POLITECNICO DI TORINO

Department of Electronics and Telecommunications

Master Degree Course in Communications and Computer Networks Engineering

Master Degree Thesis

Coded Caching and Cooperation to Boost Communication

Applications to Wireless Networks and Distributed Computing

Supervisor
Prof. Roberto Garello

Candidate
Emanuele Parrinello

Industrial Supervisors
Prof. Petros Elia
Eleftherios Lampiris

March 2018

Acknowledgements

Questo lavoro di tesi è stato svolto alla conclusione di un percorso di doppia laurea tra il Politecnico di Torino, dove ho frequentato il percorso di laurea magistrale in Communications and Computer Networks Engineering, e Eurecom, dove ho seguito la track Mobile Communications. Il lavoro di tesi è stato svolto nel periodo di luglio-dicembre 2017 a Eurecom (Biot, Francia). Mi sembra dunque doveroso ringraziare tutti coloro che nel loro piccolo mi hanno accompagnato in questa esperienza universitaria.

Vorrei innanzitutto ringraziare il professore Petros Elia e il mio collega Eleftherios Lampiris con cui ho lavorato per questo lavoro di tesi. Grazie anche al professore Roberto Garello che mi ha seguito da Torino.

Il piú speciale dei ringraziamenti va ai miei genitori che mi hanno sostenuto e guidato in questo percorso di vita. È grazie a loro che mi sono prima potuto trasferire a Torino e qualche anno dopo ad Antibes per concludere i miei studi. È grazie ai loro insegnamenti che ho superato con successo tutti i momenti di difficoltá riscontrati in questo percorso. Un grazie speciale va anche a mio fratello che è sempre stato un punto di riferimento e ha sempre creduto in me. A voi, mamma, papá e Giorgio, dedico questa tesi.

Grazie ai miei cari nonni, zii e cugini che mi hanno sempre supportato da lontano ed esultanto per ogni mio piccolo traguardo.

Un ringraziamento va anche a tutti i miei amici: quelli "storici" di Marsala, quelli conosciuti nella residenza universitaria Borsellino, quelli ormai sparsi nel mondo conosciuti durante l'Erasmus e quelli con cui trascorro la vita qui ad Antibes. Grazie per tutti i momenti di divertimento vissuti insieme.

Summary

Coded caching is a communication technique that exploits cached content at the receivers to achieve much larger throughput gains with respect to conventional uncoded caching strategies. In the simplest scenario, there exists a library with N files of unitary length stored in a single-antenna base station and K users, each with a cache of size M units of file. The base station and users are connected via a shared broadcast link of normalized capacity 1 file per unit of time. The system operates in two different phases: placement phase and delivery phase. During the off-peak hours, when the network is under-loaded, the placement phase takes place and users store in their caches content from the library. During the peak hours, say the day, each user requests a different file (worst-case scenario) from the library. The base station sends a multicast message of normalized length R units of time, which satisfies all users' requests. Coded caching allows a single coded multicast transmission to be useful to many users at the same time. The delay R achieved by the coded caching technique proposed in [5] is information theoretical optimal.

It is also of particular interest the Device-to-Device (D2D) setting, where users are connected each other via a shared link and thus can directly communicate each other. In such a setting, during the delivery phase, users are not served by the base station storing the library but they exchange messages until all requests are satisfied. In the D2D setting studied in this work, we assume that each user is equipped with L antennas. We extend the state-of-art scheme (for the single-antenna case) to this multi-antennas case which experiences a multiplicative gain of L. However, for the coded caching gains to appear, each file has to be split into a huge number of subfiles, which increases exponentially with the number of users K. For practical file sizes, there exists a maximum number of subfiles each file can be split into. Such a constraint leads to an effective achievable delay much higher than the theoretical one. A users cooperation based scheme is proposed to tackle this problem. Such a scheme turns out to outperform the state-of-art algorithm in the subpacketization constrained regime.

During the recent years, the pioneers of coded caching have found a way to use the coded caching techniques to reduce the inter-servers communication in distributed computing. In particular, they considered a task that requires to process a big dataset to finally get Q output results. It is assumed that this task can be modelled according to the MapReduce model. The latter is a framework that allows to execute a task in a distributed manner among K servers in order to reduce the overall execution time. In this framework, the execution of the task happens in three different phases:

1. the *assignment and mapping phase*, where the dataset is distributed among the servers and a pre-processing operation is performed by each of them to the received portion of the dataset in order to obtain some intermediate results,

2. the *shuffling phase*, where the servers exchange intermediate results in a D2D setting, and

3. the *reduce phase*, where each server processes a subset of all the intermediate results to get $\frac{Q}{K}$ output results.

A major performance bottleneck of this distributed computing model is the time the system spend in the shuffling phase. It turns out that, by adapting the D2D cache placement and delivery scheme to distributed computing, it is possible to substantially reduce the shuffling time. This technique is known as Coded MapReduce. In this work, we identify the subpacketization constraint as a major problem that limits the theoretical gains promised by Coded MapReduce. We adapted the cooperation based scheme proposed for D2D coded caching to the context of distributed computing. The achieved gain of the cooperation based scheme is the same as the one achieved in D2D coded caching. Despite the scheme has been thought for a wireless setting, it can be potentially applied to a wired setting as soon as the computing nodes are connected via a wired network of nodes, e.g. switches, that can perform network coding operations.

In the context of coded caching, another interesting setting is the one of a network where the K users do not have memory for caching content but each of them can connect to one of $\Lambda < K$ helper nodes spread over the network, each with a cache of size M. During the placement phase, the helper nodes store content from the library. In the delivery phase, each user requests a different file from the library and fetches with no cost a portion of the desired file from the helper node it is connected to. The base station storing the whole library, sends a message of delay R to satisfy all users' requests. This setting has been studied in the literature for the single-antenna base station case and assuming that the same number of users is connected to each helper node, i.e. assuming there are $\frac{K}{\Lambda}$ users connected to each helper node. In this work, we consider the setting where the base station storing the entire library is equipped with N_0 antennas and where there can be an unequal number of users connected to each helper node. Let me define *user profile* the vector whose each element represents the number of users connected to a given helper node. To the best of our knowledge, this problem has never been addressed before. For this setting, we developed two delivery strategies for the single-antenna and multiple-antennas base station case, respectively. While the former can be employed for any user profile, the latter can be used for a wide-but-limited feasible set of user profiles (dependent on N_0). The major contribution for this setting is the development of an information theoretical lower bound on the delay R required by the base station to serve all users which matches the one achieved by the schemes. Hence, the proposed schemes, and respective achieved delays, turn out to be optimal.

Contents

List of Figures

Notation

- $\mathbb{N}$ denotes the set of natural numbers
- $[A]$ denotes the set of numbers from 1 to A as $[A] \triangleq \{1,2,\cdots,A\} : A \in \mathbb{N}$
- π denotes a permutation of A elements $\pi : \{1,...,A\} \to \{1,...,A\}$
- $|\mathcal{B}|$ denotes the cardinality of the set $\mathcal{B}$
- $\mathbf{v}^T$ denotes the transpose of the vector $\mathbf{v}$
- $\oplus$ denotes a XOR operation
- $\binom{n}{k}$ is the binomial coefficient, i.e. $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
- $||\cdot||^2$ denotes the norm 2 operator
- $log(\cdot)$ denotes the natural logarithm
- $\mathbf{P}_\pi$ denotes the permutation matrix defined as $\mathbf{P}_\pi \triangleq [\mathbf{e}_{\pi(1)} \;\; ... \;\; \mathbf{e}_{\pi(A)}]^T$ where $\mathbf{e}_{\pi(i)}$, a standard basis vector, denotes a row vector of length A with 1 in the $\pi(i)$-th position and 0 in every other position.
- $2^{[A]}$ is the power set of the set $[A]$
- $P(n,k)$ is defined as $P(n,k) = \frac{n!}{(n-k)!}$
- $Conv(f(i))$ denotes the lower convex envelope of the points $\{(i, f(i))|i \in [A] \cup \{0\}\}$ where $f(i)$ is a function defined in the set $[A] \cup \{0\}$.
- $\mathbb{1}_B(x)$ denotes the indicator function on a set X defined as $\mathbb{1}_B(x) \triangleq \begin{cases} 1, & \text{if } x \in B \\ 0, & \text{if } x \notin B \end{cases}$ where $x \in X$ and $B \subseteq X$

Chapter 1

Fundamental Limits of Caching

1.1 Caching Overview

1.1.1 Introduction

During the course of recent years, mobile data traffic has been growing exponentially due to smartphones, tablets and high bandwidth-consuming applications. In the near future, the growth of the data traffic will become even more alarming for the network operators because of the explosion of the Internet of Things and new data-intensive applications relying on machine learning and big data. According to Cisco forecast (Figure 1.1), the overall mobile data traffic is expected to grow by 47% per year up to 49 exabytes (EB) per month in 2021 (11 EB per month in 2017). Despite many new promising technologies, such as multi-cell cooperation, network densification, massive Multiple-Input and Multiple-Output (MIMO) systems and millimiter wave communication, have been envisioned and are currently studied to be part of the 5-th (5G) generation mobile network, they do not successfully scale with the number of users.

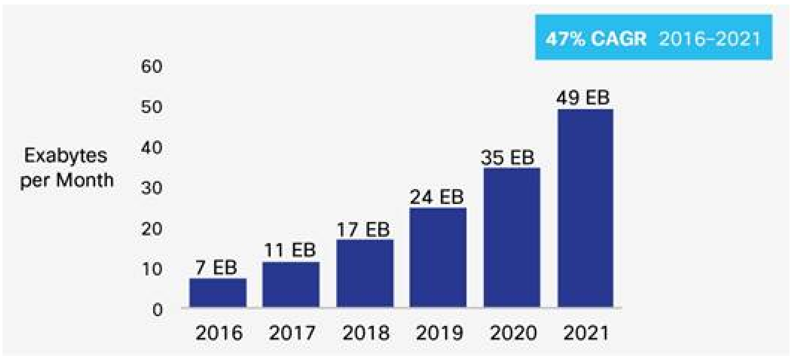

Figure 1.1. Global mobile data traffic forecast, adopted from "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper" [1].

In the last few years, there has been a clear enormous growth of mobile video traffic which, requiring much higher bit rates than other mobile content types, is predicted to account for more than 75% of the whole global traffic. In 2021, 38 exabytes of traffic out of the total 49 exabytes are expected to be due to video content (see Figure 1.2). In other words, video-on-demand traffic is driving global traffic growth. A peculiar characteristic of video traffic is the high temporal variability which makes the network overprovisioned during off-peak hours. Caching is a technique that can help to smooth the traffic and it is going to play a key role in the future wireless networks.

