$

We can notice that many of the above intermediate values have different numbers of elements, something that creates the need to zero-pad them, which would mean a much smaller effective shuffling gain. In this specific case, by adding the amounts of elements that have been transmitted we have the equivalent of $\frac{3+2+3}{2} = 4$ *elements that will be communicated, while the theoretical performance is*

$$
\frac{f - f\gamma}{K\gamma} = \frac{6}{2} = 3 \tag{9.13}
$$

which amounts to a "slow-down" of 33%*.*

[4] We note that intermediate values are not transmitted twice, but instead are broken into two smaller packets and each node transmits one of those packets. Thus the XOR sizes that are presented are, in reality, half.

Bibliography

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, pp. 2856–2867, May 2014.

[2] E. Lampiris and P. Elia, "Resolving a feedback bottleneck of multi-antenna coded caching," *arXiv preprint arXiv:1811.03935*, 2018.

[3] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, pp. 8402–8413, Dec 2013.

[4] P. Sermpezis, T. Spyropoulos, L. Vigneri, and T. Giannakas, "Femto-caching with soft cache hits: Improving performance with related content recommendation," in *GLOBECOM IEEE Global Communications Conference*, pp. 1–7, Dec 2017.

[5] N. Golrezaei, P. Mansourifard, A. F. Molisch, and A. G. Dimakis, "Base-station assisted device-to-device communications for high-throughput wireless video networks," *IEEE Transactions on Wireless Communications*, vol. 13, pp. 3665–3676, July 2014.

[6] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *2015 IEEE International Conference on Communications (ICC)*, pp. 3358–3363, June 2015.

[7] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Transactions on Communications*, vol. 62, pp. 3665–3677, Oct 2014.

[8] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 135–139, July 2016.

[9] 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, pp. 1281–1296, Feb 2018.

[10] 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.

[11] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Transactions on Information Theory*, vol. 63, pp. 3092–3107, May 2017.

[12] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Transactions on Information Theory*, vol. 62, pp. 849–869, Feb 2016.

[13] Ç. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of D2D coded caching with uncoded cache placement and one-shot delivery," *arXiv preprint arXiv:1901.05921*, 2019.

[14] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coded MapReduce," in *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 964–971, Sep. 2015.

[15] 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, pp. 109–128, Jan 2018.

[16] S. Li, S. Supittayapornpong, M. A. Maddah-Ali, and S. Avestimehr, "Coded terasort," in *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 389–398, May 2017.

[17] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "Fundamental limits of distributed data shuffling," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 662–669, Oct 2018.

[18] L. Song, C. Fragouli, and T. Zhao, "A pliable index coding approach to data shuffling," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 2558–2562, IEEE, 2017.

[19] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Caching in combination networks," in *49th Asilomar Conference on Signals, Systems and Computers*, pp. 1269–1273, Nov 2015.

[20] L. Tang and A. Ramamoorthy, "Coded caching for networks with the resolvability property," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 420–424, July 2016.

[21] A. A. Zewail and A. Yener, "Coded caching for combination networks with cache-aided relays," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2433–2437, June 2017.

[22] M. Ji, M. F. Wong, A. M. Tulino, J. Llorca, G. Caire, M. Effros, and M. Langberg, "On the fundamental limits of caching in combination networks," in *IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 695–699, June 2015.

[23] J. Zhang and O. Simeone, "Fundamental limits of cloud and cache-aided interference management with multi-antenna edge nodes," *IEEE Transactions on Information Theory*, pp. 1–1, 2019.

[24] J. S. P. Roig, F. Tosato, and D. Gündüz, "Storage-latency trade-off in cache-aided fog radio access networks," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[25] A. Roushdy, A. S. Motahari, M. Nafie, and D. Gündüz, "Cache-aided fog radio access networks with partial connectivity," in *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, April 2018.

[26] R. Karasik, O. Simeone, and S. Shamai, "How much can D2D communication reduce content delivery latency in fog networks with edge caching?," *CoRR*, vol. abs/1904.01256, 2019.

[27] S. P. Shariatpanahi, J. Zhang, O. Simeone, B. H. Khalaj, and M. A. Maddah-Ali, "Cloud-aided interference management with cache-enabled edge nodes and users," *CoRR*, vol. abs/1901.06698, 2019.

[28] M. Tao, D. Gündüz, F. Xu, and J. S. P. Roig, "Content caching and delivery in wireless radio access networks," *IEEE Transactions on Communications*, pp. 1–1, 2019.