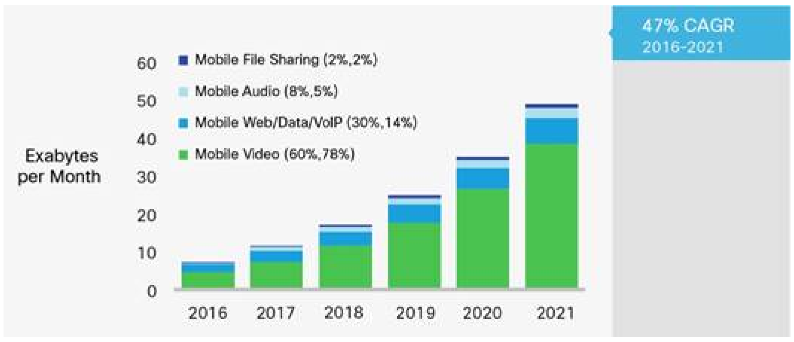

Figure 1.2. Global mobile data traffic forecast per traffic type, adopted from [1].

The high predictability of this type of traffic and the aforementioned high temporal variability rise the possibility to bring the most popular content closer to the end users during the off-peak times in order to reduce the network utilization during the peak hours. This is what, for instance, Netflix has been doing since 2011 with their own content delivery network. ISPs can operate a Netflix-operated cache in their network. As a result, nowadays almost all Netflix content is served from the local ISP's caches rather than upstream from the internet. In doing so, Netflix is making ISP's life easier since they are experiencing a core network's utilization almost undisturbed by Netflix traffic. Another important example of caching system is Facebook. The most popular social network uses a delivery network with a three level cache system: origin cache, edge cache and browser cache. The origin cache reduces backend server load, the edge cache reduces the traffic to the origin cache and browser cache reduces delay. There are many more other caching systems already deployed and well working such as Akamay, DNS, computer memory hierarchy and so on.

Broadly speaking, caching is used for three main purposes: network load reduction, load balancing and delay reduction. Motivated by the future need of delivering data, particularly video content, with high data rate (for high quality video) and low latency (for real-time video), the concept of wireless edge-caching have been studied a lot as a key-enabling technique for the 5G network. Edge-caching simply consists in bringing the content closer to end users via distributed storages throughout the network, included the edge, e.g. by placing caches co-located with the base stations. This caching approach, as well as the ones used in the aforementioned examples of caching systems, all it does is basically turning memory into bandwidth in a linear way and thus providing what we refers to as *local caching gain*.

1.1.2 Single Cache Systems

The aforementioned caching systems are commonly referred as single cache systems. A single cache system is composed of a main server containing N files and a cache that can store up to only M files. User requests a sequence of files, one at a time. If each requested file is in the cache we talk about *cache hit* and the user fetches the file from the cache, otherwise we talk about *cache miss* and the file is fetched from the server followed by a cache update for a future use. Clearly the cache is usually closer to the user than the main server, thus a cache hit would reduce the delay to acquire the file and the network load in the core network. Hence, the objective is to find an optimal algorithm to update the cache so that the number of cache misses is minimized. Let's now list briefly which are the best algorithms known in literature based on different assumptions:

- **Non-causally known requests**: Assuming the sequence of requested files is known a priori the best known algorithm is the *Belady's Algorithm* [3] which works as follows: whenever there is a miss, fetch the file from the server and update the cache by evicting the file requested farthest into the future and replacing the latter with the new requested file.
- **I.I.D requests with known distribution**: Assuming the sequence of requested files is independent and identically distributed (i.i.d.) and the file popularity distribution is known, e.g. Zipf distribution, then the *Highest Popularity First algorithm* is optimal and consists in caching the M most popular files. This algorithm does not need any cache update, unless the popularity distribution changes.
- **I.I.D requests with unknown distribution**: Let's now assume that the sequence of requests is i.i.d but the popularity distribution is not known. Then, the *Least-Frequently Used algorithm* [4] is the best one to be used and asymptotically optimal. This algorithm is based on the estimation of the popularity distribution from the entire request history:
  - keep track of the number of times each file has been requested;
  - whenever a cache miss occurs, evict the least-frequently used file and replace it with the new requested one.
- **Correlated sequence of requests and time varying popularity distribution**: In a real world system, requests are correlated in time and the file popularity is slowly time varying. Hence, no probabilistic model is available. To the best of the current knowledge, the best algorithm working under the above assumptions is the *Least-Recently Used* which update caches based only on the recent history (and not the entire history as LFU). LRU works as follow: each time a cache miss occurs, evict the least-recently-used file and replace it with the new requested one.

Despite the cache miss is commonly considered the best metric to be used in order to optimize a single cache system, there are other metrics that could be considered. For instance, assume we are interested in minimizing the worst-case delivery time, to satisfy the user request. For this analysis, we have to consider the worst-case sequence of requested files. Such a sequence is the one containing all the files of the server. It is intuitive that, the worst-case delivery time is minimized if we store in the cache a fraction $\frac{M}{N}$ of each file of the library. The worst-case delivery time is the metric used in multi-cache systems or better called *cache networks*.

1.1.3 Cache Networks

In its most general meaning, a cache network is a system comprised of a server storing a library with N files and a given number of caches that can store a certain fraction of the library. The users can get their requested files from a cache they can connect to or directly from the main server. In this system it is commonly assumed that we can fill the caches only once, i.e. there is no cache updating. Hence, the system operates in two different phases:

- **Placement phase**: This phase consists in populating the caches with the content in the library during the off-peak hours. In this phase, the demands are not known yet by the server;
- **Delivery phase**: In this phase, each user requests a file from the library. Users will get part of the requested content from the cache they can connect to and they will download the rest from the main server.

In the last years, there has been lots of research on cache networks whose information theoretical studies have revealed some fundamental insights that have debunked the conventional beliefs about caching.

Conventional Beliefs:

- Caches are useful to deliver content locally to the user.
- If the users are statistically independent, then the content stored in the several caches should be identical.
- Miss rate (percentage of time a cache miss occurs) is the most important performance metric.

Insights from Information Theory:

- Cache networks are useful to deliver content globally: the main gain is global
- If the users are statistically. independent, than the content stored in the several caches should be different.
- The most appropriate performance metric is not the miss rate but the worst-case delivery time.

These statements will become clear later on in this document.

1.2 Coded Caching in Noiseless Broadcast Channel

Consider a system with one server connected through a shared, error-free link to $K \in \mathbb{N}$ users. The server has access to a library of $N \in \mathbb{N}$, with $N \geq K$, files $W^1, ..., W^N$ each of normalized unitary size. The link capacity is normalized such that it is 1 file per unit of time. Each user $k \in \mathbb{N}$ has an isolated cache memory of size $M \in [N]$. We denote by γ the cache size normalized by the library size $\frac{M}{N}$, thus $\gamma \triangleq \frac{M}{N}$.

During the placement phase the server broadcasts the entire database of files letting the users fill their caches with a fraction γ of the library. We denote by Z_k the content stored by user k in its isolated cache. In the delivery phase, each user k requests a file W^{d_k}, where $d_k \in [N]$, from the database. The server is informed of the requests and transmits a signal of length R units of time over the shared link, which signal, along with the users' caches, will allow the users to retrieve entirely their desired file[1]. A memory-rate pair (M, R) is said to be achievable if every user k is able to recover its desired file for every possible demand vector $\mathbf{d} = (d_1, ..., d_K)$. The term $R^*(M)$ will be used here to refer to the smallest rate R such that (M, R) is achievable. The aim is to design a caching and delivery strategy that achieves $R^*(M)$.

Uncoded Delivery: The simplest strategy is for each user to cache the same γ fraction of each file during the placement phase. In the delivery phase, once the server becomes aware of the requested files from the users, it transmits sequentially and unicastly the $1 - \gamma$ remaining part of each file that the requesting user does not have in its cache. If the files requested from the K users are all different, then the delivery rate is given by

$$R_U(M) = K(1 - \gamma) \tag{1.1}$$

The factor $1 - \gamma$ is commonly referred to as *local caching gain* since it arises from having a fraction γ of the requested file available locally in the cache.
If a subset $\mathcal{K}^e \subseteq [K]$ of the K users requests the same file, then the multicast property of the shared link can be exploited. In fact, it is enough to transmit only once the fraction $1-\gamma$ of the file desired by the $\mathcal{K}^e$ users. Hence, whenever some users request the same file, some multicasting opportunities arise which help to reduce the rate $R(M)$. Multicasting opportunities can actually be forced to arise even when the users' demands are different as long as the caches are filled with different content in a proper structured manner. This idea came up for the first time in [5] where the authors study the fundamental limits of the caching networks described in this section.

[1]Throughout this document we will refer to R as delay and/or rate interchangeably, as it is common to do in the coded caching community.

Maddah-Ali and Niesen proposed in [5] a caching placement and delivery strategy that, under the condition of uncoded cache placement, have been proven recently in [6] to be optimal. The optimality has been proven by converting the caching problem into an index coding problem and leveraging the index coding bounds presented in [9].

Before describing the general scheme, an example can help the reader to understand the main idea.

Example Consider a system with $K = 4$ users and $N = 4$ files. Each user has a cache of size $M = 2$. In the placement phase, the files $W^n, \forall n \in [4]$ are split in 6 subfiles as follow: $W^n = \{W^n_{12}, W^n_{13}, W^n_{14}, W^n_{23}, W^n_{24}, W^n_{34}\}$. Then, the server transmits all the subfiles of each file and the users fill their caches with the following subfiles:

$$Z_1 = \{W^n_{12}, W^n_{13}, W^n_{14} \ \forall n \in [4]\}$$

$$Z_2 = \{W^n_{12}, W^n_{23}, W^n_{24} \ \forall n \in [4]\}$$

$$Z_3 = \{W^n_{13}, W^n_{23}, W^n_{34} \ \forall n \in [4]\}$$

$$Z_4 = \{W^n_{14}, W^n_{24}, W^n_{34} \ \forall n \in [4]\}$$

Each of the above messages/signals is a bitwise XOR of three subfiles.

Let's now assume that each user requests a different file so that users 1,2,3,4 request files W^1, W^2, W^3, W^4, respectively. Then, the best delivery strategy will create and sequentially transmit the following messages:

$$\mathbf{X}_{123} = W^1_{23} \oplus W^2_{13} \oplus W^3_{12}$$

$$\mathbf{X}_{124} = W^1_{24} \oplus W^2_{14} \oplus W^4_{12}$$

$$\mathbf{X}_{134} = W^1_{34} \oplus W^3_{14} \oplus W^4_{13}$$

$$\mathbf{X}_{234} = W^2_{34} \oplus W^3_{24} \oplus W^4_{23}$$

Let us now focus on user 1 receiving the transmitted signal $\mathbf{X}_{123}$, whose index 123 indicates that the transmission is intended for users 1,2,3. By using the content stored in its cache it can get its desired subfile W^1_{23}. In fact, user 1 knows W^2_{13} and W^3_{12} which can remove from $\mathbf{X}_{123}$ to get W^1_{23} free of interference of the subfiles intended for users 2 and 3. From the received messages $\mathbf{X}_{124}, \mathbf{X}_{134}$ user 1 can successfully decode W^1_{24} and W^1_{34}. Similarly the other users can successfully decode their desired subfiles. It is left to the reader to check that after the above 4 transmissions, all users, with the help of their caches, will have obtained their requested files.

The total time needed to deliver successfully all the subfiles to the 4 users is $R(2) = number\ of\ transmissions \times size\ of\ one\ subfile = 4 \times \frac{1}{6} = \frac{2}{3}$. If we transmitted the subfiles with the uncoded strategy it would take $R_U(2) = 4(1 - \frac{1}{2}) = 2$. Hence, coded caching allows to gain an enormous multiplicative factor of $\frac{R_U(2)}{R(2)} = 3$.

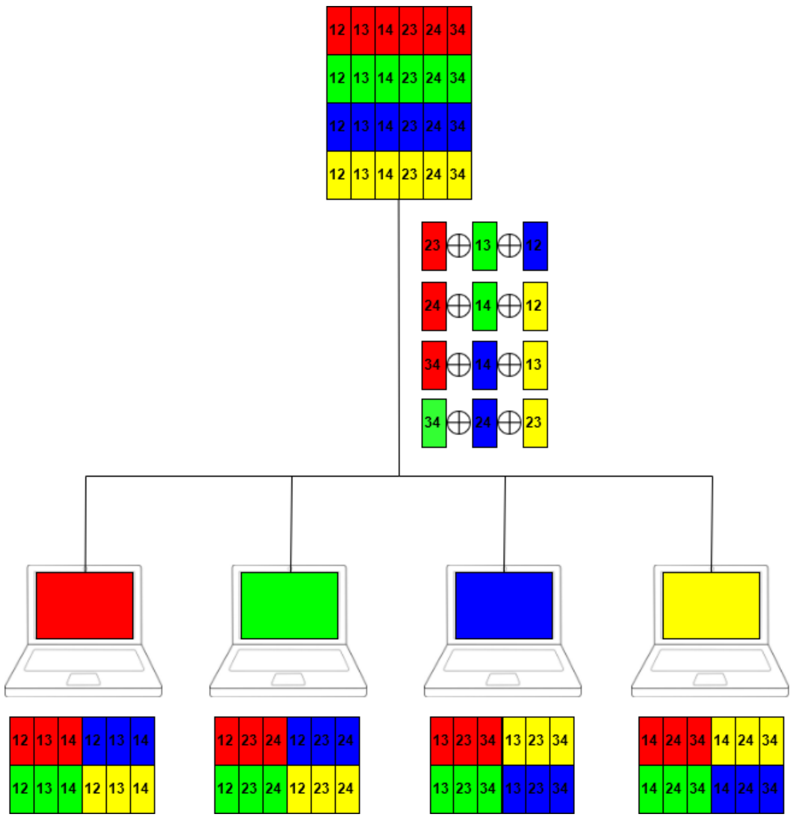

Figure 1.3. Pictorial representation of a coded caching system with $K = 4$ users, $N = 4$ files and caches of size $M = 2$. Each requested file is represented by a different color. The content stored by the users during the placement phase is depicted below the laptops. The chunks in which each file is divided are named with a set of $K\gamma = 2$ numbers.

A pictorial representation of the discussed example can be found in Figure 1.3.

In the following two paragraphs there are described the optimal general placement and delivery strategies for the considered caching problem.

Cache Placement Algorithm Each file W^n of the library is split into $S = \binom{K}{K\gamma}$ disjoint equally-sized subfiles as follows:

$$W^n = (W^n_\tau : \tau \subseteq [K] : |\tau| = K\gamma) \tag{1.2}$$

For each $n \in [N]$, subfile W^n_τ is placed in the cache of user k if $k \in \tau$. Hence, each user stores in its cache a total of $N\binom{K-1}{K\gamma-1}$ subfiles each of size $\frac{1}{\binom{K}{K\gamma}}$ which fill the whole user's local cache since

$$N\binom{K-1}{K\gamma-1}\frac{1}{\binom{K}{K\gamma}} = N\gamma = M \tag{1.3}$$

The pseudo-code of the placement strategy can be found in algorithm 1.

```
1 procedure PLACEMENT($W^1, ..., W^N$)
2 for all $n \in [N]$ do
3     Partition $W^n$ into $\binom{K}{K\gamma}$ disjoint subfiles :
4     $W^n = (W^n_\tau : \tau \subseteq [K] : |\tau| = K\gamma)$
5 end
6 $Z_k = (W^n_\tau : k \in \tau, \forall n \in [N]), \forall k \in [K]$
7 end procedure
```

Algorithm 1: MAN Placement Strategy

Delivery Algorithm Thanks to the placement strategy described in Algorithm 1, the Maddah-Ali and Niesen (MAN) delivery algorithm can satisfy all the user requests $(d_1, ..., d_K)$ with a total delay given by:

$$R(M) \leq \frac{K(1-\gamma)}{K\gamma + 1} \tag{1.4}$$

where the equality holds for the worst-case, i.e. when all users request a different file. In the following it is described the delivery strategy for the worst-case.

Consider a subset $Q \in [K]$ of $K\gamma + 1$ users and all the subsets $U \subset Q$ such that $|U| = K\gamma$. All the users in any subset U share one subfile stored in their caches which is needed by the remaining user in Q. In the delivery phase, for each of the $\binom{K}{K\gamma+1}$ sets Q, the server creates a message denoted by $\mathbf{X}_Q$ and transmits it to the $K\gamma + 1$ users in the set Q. $\mathbf{X}_Q$ is a XOR of $K\gamma + 1$ subfiles. Each subfile in the XOR is desired by one of the $K\gamma + 1$ users the message is intended for and it is available in the local caches of the other $K\gamma$ users. To be precise, the signal $\mathbf{X}_Q$ is constructed as follows:

$$\mathbf{X}_Q = \bigoplus_{k \in Q} W^{d_k}_{Q \setminus \{k\}}. \tag{1.5}$$

Hence, the total number of transmissions is $\binom{K}{K\gamma+1}$ and the total delay can be written as

$$R(M) = \frac{\binom{K}{K\gamma+1}}{\binom{K}{K\gamma}} = \frac{K(1-\gamma)}{K\gamma + 1}. \tag{1.6}$$

The pseudo-code of the scheme can be found in algorithm 2.

```
1 procedure DELIVERY($W^1, ..., W^N,\ d_1, ..., d_K$)
2 for  $Q \subset [K]\ :\ |Q| = K\gamma + 1$ do
3     Transmits $\mathbf{X}_Q = \bigoplus_{k \in Q} W^{d_k}_{Q \setminus \{k\}}$
4 end
5 end procedure
```

Algorithm 2: MAN Delivery Strategy

Discussion Let us now analyse carefully the expression of the rate given in equation (1.6). The factor K is simply the delay we would have if users had no memory to dedicate to caching. It has been already mentioned that the factor $1-\gamma$ is usually referred as *local caching gain* and it arises from the availability of a fraction γ of the requested files at the users' caches. Such a gain indicates a reduction in the rate which is relevant only when the local cache size M is of the order of the number of files N. Clearly, in a real system the value of γ would be very low, hence the local caching gain would be of negligible interest to improve the system performance. The last term $\frac{1}{K\gamma+1}$ is referred to as *global caching gain* and it contributes as a multiplicative reduction of the rate. This gain arises from jointly optimizing both the placement and the delivery phases, ensuring that in the delivery phase several different demands can be satisfied with a single coded multicast transmission. The global caching gain depends on the normalized cumulative cache size $K\gamma$ and, in contrast to the local caching gain, is relevant whenever the cumulative cache size KM is in the order of the number of files N. Figure 1.4 shows the rate required in the delivery phase to serve $K = 30$ users requesting different files from a library of $N = 30$ files as a function of the local cache size. The picture shows the huge gap between the rate needed with uncoded and coded delivery. The enormous gains promised by coded caching arise from the fact that the placement and delivery strategy are optimized to exploit the multicast property of the channel between the server and the users. Such a property is more clear in a wireless environment where the medium is naturally broadcast. All the known state-of-art techniques used today to serve users in a wireless environment try to break this property to avoid inter-user interference. Coded caching is the first technique (apart from the trivial multicasting used when users request the same content) that truly exploits the broadcast property of the wireless medium.

1.3 Coded Caching in D2D Noiseless Networks

Maddah-Ali and Niesen have provided a real breakthrough in the research in caching networks and more generally in the research of new communication paradigms that could handle the enormous amount of traffic that will be generated by killer applications such as on-demand video streaming. Researchers begun to study coded multicasting in several settings of relevant interest. Infrastructureless communication has been considered for a while as a promising way of increasing the network capacity. Device-to-Device (D2D) communication can, in fact, easily, leverage on the spatial reuse to increase the network capacity. The authors in [7] considered the caching problem in a D2D network in order to see if any further gain can be obtained by combining spatial reuse and coded multicasting.