[29] F. Xu and M. Tao, "Fundamental limits of decentralized caching in fog-rans with wireless fronthaul," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1430–1434, June 2018.

[30] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," *IEEE Transactions on Information Theory*, vol. 62, pp. 3212–3229, June 2016.

[31] J. Hachem, N. Karamchandani, and S. Diggavi, "Multi-level coded caching," in *IEEE International Symposium on Information Theory*, pp. 56–60, June 2014.

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

[33] A. Sengupta, R. Tandon, and T. C. Clancy, "Improved approximation of storage-rate tradeoff for caching via new outer bounds," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1691–1695, June 2015.

[34] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," *IEEE Transactions on Information Theory*, vol. 63, pp. 4388–4413, July 2017.

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

[36] C. Wang, S. Saeedi Bidokhti, and M. Wigger, "Improved converses and gap results for coded caching," *IEEE Transactions on Information Theory*, vol. 64, pp. 7051–7062, Nov 2018.

[37] 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, pp. 647–663, Jan 2019.

[38] M. Cheng, J. Jiang, Q. Yan, and X. Tang, "Constructions of Coded Caching schemes with flexible memory size," *IEEE Transactions on Communications*, vol. 67, pp. 4166–4176, June 2019.

[39] 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, pp. 5524–5537, Oct 2016.

[40] S.-E. Elayoubi and J. Roberts, "Performance and cost effectiveness of caching in mobile access networks," in *Proceedings of the 2Nd ACM Conference on Information-Centric Networking*, ACM-ICN '15, (New York, NY, USA), pp. 79–88, ACM, 2015.

[41] S. P. Shariatpanahi, G. Caire, and B. Hossein Khalaj, "Physical-layer schemes for wireless coded caching," *IEEE Transactions on Information Theory*, vol. 65, pp. 2792–2807, May 2019.

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

[43] A. Tolli, S. P. Shariatpanahi, J. Kaleva, and B. Khalaj, "Multicast beamformer design for coded caching," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1914–1918, June 2018.

[44] L. Zheng and D. N. C. Tse, "Communication on the Grassmann manifold: a geometric approach to the noncoherent multiple-antenna channel," *IEEE Transactions on Information Theory*, vol. 48, pp. 359–383, Feb 2002.

[45] S. A. Jafar and S. Vishwanath, "Generalized Degrees of Freedom of the symmetric Gaussian K user Interference Channel," *IEEE Transactions on Information Theory*, vol. 56, pp. 3297–3303, July 2010.

[46] A. Gholami Davoodi and S. A. Jafar, "Aligned image sets under channel uncertainty: Settling conjectures on the collapse of degrees of freedom under finite precision CSIT," *IEEE Transactions on Information Theory*, vol. 62, pp. 5603–5618, Oct 2016.

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

[48] A. D. Wyner, "Shannon-theoretic approach to a gaussian cellular multiple-access channel," *IEEE Transactions on Information Theory*, vol. 40, pp. 1713–1727, Nov 1994.

[49] A. Gholami Davoodi and S. A. Jafar, "Generalized degrees of freedom of the symmetric k user interference channel under finite precision CSIT," *IEEE Transactions on Information Theory*, vol. 63, pp. 6561–6572, Oct 2017.

[50] O. O'Malley, "Terabyte sort on apache hadoop," *Available online at: http://sortbenchmark.org/Yahoo-Hadoop.pdf*, pp. 1–3, May 2008.

[51] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, pp. 1029–1040, Aug 2015.

[52] 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, pp. 5821–5833, Sep. 2017.

[53] L. Tang and A. Ramamoorthy, "Coded caching schemes with reduced subpacketization from linear block codes," *IEEE Transactions on Information Theory*, vol. 64, pp. 3099–3120, April 2018.

[54] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Transactions on Information Theory*, vol. 64, pp. 5755–5766, Aug 2018.

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

[56] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 36, pp. 1176–1188, June 2018.

[57] A. Lozano, R. W. Heath Jr, and J. G. Andrews, "Fundamental limits of cooperation," *arXiv preprint arXiv:1204.0011*, 2012.

[58] G. Caire and S. Shamai, "On the achievable throughput of a multiantenna gaussian broadcast channel," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1691–1706, 2003.

[59] S. A. Jafar and A. J. Goldsmith, "Isotropic fading vector broadcast channels: The scalar upper bound and loss in degrees of freedom," *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 848–857, 2005.