System Model A D2D network can be modeled as a grid network formed by K users placed on a regular grid on the unit square, with minimum distance $\frac{1}{\sqrt{K}}$. Users can communicate each other via D2D communication without any infrastructure. If a user i transmits a packet to user j, then the transmission is successful if: 1) the distance between user i and j is smaller than the transmission range r (function of the transmitting power); 2) any other user k transmitting simultaneously, is at a distance $d(k, j) > r$ from

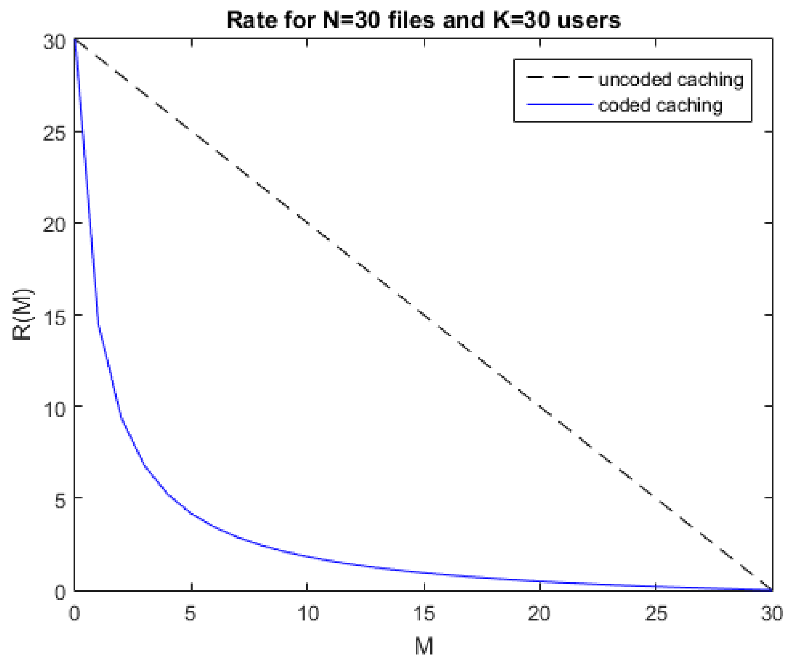

Figure 1.4. Plot of the rate-memory trade-off for a caching system with N=30 files and K=30 users.

the receiver j. In the following, *interference limited area* will be used to refer to a portion of the network where only one user at a time, among the ones in that area, can transmit free of interference. First, let us choose the transmission range $r \geq \sqrt{2}$ so that only one user at a time can transmit in the whole network.

Problem Formulation As in the broadcast channel, we consider a server having access to a library with N files. All the users in the network have an isolated cache of memory size M. In the placement phase the users will store a fraction $\gamma = \frac{M}{N}$ of the library in their caches, while in the delivery phase they will "collaborate" to obtain the requested files. Recalling that there is no infrastructure, user k will receive the missing part of the requested file, not available in its cache, from the other users in the network. An example of how coded caching works in a D2D network is described below.

Example Consider a network with $K = 4$ users each with a cache of size M. The library contains $N = 4$ files. In the placement phase, the files $W^n, \forall n \in [4]$ are split in 12 subfiles $W^n_{\tau,p}$ as follows:

$$W^n = \{W^n_{12,p}, W^n_{13,p}, W^n_{14,p}, W^n_{23,p}, W^n_{24,p}, W^n_{34,p} \ \forall p \in \tau\} \tag{1.7}$$

Then, the server transmits all the subfiles of each file and the users fill their caches $Z_k,\ k \in [4]$ with the following subfiles:

$$Z_1 = \{W^n_{12,p}, W^n_{13,p}, W^n_{14,p} \ \forall p \in \tau \ \forall n \in [4]\}$$

$$Z_2 = \{W^n_{12,p}, W^n_{23,p}, W^n_{24,p}\ \forall p \in \tau\ \forall n \in [4]\}$$
$$Z_3 = \{W^n_{13,p}, W^n_{23,p}, W^n_{34,p}\ \forall p \in \tau\ \forall n \in [4]\}$$
$$Z_4 = \{W^n_{14,p}, W^n_{24,p}, W^n_{34,p}\ \forall p \in \tau\ \forall n \in [4]\}$$

Let us now assume that each user requests a different file, e.g. user 1 requests file W^1, user 2 requests file W^2, user 3 requests file W^3 and user 4 requests file W^4. Then, the best delivery strategy will operate as follows.

Consider all subsets of $K\gamma + 1 = 3$ users, e.g. $Q = \{1,2,3\}$. For the considered set Q, let each user transmits consecutively the following messages:

$$\mathbf{X}_{1,23} = W^2_{13,1} \oplus W^3_{12,1}$$
$$\mathbf{X}_{2,13} = W^1_{23,2} \oplus W^3_{12,2}$$
$$\mathbf{X}_{3,12} = W^1_{23,3} \oplus W^2_{13,3}$$

where $\mathbf{X}_{i,Q\setminus\{i\}}$ denotes the transmission from user i intended for users in the set $Q \setminus \{i\}$. For the other 3 subsets $\{1,2,4\}, \{1,3,4\}, \{2,3,4\}$ of size $K\gamma + 1 = 3$ the following transmissions will happen:

$Q = \{1,2,4\}$	$Q = \{1,3,4\}$	$Q = \{2,3,4\}$
$\mathbf{X}_{1,24} = W^2_{14,1} \oplus W^4_{12,1}$	$\mathbf{X}_{1,34} = W^3_{14,1} \oplus W^4_{13,1}$	$\mathbf{X}_{2,34} = W^3_{24,2} \oplus W^4_{23,2}$
$\mathbf{X}_{2,14} = W^1_{24,2} \oplus W^4_{12,2}$	$\mathbf{X}_{3,14} = W^1_{34,3} \oplus W^4_{13,3}$	$\mathbf{X}_{3,24} = W^2_{34,3} \oplus W^4_{23,3}$
$\mathbf{X}_{4,12} = W^1_{24,4} \oplus W^2_{14,4}$	$\mathbf{X}_{4,13} = W^1_{34,4} \oplus W^3_{14,4}$	$\mathbf{X}_{4,23} = W^2_{34,4} \oplus W^3_{24,4}$

Let us focus on user 1 receiving the signal $\mathbf{X}_{2,13}$ transmitted by user 2 and intended for users 1 and 3. By using the content stored in its cache, user 1 can get its desired subfile $W^1_{23,2}$. In fact, user 1 knows $W^3_{12,2}$ which can remove from $\mathbf{X}_{2,13}$ to get $W^1_{23,2}$ free of interference of the messages intended for user 3. Analogously, user 1 can do the same with the other received messages intended for him. With the same approach, the other users can decode their desired subfiles. It is left to the reader to check that after the above $3\binom{4}{3} = 12$ transmissions, all users, with the help of their caches, will have obtained their desired files.

The total time needed by the 4 users to exchange messages to finally obtain their desired files is $R(2) = \frac{12}{12} = 1$.

Placement and Delivery Strategy As we can see from the above example, the placement strategy is almost the same as in the broadcast channel. In fact, each file W^n is split into $S = K\gamma\binom{K}{K\gamma}$ subfiles $W^n_{\tau,p}, \forall \tau \subseteq [K] : |\tau| = K\gamma, \forall p \in \tau$ and each user k stores subfiles whose index τ is such that $k \in \tau$.
Also the delivery strategy is similar to the one proposed by Maddah-Ali and Niesen for the broadcast channel. The main difference is that now, given a subset of $K\gamma + 1$ users, we have to "sacrifice" one to be the transmitter because there is no infrastructure that can

deliver the content. Given a set Q of $K\gamma+1$ users, every subset $U \subset Q$ of $K\gamma$ users share $K\gamma$ subfiles that are desired by the remaining user in the set Q. Building on the above reasonings, The authors of [7] proposed the delivery strategy given in the pseudo-code of algorithm 3.

```
1 procedure DELIVERY($W^1, ..., W^N,\ d_1, ..., d_K$)
2 for $Q \subset [K]\ :\ |Q| = K\gamma + 1$ do
3     for all $i \in Q$ do
4         Transmit $\mathbf{X}_{i,Q\setminus\{i\}} = \bigoplus_{k\in Q\setminus\{i\}} W^{d_k}_{Q\setminus\{k\},i}$
5     end
6 end
7 end procedure
```

Algorithm 3: Delivery Strategy for D2D coded Caching

The described strategy requires $(K\gamma+1)\binom{K}{K\gamma+1}$ multicast transmissions to serve successfully all K users. Therefore, the worst-case (when all users request a different file) transmission rate achieved by the algorithm is

$$R(M) = \frac{(K\gamma+1)\binom{K}{K\gamma+1}}{(K\gamma)\binom{K}{K\gamma}} = \frac{K(1-\gamma)}{K\gamma} \tag{1.8}$$

which has been proven to be order-optimal in [7][2].

Comparing the rate achieved by coded caching in D2D networks and the one achieved in the broadcast channel, we notice how the lack of infrastructure lead to a mere loss of 1 in the multiplicative gain due to coded caching which becomes negligible for high values of $K\gamma$.

Coded Multicasting and Spatial Reuse Let us now focus on the case in which the transmission range is $r \leq \sqrt{2}$. In this case, the transmission range can be chosen in order to have localized D2D communication and therefore allow for some spatial reuse. The network is divided in clusters of equal size, each containing $\frac{K}{K'}$ users, where K' denotes the number of clusters. Users can communicate only with the other nodes within the same cluster. It is assumed here that the total cache capacity of each cluster is sufficient to store the whole library, i.e. $\frac{K}{K'} \cdot M \geq N$. Each cluster can be considered a separate network. The simplest transmission policy consists in partitioning the set of clusters into $\mathcal{K}$ reuse sets, such that the clusters within the same reuse set do not interfere each other and can be active simultaneously. Clearly, by using the above delivery strategy the

[2]An achievable rate $R(M)$ is said to be order-optimal if there exists a finite value $c \in \mathbb{R}$ such that $R(M) \leq c \cdot R^*(M)$ for any value of the system parameters M, N, K, where $R^*(M)$ is the optimal rate.

following transmission rate is achievable:

$$R(M) = \mathcal{K}\frac{\frac{K}{K'}(1-\gamma)}{\frac{K}{K'}\gamma} = \mathcal{K}\frac{K(1-\gamma)}{K\gamma} \tag{1.9}$$

If we allow for full spatial reuse ($\mathcal{K} = 1$), then the achieved rate coincides with the one given by pure coded caching regardless of K'. This suggests that there is no fundamental cumulative gain by using both spatial reuse and coded caching. However, a look at the two extremes reveals a more subtle trade-off. Without any spatial reuse, coded multicasting requires a subpacketization (number of subfiles in which each file is split into) of $S = K\gamma\binom{K}{K\gamma}$ which may be very high for high values of K and M. This is actually a very well-known problem in coded caching that is deeply investigated in this work. At the other extreme, we can allow for the maximum spatial reuse, i.e. letting the cluster size be the minimum possible to store the whole library. In this case, we can store M whole different files into each node and in the delivery phase we can transmit whole files without any coding as in [8]. In this case, the achieved rate is $\frac{1}{\gamma}$ which is almost as good as the coded scheme.

1.4 Practical Coded Caching: the Main Challenges to Make it Feasible

In the previous sections, we have shown the potentials of coded caching both for the common broadcast channel and for a D2D network. However, there are many challenges that have to be addressed to make it practical. In the following, we briefly describe the main directions that the research in this area is following with a major focus on the so-called subpacketization constraint which will be highly investigated in this work.

Online Caching

The caching problem described in the previous sections has two distinct phases: cache placement phase and delivery phase. The cache is updated only in the placement phase and not anymore during the delivery phase. Nowadays, many caching systems use online cache updates. Such algorithms update the caches during the delivery phase based on some "local" variation of the file popularity. The most common and effective algorithm is the LRU which has been mentioned already in the introduction. In [10] the authors proposes an algorithm that updates the caches online.

1.4.1 Nonuniform File Popularities

In [5] and [7] it is assumed that all files in the library have the same popularity. In real systems, files have different popularities. In this case, the cache placement and delivery strategy described above may not be optimal. It is of strong interest to find optimal, or sub-optimal, schemes for any popularity distribution. Some work towards this direction has been done in [11] and [12].

1.4.2 More General Networks

The two networks considered above, broadcast channel and D2D, are the simplest and most studied. However, many other network topologies could be considered. It is of particular interest to consider the network where the caches are separated from the user terminals and spread throughout the network. Such a network will be addressed in the last chapter of this thesis. A tree topology with caches at several levels is also another interesting direction. Moreover, till now the studied strategies consider equally sized caches. Studying the impact of different caches' size is another interesting topic.

1.4.3 Decentralized Caching

The massive gains promised by coded caching rely on a meticulous content placement in the user's caches. Crucially for the described schemes, both the number and the identity of the users which will be present in the delivery phase have to be known a priori for the placement. Recalling that the placement phase happens during the off-peak hours, say during the night, and the delivery phase during the peak-hours, say during the day, then assuming to know the users that will show up and request files from the library the day after is not very realistic. Moreover, the delivery scheme assumes the requests to be synchronized. In practice, users join and leave the system over a period of several hours, resulting in a time-varying number of users. In order to handle these issues, the placement place needs to be decentralized, i.e. not orchestrated by a central server. Maddah Ali and Niesen have developed in [13] a random caching placement that can deal with all the mentioned issues.

1.4.4 The Subpacketization Problem

The massive theoretical gains promised by coded caching remain – under some realistic assumptions – hard-bounded by small values in the finite file-size regime. Before analysing in details such a problem let us introduce another measure of performance which will be useful to investigate the problem. Equivalently to the rate $R(M)$, in the literature it is highly used the sum Degree of Freedom (DoF) of the system which is defined as:

$$DoF(\gamma) \triangleq \frac{K(1-\gamma)}{R(M)} \tag{1.10}$$

which captures the effect of caching and represents the number of users that are served in one unit of time. The degree of freedom represents the number of users that can be served at the same time by using the same time-frequency resource. We can now define the gain given by caching in terms of achieved DoF as follows:

$$G \triangleq DoF(\gamma) - DoF(0) \tag{1.11}$$

where $DoF(0)$ represents the DoF of the system when users cannot cache any content. The gain G represents the number of extra users that can be served at a time as a consequence of introducing caching.

In the following we will present the subpacketization problem for the broadcast channel studied in [18].

The MAN algorithm for the broadcast channel achieves the sum DOF

$$DoF(\gamma) = K\gamma + 1 \tag{1.12}$$

which leads to a coded caching gain of

$$G = K\gamma \tag{1.13}$$

since in the broadcast channel $DoF(0) = 1$.

The algorithm proposed in [5] for the broadcast channel requires the splitting of files into $\binom{K}{K\gamma}$ subfiles which increases exponentially with K. This becomes clear by bounding the aforementioned binomial coefficient as follows:

$$\left(\frac{1}{\gamma}\right)^{K\gamma} \leq \binom{K}{K\gamma} \leq \left(\frac{e}{\gamma}\right)^{K\gamma}. \tag{1.14}$$

However, the finite file size limits the number of subfiles each file can be split into and consequently limits also the number of users over which it is possible to encode.

Nowadays, the atomic unit of storage in hard drives is of size 512 bytes and the disk drive industry is willing to move this soon to 4096 bytes. Consequently, the maximum number of chunks that a file can be split into is much smaller than itself size. It should be also recalled that each subfile will be encapsulated into some packet with an header size which becomes more and more dominant as the subfile size decreases.

Defining by S_{max} the maximum number of subfiles each file can be split into, the maximum number of users $\bar{K}$ that an instance of MAN algorithm can treat is given by

$$\bar{K} = \underset{K}{argmax}\{K : \binom{K}{K\gamma} \leq S_{max}\} \tag{1.15}$$

and gives an effective gain of

$$\frac{logS_{max}}{1 + log\frac{1}{\gamma}} \leq \bar{G} \leq \frac{logS_{max}}{log\frac{1}{\gamma}}. \tag{1.16}$$

Hence, assuming for the sake of analysis that $K = \alpha\bar{K}$, with $\alpha \in \mathbb{N}$, we would need to repeat the MAN scheme $\alpha = \frac{K}{\bar{K}}$. THe achieved rate would be $R(M) = \frac{K}{\bar{K}}\frac{\bar{K}(1-\gamma)}{\bar{K}\gamma+1}$ which leads to the effective achievable sum DoF of $\overline{DoF}(\gamma) = \bar{K}\gamma + 1 \ll K\gamma + 1$.

For example, consider a system with $K = 60$ users and $\gamma = 0.2$. MAN scheme would require splitting each file into $S = \binom{60}{12} \approx 10^{12}$ subfiles. Thus, the gain $G = K\gamma = 12$ (corresponding to a delay $R(M) = 3.69$) promised by MAN scheme can be actually achieved only if the file size is higher than 1 Terabit. Being realist, let us suppose that each file is 1 Gigabit sized and that we can allow for a minimum subfile size of 1 Kilobit which results in a subpacketization constraint of $S_{max} \approx 10^6$. Under such a constraint we could for example treat 30 users at a time leading to a gain of $\bar{G} = 6$ which is half of the theoretical promised gain.

A first attempt in reducing the subpacketization was done in the work in [14] where the coded caching problem was reformulated into a placement-delivery (PD) array combinatorial design problem to design an algorithm achieving a theoretical gain of $G = K\gamma - 1$ with a reduced subpacketization of $S = \left(\frac{1}{\gamma}\right)^{K\gamma-1}$. Under the subpacketization constraint S_{max} the PD algorithm achieves a gain $G_{PD} = \frac{log S_{max}}{log\frac{1}{\gamma}}$ which concides with the maximum gain that MAN scheme could achieve (see equation (1.16)). The authors of [14] show that the PD scheme can decrease, with respect to MAN scheme, the required file size of a factor that increases exponentially with K for a gain loss of only 1. Another important result came out in [16] where constructions that tradeoff performance and subpacketization are provided requiring though that $K > \frac{4}{\gamma^2}$ in order to have gains bigger than 1. The recent work in [15] employs Ruzsa-Szemérедi graphs to build an algorithm achieving (for very large unrealistic K) a gain that scales with K, with a subpacketization that scales with $K^{1+\delta}$ for some arbitraly small positive δ. For realistic values of S_{max} and γ, all the aforementioned algorithms improve MAN effective gain by a factor that remains hard bounded and small.

It is worth to recall that the subpacketization problem becomes even more severe when coded caching is applied in other network topologies such us D2D and multi-server, or equivalenty Multi-Input Single-Output (MISO) broadcast channel, setting (see [7] and [17]). Thus, a deep study of the subpacketization problem is necessary to make coded-caching practical and scalable with the system parameters.

A truly breakthrough has been done by Lampiris and Elia in [18] where they show how adding transmitters (antennas for a wireless setting) can drammatically boost the effective coded-caching gain under the subpacketization constraint S_{max}. The scheme developed in [18] allows in a system with L transmitters to achieve an effective coded-caching gain that is L times higher than any other known algorithm. Such a scheme leverages 2 main principles: 1) group users in groups of L users so that users within the same group cache the same content and 2) use zero-forcing precoding to null-out intra-group interference which cannot be nulled by caching. The same principle is behind the low subpacketization schemes that will be proposed in the next chapter for D2D networks.

Chapter 2

D2D Caching Networks: Boosting the Performance via Multiple Antennas and Users' Cooperation

2.1 Introduction

In this chapter we study the single-hop D2D network where cache-enabled nodes operate within the coded caching framework to satisfy their requests. It was shown in [7] that coded caching can achieve the information theoretical lower bound within a constant multiplicative factor. The gains arising from coded multicasting, in reality are highly constrained by the required extremely high number of subfile in which the files of the library have to be split. We recall that the rate achieved by the centralized algorithm in [7] is $R(M) = \frac{K(1-\gamma)}{K\gamma}$, where γ is the user's cache size M normalized by the library size N. The aforementioned rate results in a theoretical achievable sum degree of freedom of $DoF(\gamma) = K\gamma$. Analogously to the broadcast channel, the subpacketization constraint S_{max} limits the maximum number of nodes $\bar{K}$ over which it is possible to encode and it can be formulated as follows

$$\bar{K} = \underset{K}{argmax}\{K : K\gamma\binom{K}{K\gamma} \leq S_{max}\}. \tag{2.1}$$

Because of the above subpacketization constraint, the set of K users has to be divided in subsets of $\bar{K}$ users so that the coded caching placement and delivery strategy is consecutively applied to each of this smaller sets.[1] As a result, because of the constraint S_{max}, the achieved rate is $\bar{R}(M) = \frac{K}{\bar{K}}\frac{\bar{K}(1-\gamma)}{\bar{K}\gamma}$ corresponding to an effective maximum achievable sum degree of freedom of $\overline{DoF}(\gamma) = \bar{K}\gamma$. Inspired by by the work of Lampiris and Elia in [18] for the MISO broadcast channel, where multiple antennas at the base station (colocated with the library) and users grouping are leveraged to reduce the subpacketization

[1]For ease of analysis, we assume that $\bar{K}$ divides K.

and consequently get higher effective DoF, we wonder if the grouping approach and the addition of more antennas at the user terminals can help in increasing the theoretical and/or effective achievable DoF.