[60] C. Huang, S. A. Jafar, S. Shamai, and S. Vishwanath, "On degrees of freedom region of MIMO networks without channel state information at transmitters," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 849–857, 2012.

[61] A. Lapidoth, S. Shamai, and M. Wigger, "On the capacity of fading MIMO Broadcast Channels with imperfect transmitter side-information," *arXiv preprint cs/0605079*, 2006.

[62] C. S. Vaze and M. K. Varanasi, "The degree-of-freedom regions of MIMO Broadcast, Interference, and cognitive radio channels with no CSIT," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5354–5374, 2012.

[63] H. Weingarten, S. Shamai, and G. Kramer, "On the compound MIMO Broadcast Channel," in *Proceedings of Annual Information Theory and Applications Workshop UCSD*, Citeseer, 2007.

[64] T. Gou, S. A. Jafar, C. Wang, *et al.*, "On the degrees of freedom of finite state compound wireless networks.," *IEEE Trans. Information Theory*, vol. 57, no. 6, pp. 3286–3308, 2011.

[65] M. A. Maddah-Ali, "On the degrees of freedom of the compound MISO Broadcast Channels with finite states," in *International Symposium on Information Theory Proceedings (ISIT)*, pp. 2273–2277, IEEE, 2010.

[66] M. Maddah-Ali and D. Tse, "Completely stale transmitter channel state information is still very useful," *Information Theory, IEEE Transactions on*, 2012.

[67] S. A. Jafar, "Blind interference alignment," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 3, pp. 216–227, 2012.

[68] S. Yang, M. Kobayashi, D. Gesbert, and X. Yi, "Degrees of Freedom of time correlated MISO broadcast channel with delayed CSIT," *IEEE Transactions on Information Theory*, vol. 59, no. 1, pp. 315–328, 2013.

[69] T. Gou and S. A. Jafar, "Optimal use of current and outdated channel state information: Degrees of freedom of the miso bc with mixed csit," *IEEE Communications Letters*, vol. 16, no. 7, pp. 1084–1087, 2012.

[70] J. Chen and P. Elia, "Degrees-of-freedom region of the miso broadcast channel with general mixed-csit," *arXiv preprint arXiv:1205.3474*, 2012.

[71] J. Chen and P. Elia, "Toward the performance vs. feedback tradeoff for the two-user miso broadcast channel," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8336–8356, 2013.

[72] R. Tandon, S. A. Jafar, S. Shamai, and H. V. Poor, "On the synergistic benefits of alternating csit for the miso broadcast channel," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4106–4128, 2013.

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

[74] M. Salehi, A. Tölli, S. P. Shariatpanahi, and J. Kaleva, "Subpacketization-rate trade-off in multi-antenna coded caching," *arXiv preprint arXiv:1905.04349*, 2019.

[75] J. Zhang, F. Engelmann, and P. Elia, "Coded caching for reducing csit-feedback in wireless communications," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1099–1105, Sep. 2015.

[76] E. Piovano, H. Joudeh, and B. Clerckx, "On coded caching in the overloaded miso broadcast channel," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2795–2799, June 2017.

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

[78] E. Piovano, H. Joudeh, and B. Clerckx, "Generalized degrees of freedom of the symmetric cache-aided miso broadcast channel with partial csit," *IEEE Transactions on Information Theory*, pp. 1–1, 2019.

[79] K. Ngo, S. Yang, and M. Kobayashi, "Scalable content delivery with coded caching in multi-antenna fading channels," *IEEE Transactions on Wireless Communications*, vol. 17, pp. 548–562, Jan 2018.

[80] E. Lampiris and P. Elia, "Bridging two extremes: Multi-antenna coded caching with reduced subpacketization and CSIT," *SPAWC*, 2019.

[81] E. Lampiris and P. Elia, "Full coded caching gains for cache-less users," in *IEEE Information Theory Workshop (ITW)*, pp. 1–5, Nov 2018.

[82] G. Agnarsson and R. Greenlaw, *Graph theory: Modeling, applications, and algorithms*. Prentice-Hall, Inc., 2007.

[83] A. Goel, M. Kapralov, and S. Khanna, "Perfect matchings in $\mathcal{O}(n \log n)$ time in regular bipartite graphs," *SIAM Journal on Computing*, vol. 42, no. 3, pp. 1392–1404, 2013.

[84] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Transactions on Information Theory*, vol. 57, pp. 1479–1494, March 2011.