2.2 Problem Statement

Consider a D2D network with K users $1, ..., K$ equipped with L antennas each. Each user has a cache where it can store content corresponding to M equally-sized files of normalized size 1. Users request a file from a library containing N files $W^1, W^2, ..., W^N$. For practical reasons, we assume that each file cannot be split into more than S_{max} subfiles. We will refer to it as the subpacketization constraint.

Regarding the communication medium, we will focus on the wireless fully-connected setting where, in each time slot, only one user at a time can transmit through the wireless shared medium. At each point there will be a set of active receivers and active transmitters. Assuming a set of L_c active users, denoted by $\mathcal{G} \subset [K]$ jointly transmitting vector $\mathbf{X} \in \mathbf{C}^{L_c L \times 1}$, then the received signal at a receiving user k takes the form

$$\mathbf{y}_k = \mathbf{H}_{k,\mathcal{G}}^T \mathbf{X} + \mathbf{w}_k, \quad k \in [K] \tag{2.2}$$

where $\mathbf{X}$ satisfies a power constraint $\mathbb{E}(||\mathbf{X}||) < P$, where $\mathbf{H}_{k,\mathcal{G}} \in \mathbf{C}^{L_c L \times L}$ is the (potentially random) fading channel matrix between the transmitting set of users $\mathcal{G}$, each equipped with L antennas, and the receiving user k, i.e. $\mathbf{H}_{k,\mathcal{G}} \in \mathbf{C}^{L_c L \times L} =$ $[\mathbf{H}_{k,\mathcal{G}(1)} \cdots \mathbf{H}_{k,\mathcal{G}(L_c)}]$ with $\mathbf{H}_{k,\mathcal{G}(j)}$ being the $L \times L$ channel matrix between user k and user $\mathcal{G}(j)$ (which represents the j-th user of the set $\mathcal{G}$). Each element of $\mathbf{H}_{k,\mathcal{G}(j)}$ is a coefficient representing the propagation channel between one antenna of user k and one antenna of user $\mathcal{G}(j)$. Finally, $\mathbf{w}_k$ denotes the i.i.d vector of unit-power Additive-White-Gaussian-Noise (AWGN) noise at receiver k. We assume the system to operate in the high Signal-to-Noise-Ratio (SNR) regime (high P), and we assume perfect channel state information (CSI) at the active receivers and transmitters, i.e. they have a perfect knowledge of the state of the propagation channels. More precisely, we will always assume that a given user k has full knowledge of the channels $\mathbf{H}_{k,j}$ for $j \in [K]\backslash\{k\}$ but not of the channel matrices between other users, i.e. $\mathbf{H}_{j,i},\ i,j \in [K]\backslash\{k\}$[2]. We will refer to it as *local CSI*, while we will talk about *global CSI* when each user knows the channel matrices between other users.

The system operates under the coded caching framework considered in this thesis where during the off-peak hours the caching placement takes place and during the peak-hours each user requests a file from the library that will be served with a proper delivery algorithm from the other users. As usual, we denote by $R(M)$ the rate achieved by a given caching-delivery scheme for the worst-case scenario, i.e. when all users request a

[2]We assume that the system works in Frequency Division Duplex (FDD) mode so that the channel state between two antennas is the same regardless of the direction of the transmission.

different file. We will also denote by DoF the achievable degree of freedom and by $\overline{DoF}$ the achievable degree of freedom under the subpacketization constraint S_{max}.

The rest of this chapter is organized as follows. We first describe a generalized content placement strategy followed by the description of three different delivery strategies. In the section subsequent the third strategy we provide a set of examples to better help the reader to understand the delivery schemes. Then, a summary of results is presented, followed by a final discussion and conclusions.

2.3 A Generalized Content Placement Strategy

In this section, we present a generic content placement strategy preceding all our proposed delivery schemes that will be presented in the next sections. Before proceeding to the description of the scheme we define *cooperation factor*, denoted by $L_c \in \mathbb{N}, L_c \ll K$ and L_c dividing K, the number of users which we will let to cooperate in the delivery phase. The meaning and the way this parameter is used will become clear throught the description of both the cache placement and delivery phase. Once the designer chooses the value of the cooperative factor L_c, which we can anticipate will have a strong impact on the subpacketization, the cache placement is designed as if there were only $K' \triangleq \frac{K}{L_c}$ isolated caches.

We now proceed to the detailed description of the caching algorithm which is inspired by the Maddah-Ali et al. placement scheme in [5].

We first partition the set of K users $k = 1,2,...,K$ into K' disjoint groups

$$\mathcal{G}_i = \{lK' + i,\ l = 0,1,...,L_c - 1\},\ for\ i = 1,2,...,K' \tag{2.3}$$

of $|\mathcal{G}_i| = L_c$ users per group. In doing so, we aim to serve $K'\gamma$ groups at a time. To this end, each file W^n of the library is split into $S_{L_c} = LK'\gamma\binom{K'}{K'\gamma}$ disjoint equally-sized subfiles labelled with three indices as follows:

$$W^n = (W^n_{\tau,p,l} : \tau \subseteq [K'] : |\tau| = K'\gamma, p \in \tau, l \in [L])$$

.

For each $n \in [N]$, subfile $W^n_{\tau,p,l}$ is placed in the cache of user k if $(i \in \tau) \wedge (k \in \mathcal{G}_i)$. Hence, each user stores in its cache a total of $NLK'\gamma\binom{K'-1}{K'\gamma-1}$ subfiles each of size $\frac{1}{LK'\gamma\binom{K'}{K'\gamma}}$ which fill the whole user's local cache since

$$NLK'\gamma\binom{K'-1}{K'\gamma-1}\frac{1}{LK'\gamma\binom{K'}{K'\gamma}} = N\gamma = M$$

Remark: In case of single antenna users ($L = 1$) and no cooperation among users ($L_c = 1$), the placement scheme and the subpacketization are the same as the ones in [7].

Notice that, by using the described cache placement strategy, the higher is the value of the chosen L_c the lower is the subpacketization.

A pictorial illustration of the caching placement for a system with 6 single-antenna users requesting files from a library with 6 files, having caches of size 4 and using a cooperation factor of $L_c = 2$ is given in Figure 2.1. Each small colored square represents a different subfile, whose color correspond to a different files. Thus, for example the red square with label 23,2 is cached at users in groups $\mathcal{G}_2$ and $\mathcal{G}_3$. Each big square labelled by $Ui, i \in [6]$ represents a different user and its color corresponds to the file that the user requests. Thus for example, user $U1$ requests the file corresponding to the red color.

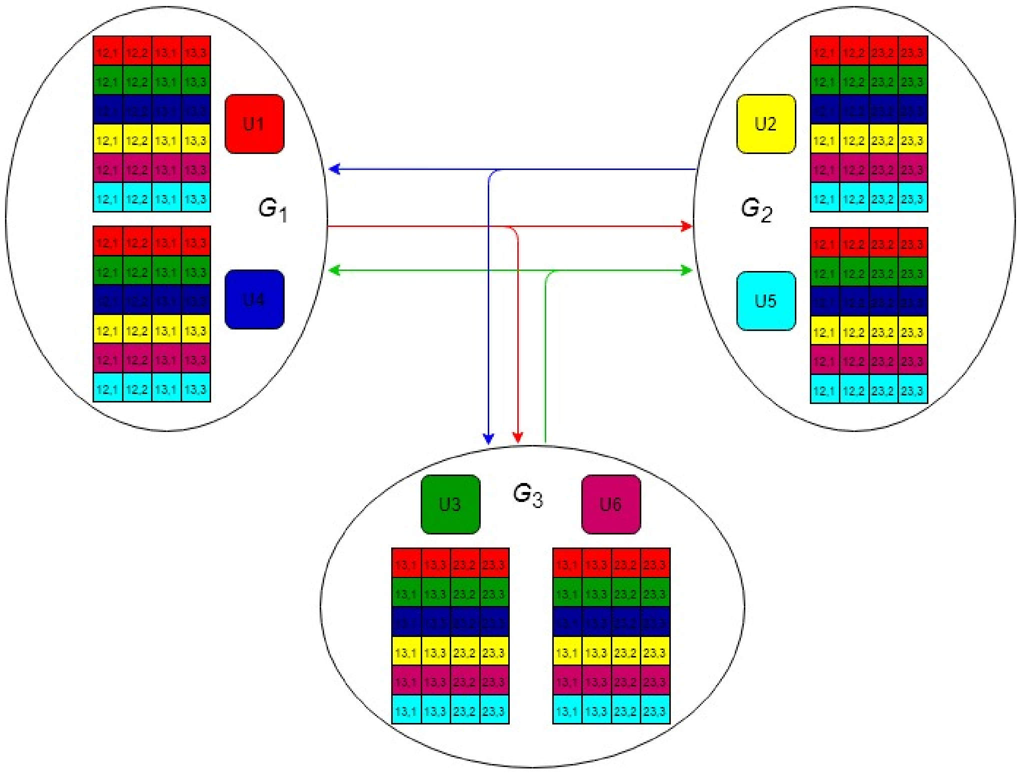

Figure 2.1. Cache placement for $K = 6$, $N = 6$, $M = 4$, $L = 1$ and $L_c = 2$

2.4 XOR Based Delivery Strategy

In this section, we present a delivery scheme achieving the rate

$$R(M) = \frac{K(1-\gamma)}{LK\gamma}. \tag{2.4}$$

The scheme we are going to describe is an extension of the one presented in [7] to the case of multiple-antennas users. The L per user antennas allow to reach an L-fold gain over the well known scheme for single-antenna users. The subpacketization required by the scheme is $S_1 = LK\gamma\binom{K}{K\gamma}$. Hence, during the off-peak hours, the placement algorithm is run with the cooperation factor set to 1, i.e. $L_c = 1$, which means that the groups, as defined in equation (2.3), are actually comprised of only 1 user. In other words, the set of users is not partitioned in groups.

2.4.1 Description of the Scheme

Consider a subset $Q \in [K]$ of $|Q| = K\gamma + 1$ users and all the subsets $U \subset Q$ such that $|U| = K\gamma$. Any subset U of users share $K\gamma$ bunches of L subfiles stored in their caches which are needed by the remaining user in Q. Moreover, there are $\binom{K\gamma}{K\gamma-1} = K\gamma$ subsets U containing a given user k. Hence, every user k has $K\gamma$ bunches of L subfiles, each of which bunches needed by one of the other $K\gamma$ users in Q and known by the remaining $K\gamma - 1$ users in Q.