[85] M. Mohammadi Amiri, Q. Yang, and D. Gündüz, "Decentralized caching and coded delivery with distinct cache capacities," *IEEE Transactions on Communications*, vol. 65, pp. 4657–4669, Nov 2017.

[86] A. Sengupta, R. Tandon, and T. C. Clanc, "Layered caching for heterogeneous storage," in *IEEE 50th Asilomar Conference on Signals, Systems and Computers*, 2016.

[87] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Coded caching for heterogeneous systems: An optimization perspective," *arXiv preprint arXiv:1810.08187*, 2018.

[88] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Device-to-device coded caching with heterogeneous cache sizes," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[89] D. Cao, D. Zhang, P. Chen, N. Liu, W. Kang, and D. Gündüz, "Coded caching with heterogeneous cache sizes and link qualities: The two-user case," *arXiv preprint arXiv:1802.02706*, 2018.

[90] B. Asadi, L. Ong, and S. J. Johnson, "Centralized caching with unequal cache sizes," in *IEEE Inf. Theory Workshop (ITW)*, Nov 2018.

[91] S. Wang, W. Li, X. Tian, and H. Liu, "Fundamental limits of heterogenous cache," *arXiv preprint arXiv:1504.01123*, 2015.

[92] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Centralized coded caching with heterogeneous cache sizes," in *Wireless Communications and Networking Conference (WCNC)*, IEEE, 2017.

[93] K. Fukuda and T. Matsui, "Finding all the perfect matchings in bipartite graphs," *Applied Mathematics Letters*, vol. 7, no. 1, pp. 15 – 18, 1994.

[94] M. Li, L. Ong, and S. J. Johnson, "Improved bounds for multi-sender index coding," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 3060–3064, June 2017.

[95] E. Parrinello, A. Ünsal, and P. Elia, "Coded caching with shared caches: Fundamental limits with uncoded prefetching," *arXiv preprint arXiv:1809.09422*, 2018.

[96] A. Tölli, S. P. Shariatpanahi, J. Kaleva, and B. Khalaj, "Multi-antenna interference management for coded caching," *arXiv preprint arXiv:1711.03364*, 2017.

[97] J. Zhao, M. M. Amiri, and D. Gündüz, "A low-complexity cache-aided multi-antenna content delivery scheme," *arXiv preprint arXiv:1903.03856*, 2019.

[98] A. Destounis, M. Kobayashi, G. Paschos, and A. Ghorbel, "Alpha fair coded caching," in *15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 1–8, May 2017.

[99] Y. Cao and M. Tao, "Treating content delivery in multi-antenna coded caching as general message sets transmission: A dof region perspective," *IEEE Transactions on Wireless Communications*, 2019.

[100] A. Ghorbel, M. Kobayashi, and S. Yang, "Content delivery in erasure broadcast channels with cache and feedback," *IEEE Transactions on Information Theory*, vol. 62, pp. 6407–6422, Nov 2016.

[101] M. Mohammadi Amiri and D. Gündüz, "Cache-aided content delivery over erasure broadcast channels," *IEEE Transactions on Communications*, vol. 66, pp. 370–381, Jan 2018.

[102] S. Kamel, M. Sarkiss, and M. Wigger, "Decentralized joint cache-channel coding over erasure broadcast channels," in *IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pp. 1–6, April 2018.

[103] S. Kim, S. Mohajer, and C. Suh, "Coding across heterogeneous parallel erasure broadcast channels is useful," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1883–1887, June 2017.

[104] S. Saeedi Bidokhti, M. Wigger, and R. Timo, "Noisy broadcast networks with receiver caching," *IEEE Transactions on Information Theory*, vol. 64, pp. 6996–7016, Nov 2018.

[105] L. Zheng, Z. Wang, Q. Yan, Q. Chen, and X. Tang, "On the coded caching based wireless video transmission scheme," in *IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1–6, July 2016.

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

[107] A. Ghorbel, K. Ngo, R. Combes, M. Kobayashi, and S. Yang, "Opportunistic content delivery in fading broadcast channels," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, Dec 2017.

[108] N. D. Sidiropoulos, T. N. Davidson, and Zhi-Quan Luo, "Transmit beamforming for physical-layer multicasting," *IEEE Transactions on Signal Processing*, vol. 54, pp. 2239–2251, June 2006.

[109] I. Bergel and S. Mohajer, "Cache-aided communications with multiple antennas at finite SNR," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 1682–1691, Aug 2018.