In the delivery phase, for each of the $\binom{K}{K\gamma+1}$ sets Q, each user $i \in Q$ creates the $L \times 1$ vector

$$\mathbf{X}_{i,Q\backslash\{i\}} = \begin{bmatrix} \bigoplus\limits_{k\in Q\backslash\{i\}} W^{d_k}_{Q\backslash\{k\},i,1} \\ \vdots \\ \bigoplus\limits_{k\in Q\backslash\{i\}} W^{d_k}_{Q\backslash\{k\},i,L} \end{bmatrix} \tag{2.5}$$

and transmits it to the $K\gamma$ users in $Q\backslash\{i\}$. Each user $k \in Q\backslash\{i\}$ receives the signal $\mathbf{y}_k = \mathbf{H}_{k,i}\mathbf{X}_{i,Q\backslash\{i\}}$[3] which is decoded using the Zero-Forcing (ZF) receiver $\mathbf{H}_{k,i}^{-1}$ thanks to which the vector $\mathbf{X}_{i,Q\backslash\{i\}}$ is decoded with no errors. Next, user k constructs the vector $\mathbf{X}_{i,Q\backslash\{i,k\}}$ with the help of its cache and subtracts it from $\mathbf{X}_{i,Q\backslash\{i\}}$ to obtain

$$[W^{d_k}_{Q\backslash\{k\},i,1} \cdots W^{d_k}_{Q\backslash\{k\},i,L}]^T \tag{2.6}$$

free of inter-users interference.

In algorithm 4 a pseudo-code of the delivery algorithm is sketched to give a compact view of it.

[3]Here the noise is neglected because we are interested in characterizing the DoF region.

```
1 procedure DELIVERY($W^1, ..., W^N,\ d_1, ..., d_K$)
2 for $Q \subset [K]\ :\ |Q| = K\gamma + 1$ do
3     for all $i \in Q$ do
4         Transmit $\mathbf{X}_{i,Q\backslash\{i\}} = \begin{bmatrix} \bigoplus_{k\in Q\backslash\{i\}} W^{d_k}_{Q\backslash\{k\},i,1} \\ \vdots \\ \bigoplus_{k\in Q\backslash\{i\}} W^{d_k}_{Q\backslash\{k\},i,L} \end{bmatrix}$
5     end
6 end
7 end procedure
```

Algorithm 4: XOR based strategy

2.4.2 Performance Evaluation

The algorithm uses $(K\gamma+1)\binom{K}{K\gamma+1}$ transmissions ($K\gamma+1$ transmissions for each set Q) of XORs to serve all the users and each XOR is of size $\frac{1}{S_1}$, where we remind that S_1 denotes the subpacketization required by the scheme, i.e. $S_1 = LK\gamma\binom{K}{K\gamma}$. Hence, the rate achieved by the proposed scheme is

$$R(M) = \frac{(K\gamma+1)\binom{K}{K\gamma+1}}{LK\gamma\binom{K}{K\gamma+1}} = \frac{K(1-\gamma)}{LK\gamma}. \tag{2.7}$$

Under the subpacketization constraint S_{max}, we can encode over only a limited number of nodes $\bar{K}$ whose value is given below

$$\bar{K} = \underset{K}{argmax}\{K : LK\gamma\binom{K}{K\gamma} \leq S_{max}\}. \tag{2.8}$$

Consequently, we can treat only $\bar{K}$ users at a time leading to a lower rate of

$$\bar{R}(M) = \frac{K}{\bar{K}}\frac{\bar{K}(1-\gamma)}{L\bar{K}\gamma} \tag{2.9}$$

which correspond to an effective DoF of

$$\overline{DoF}(\gamma) = \frac{K(1-\gamma)}{\bar{R}(M)} = L\bar{K}\gamma \tag{2.10}$$

2.5 Cooperation Based Strategy

The scheme presented in the previous section achieves the theoretical DoF of $LK\gamma$ with a huge subpacketization, precisely with required subpacketization $S_1 = LK\gamma\binom{K}{K\gamma}$. In this section, we present a novel scheme that, thanks to cooperation among users, achieves the same theoretical performance of the aforementioned delivery strategy (Algorithm 4) with an unbounded low subpacketization. Users in the system are placed in groups of L_c users so that all the ones being part of the same group store the same content in their local caches. For the placement phase, the caching strategy presented in section 2.3 with parameter L_c is used. The grouping approach along with a global CSI knowledge will allow us to build an achievable scheme working with subpacketization $S_{L_c} = LK'\gamma\binom{K'}{K'\gamma}$[4] and achieving the theoretical DoF of $LK\gamma$. As we will see later on in this section, the reduced subpacketization leads to an effective $\overline{DoF}_{L_c}$ which is L_c times higher than the one achieved by the XOR based strategy. The details of the algorithm and the analysis of the performance is given in the following subsections.

2.5.1 Description of the Scheme

As a first step, all possible subsets $Q \subset [K']$ of size $|Q| = K'\gamma + 1$ are constructed. Let us now focus on a given subset Q. With a similar reasoning as in the XOR based strategy, each user in group $\mathcal{G}_i$ such that $i \in Q$ has $K'\gamma$ bunches of L subfiles that are not stored in the caches of users belonging to one of the other $K'\gamma$ groups, say users in group $\mathcal{G}_j,\ j \in Q\backslash\{i\}$, and known by all the users in the remaining $K'\gamma - 1$ groups, i.e. groups $\mathcal{G}_k, \forall k \in Q\backslash\{i,j\}$. Since each group contains L_c users and assuming that all users request a different file from the library, group $\mathcal{G}_i$ has $L_cK'\gamma L = K\gamma L$ subfiles to transmit which are desired by the other $K'\gamma$ groups and which can be decoded without interference. All these subfiles are transmitted in one channel use and each bunch of L subfiles is intended for one of the receiving $K\gamma$ users. All users in the transmitting group participate in the transmission and all together act as a unique distributed transmitter.

Before proceeding to the detailed description of the algorithm, let us define $\mathbf{H}_{\mathcal{G}_i\mathcal{G}_j}$ to be the channel matrix between the transmitting group $\mathcal{G}_i$ and receiving group $\mathcal{G}_j$, i.e.

$$\mathbf{H}_{\mathcal{G}_i\mathcal{G}_j} \triangleq \begin{bmatrix} \mathbf{H}_{ij} & \cdots & \mathbf{H}_{(i+K-\frac{K}{L_c})j} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{i(j+K-\frac{K}{L_c})} & \cdots & \mathbf{H}_{(i+K-\frac{K}{L_c})(j+K-\frac{K}{L_c})} \end{bmatrix} \tag{2.11}$$

where we remind that $\mathbf{H}_{i,j}$ represents the $L \times L$ channel matrix between user i and user j.

[4]We assume that K is a multiple integer of the chosen cooperation factor L_c

Transmitting Phase

In the delivery phase, for each of the $\binom{K'}{K'\gamma+1}$ sets Q of size $|Q| = K'\gamma + 1$, let us denote the set of active users in a given time slot by $\mathcal{U}_Q = \cup_{i\in Q}\mathcal{G}_i$. Consider sequentially now all the groups $\mathcal{G}_i$ such that $i \in Q$, all the L_c users within this group will act as a distributed MIMO transmitter to create an $L_cL \times 1$ vector $\mathbf{X}_{\mathcal{G}_i,\mathcal{U}_Q\backslash\{\mathcal{G}_i\}}$ which will multicast to the $K\gamma$ users in the set $\mathcal{U}_Q\backslash\{\mathcal{G}_i\}$.

$\mathbf{X}_{\mathcal{G}_i,\mathcal{U}_Q\backslash\{\mathcal{G}_i\}}$ is constructed as the sum of $K'\gamma$ precoded vectors $\mathbf{V}_{\mathcal{G}_i\mathcal{G}_{k'}}$ of size $L_cL \times 1$. Each vector is intended for a different group $\mathcal{G}_{k'}, k' \in Q\backslash\{i\}$ and it is precoded by a matrix which is the inverse of the channel matrix between the transmitting group $\mathcal{G}_i$ and the group of users $\mathcal{G}_{k'}$ for which that vector is intended. This precoding matrix is more commonly known as Zero Forcing (ZF) precoder. The constructed vector, transmitted by group $\mathcal{G}_i$, takes the form

$$\mathbf{X}_{\mathcal{G}_i,\mathcal{U}_Q\backslash\{\mathcal{G}_i\}} = \sum_{\mathcal{G}_{k'}\subset\mathcal{U}_Q\backslash\{\mathcal{G}_i\}} \mathbf{H}^{-1}_{\mathcal{G}_{k'}\mathcal{G}_i}\mathbf{V}_{\mathcal{G}_i\mathcal{G}_{k'}} \tag{2.12}$$

Each vector $\mathbf{V}_{\mathcal{G}_i\mathcal{G}_{k'}}$ is the concatenation of L_c vectors of size $L \times 1$, each intended for a different user of the intended group $\mathcal{G}_{k'}$. Precisely, $\mathbf{V}_{\mathcal{G}_i\mathcal{G}_{k'}}$ is constructed as follows:

$$\mathbf{V}_{\mathcal{G}_i\mathcal{G}_{k'}} = \left[W^{d_{\mathcal{G}_{k'}(1)}}_{Q\backslash\{k'\},i,1} \cdots W^{d_{\mathcal{G}_{k'}(1)}}_{Q\backslash\{k'\},i,L} \cdots W^{d_{\mathcal{G}_{k'}(L_c)}}_{Q\backslash\{k'\},i,1} \cdots W^{d_{\mathcal{G}_{k'}(L_c)}}_{Q\backslash\{k'\},i,L}\right]^T \tag{2.13}$$

where the symbol $\mathcal{G}_i(l)$ denote the l-th user in group $\mathcal{G}_i$. After the construction of $\mathbf{X}_{\mathcal{G}_i,\mathcal{U}_Q\backslash\{\mathcal{G}_i\}}$, each user $j = \mathcal{G}_i(l)$ in transmitting group $\mathcal{G}_i$ transmits:

$$\mathbf{X}_{j,\mathcal{U}_Q\backslash\{\mathcal{G}_i\}} = E_j \sum_{\mathcal{G}_{k'}\in\mathcal{U}_Q\backslash\{\mathcal{G}_i\}} \mathbf{H}^{-1}_{\mathcal{G}_{k'}\mathcal{G}_i}\mathbf{V}_{\mathcal{G}_i\mathcal{G}_{k'}} \tag{2.14}$$

where $E_j \triangleq [\underline{0}\cdots\underline{0}I_L\underline{0}\cdots\underline{0}]$ is an $L \times LL_c$ matrix selecting the l-th $L \times 1$ vector from an $L_cL \times 1$ vector and I_L is an $L \times L$ identity matrix placed in the l-th $L \times L$ block (sub-matrix) of E_j.