[110] A. D. Wyner, "Shannon-theoretic approach to a gaussian cellular multiple-access channel," *IEEE Transactions on Information Theory*, vol. 40, pp. 1713–1727, Nov 1994.

[111] E. Lampiris, A. E. Gamal, and P. Elia, "Wyner's network on caches: Combining receiver caching with a flexible backhaul," *IEEE International Symposium on Information Theory (ISIT)*, 2019.

[112] M. Wigger, R. Timo, and S. Shamai, "Complete interference mitigation through receiver-caching in wyner's networks," in *IEEE Information Theory Workshop (ITW)*, pp. 335–339, Sep. 2016.

[113] F. Xu and M. Tao, "Cache-aided interference management in partially connected wireless networks," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, Dec 2017.

[114] X. Yi and G. Caire, "Topological coded caching," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2039–2043, July 2016.

[115] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Cache-aided interference management in wireless cellular networks," *IEEE Transactions on Communications*, vol. 67, pp. 3376–3387, May 2019.

[116] V. V. Veeravalli and A. El Gamal, *Interference Management in Wireless Networks: Fundamental Bounds and the Role of Cooperation*. Cambridge University Press, 2018.

[117] A. El Gamal and V. V. Veeravalli, "Flexible backhaul design and degrees of freedom for linear interference networks," in *IEEE International Symposium on Information Theory*, pp. 2694–2698, June 2014.

[118] A. E. Gamal, V. S. Annapureddy, and V. V. Veeravalli, "Degrees of freedom (dof) of locally connected interference channels with coordinated multi-point (comp) transmission," in *IEEE International Conference on Communications (ICC)*, pp. 2293–2297, June 2012.

[119] Te Han and K. Kobayashi, "A new achievable rate region for the interference channel," *IEEE Transactions on Information Theory*, vol. 27, pp. 49–60, January 1981.

[120] I. Maric, R. Dabora, and A. J. Goldsmith, "Relaying in the presence of interference: Achievable rates, interference forwarding, and outer bounds," *IEEE Transactions on Information Theory*, vol. 58, pp. 4342–4354, July 2012.

[121] 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)*, pp. 1291–1295, June 2018.

[122] E. Lampiris, D. Jiménez Zorrilla, and P. Elia, "Mapping heterogeneity does not affect wireless Coded MapReduce," *IEEE International Symposium on Information Theory (ISIT)*, 2019.

[123] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008.

[124] K. Shim, "MapReduce algorithms for big data analysis," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2016–2017, 2012.

[125] A. Kumar, M. Kiran, and B. Prathap, "Verification and validation of MapReduce program model for parallel k-means algorithm on hadoop cluster," in *Computing, Communications and Networking Technologies (ICCCNT), Fourth International Conference on*, IEEE, 2013.

[126] J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[127] K. Slagter, C.-H. Hsu, Y.-C. Chung, and D. Zhang, "An improved partitioning mechanism for optimizing massive data analysis using MapReduce," *The Journal of Supercomputing*, vol. 66, no. 1, pp. 539–555, 2013.

[128] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, pp. 1514–1529, March 2018.

[129] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "The case for evaluating MapReduce performance using workload suites," in *IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 390–399, July 2011.

[130] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding," *arXiv preprint arXiv:1612.03301*, 2016.

[131] Q. Yu, M. Maddah-Ali, and S. Avestimehr, "Polynomial codes: an optimal design for high-dimensional coded matrix multiplication," in *Advances in Neural Information Processing Systems*, 2017.

[132] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *Int. Symp. on Inf. Theory (ISIT)*, IEEE, 2017.

[133] A. B. Das, L. Tang, and A. Ramamoorthy, "C3LES: Codes for coded computation that leverage stragglers," *arXiv preprint arXiv:1809.06242*, 2018.

[134] C. Suh and K. Ramchandran, "Exact-repair MDS codes for distributed storage using interference alignment," in *Int. Symp. on Inf. Theory (ISIT)*, June 2010.

[135] Q. Yu, N. Raviv, J. So, and A. S. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security and privacy," *arXiv preprint arXiv:1806.00939*, 2018.

[136] F. Li, J. Chen, and Z. Wang, "Wireless mapreduce distributed computing," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1286–1290, June 2018.

[137] J. Zhang and O. Simeone, "Improved latency-communication trade-off for map-shuffle-reduce systems with stragglers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8172–8176, May 2019.

[138] Q. Yan, S. Yang, and M. Wigger, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," in *IEEE Information Theory Workshop (ITW)*, pp. 1–5, Nov 2018.