A compact description of the transmission scheme is provided in the form of pseudocode in Algorithm 5.

Decoding phase

A user k belonging to a receiving group $\mathcal{G}_r : r \in Q\backslash\{i\}$, with $\mathcal{G}_i$ being the transmitting group, receives:

$$\mathbf{y}_k = E_k\mathbf{V}_{\mathcal{G}_i\mathcal{G}_r} + \underbrace{E_k\mathbf{H}_{\mathcal{G}_r\mathcal{G}_i} \sum_{\mathcal{G}_{k'}\in\mathcal{U}_Q\backslash\{\mathcal{G}_i,\mathcal{G}_r\}} \mathbf{H}^{-1}_{\mathcal{G}_{k'}\mathcal{G}_i}\mathbf{V}_{\mathcal{G}_i\mathcal{G}_{k'}}}_{interference\ term\ I_k} \tag{2.15}$$

where the first addend in equation (2.15) is an $L \times 1$ vector of subfiles desired by user k and the term I_k is an interference term containing subfiles desired by users in the other intended receiving groups, i.e. groups $\mathcal{G}_{k'} \in \mathcal{U}_Q\backslash\{\mathcal{G}_i, \mathcal{G}_r\}$.

```
1 procedure DELIVERY($W^1, ..., W^N,\ d_1, ..., d_K$)
2 for $Q \subset [K'] \ : \ |Q| = K'\gamma + 1$ do
3     $\mathcal{U}_Q = \bigcup_{i \in Q} \mathcal{G}_i$
4     for $i \in Q$ do
5         Transmit $\underline{X}_{\mathcal{G}_i, \mathcal{U}_Q \setminus \{\mathcal{G}_i\}} = \sum_{\mathcal{G}_{k'} \subset \mathcal{U}_Q \setminus \{\mathcal{G}_i\}} \mathbf{H}^{-1}_{\mathcal{G}_{k'}\mathcal{G}_i} \begin{bmatrix} W^{d_{\mathcal{G}_{k'}(1)}}_{Q \setminus \{k'\}, i, 1} \\ \vdots \\ W^{d_{\mathcal{G}_{k'}(1)}}_{Q \setminus \{k'\}, i, L} \\ \vdots \\ W^{d_{\mathcal{G}_{k'}(L_c)}}_{Q \setminus \{k'\}, i, 1} \\ \vdots \\ W^{d_{\mathcal{G}_{k'}(L_c)}}_{Q \setminus \{k'\}, i, L} \end{bmatrix}$
6     end
7 end
8 end procedure
```

Algorithm 5: Cooperation Based Delivery Algorithm

For this algorithm, we assume that each user has global perfect CSI knowledge, thus user k knows all the channel matrices $\mathbf{H}_{\mathcal{G}_{k'}\mathcal{G}_i}, k' \in Q \setminus \{i\}$. Moreover, all the $\mathbf{V}_{\mathcal{G}_i \mathcal{G}_{k'}}$ in I_k contain elements of the form $W^{d_{\mathcal{G}_{k'}(l)}}_{Q \setminus \{k'\}, i, l} : k' \neq r$, which are known by user k since it has them in its cache. Thus, user k can reconstruct the interference term I_k and subtract it from the received signal $\mathbf{y}_k$. At the end of the decoding phase, user k gets free of inter and intra group interference the following L desired subfiles:

$$E_k \mathbf{V}_{\mathcal{G}_i \mathcal{G}_r} = \left[W^{d_k}_{Q \setminus \{r\}, i, 1} \cdots W^{d_k}_{Q \setminus \{r\}, i, L} \right]^T . \tag{2.16}$$

2.5.2 Performance Evaluation

The algorithm creates $\binom{K'}{K'\gamma+1}$ sets Q and for each of them $K'\gamma + 1$ transmissions occur. Hence, the scheme takes $\binom{K'}{K'\gamma+1}(K'\gamma + 1)$ transmissions to serve all K users. Since each transmitted precoded vector needs $\frac{1}{\binom{K'}{K'\gamma} L K'\gamma}$ units of time to be transmitted, the overall delay is

$$R_{L_c}(M) = \frac{(K'\gamma + 1)\binom{K'}{K'\gamma+1}}{LK'\gamma\binom{K'}{K'\gamma}} = \frac{K(1-\gamma)}{LK\gamma} \tag{2.17}$$

and it is achieved with subpacketization $S_{L_c} = LK'\gamma\binom{K'}{K'\gamma}$. As a consequence, under the subpacketization constraint S_{max}, the maximum number of users over which we can encode becomes

$$\bar{K}_{L_c} = \underset{K}{argmax} \{ K : L \frac{K}{L_c} \gamma \binom{\frac{K}{L_c}}{\frac{K}{L_c}\gamma} \leq S_{max} \} \tag{2.18}$$

Thus, the effective achievable rate is

$$\bar{R}_{L_c}(M) = \frac{K}{\bar{K}_{L_c}} \frac{\bar{K}_{L_c}(1-\gamma)}{L\bar{K}_{L_c}\gamma} \tag{2.19}$$

We can now notice that $\bar{K}_{L_c} = L_c\bar{K}$, where $\bar{K}$ is the maximum number of nodes over which is possible to encode is the XOR based strategy is used (which uses $L_c = 1$). Hence, the achieved effective sum DoF is

$$\overline{DoF}_{L_c}(\gamma) = LL_c\bar{K}\gamma \tag{2.20}$$

which is L_c times higher than the one achieved by the XOR based algorithm under the subpacketization constraint.

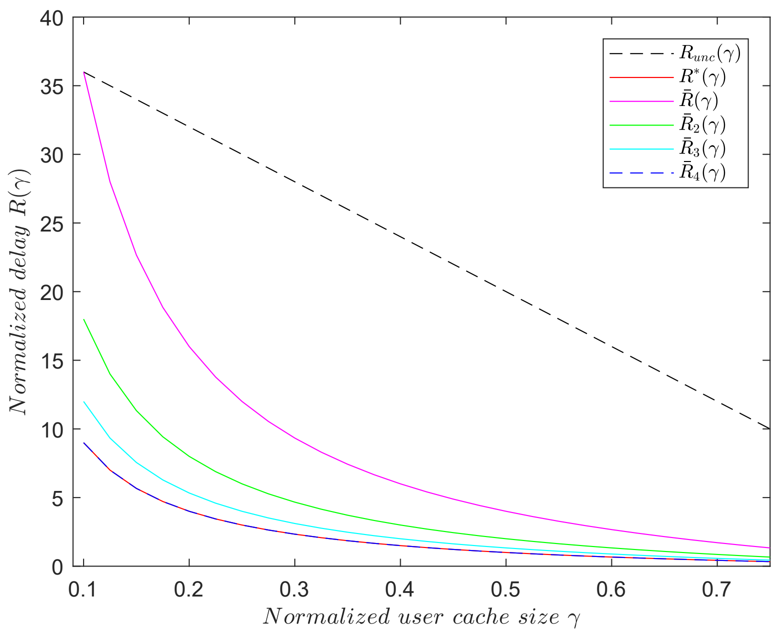

Figure 2.2. The picture shows the achieved rate as a function of the per user normalized cache size. The rates are the ones for a system with $N = K = 40$, $L = 1$ and $\bar{K} = 10$. $R_{unc}(\gamma)$ refers to the uncoded scheme, $R^*(\gamma)$ to the theoretical optimal rate, $\bar{R}(\gamma)$ to the rate achieved by the XOR based scheme under the subpacketization constraint and $R_{L_c}(\gamma), L_c = 2,3,4$ to the rates achieved by the coopearation based scheme.

Figure 2.2 helps us to understand the benefits of coding and users' cooperation. The plot shows the achievable rates for a system with $N = K = 40$, $L = 1$ and $\bar{K} = 10$ (due to a given subpacketization S_{max}). First, we notice that even if the maximum allowable subpacketiation limits the maximum number of users over which it is possible to encode to only the 25% of the total number of users, the rate achieved by the coded scheme (XOR based algorithm) is significantly smaller than the one achieved by the uncoded scheme, pink and black dashed lines respectively. Secondly, we see how by allowing some cooperation among users we can further bring down the delay. Finally, a cooperation factor of $L_c = 4$ would allow to achieve the theoretical maximum degree of freedom, whose corresponding rate is the one given by the red line, even under the subpacketization constraint. The multiplicative gains arising from cooperation are evident from the plot.However, we recall that the increased gains come at the expenses of increased CSI knowledge.

2.6 A Low Subpacketization Strategy with only Local CSI Knowledge

In this section we present a novel scheme that can be useful when global CSI is not available at users and there is a subpacketization constraint S_{max}. Nevertheless, this algorithm achieves a lower theoretical performance than the other two already discussed.

The following discussion is kept on purpose informal in order to give an idea of the proposed scheme and its achieved performance. Without caching, a degree of freedom of L within the same interference limited area can be easily achieved thanks to the L per user antennas. With caching, thanks to the generalized content placement strategy with cooperation factor L_c, the side information (cached content) available at each user allows easily to serve, via D2D communication, $\frac{K}{L_c}\gamma$ users with only one transmission. Hence, from the two above observations we can easily infer that a total number of $LK'\gamma$ subfiles can be successfully delivered with one (multicast) transmission. This DoF of $LK'\gamma$ is easily achievable by repeating L_c times the described XOR based scheme. At each time the scheme would serve $K' = \frac{K}{L_c}$ users, each belonging to a different group. However, if $L_c > 1$, the cached content makes possible an additional out of interference unicast transmission in parallel to the aforementioned multicast one. As a result, taking advantage of the grouping approach used in the scheme presented in the previous section, this novel scheme achieves a theoretical sum DoF of $DoF(\gamma) = L(K'\gamma + 1)$ with reduced subpacketizaticon S_{L_c}. There is clearly a tradeoff between the achieved degree of freedom and the subpacketization. Given the subpacketization constraint S_{max}, there exist a minimum value of L_c that satisfies the constraint. We remind that in practice the smaller is the payload (a subfile to transmit) the higher will be the performance loss due to header which is of fixed size. As a consequence, one may want to use an higher value of L_c to further reduce the subpacketization. On the other hand, by reducing the subpacketization, i.e. by increasing L_c, we also reduce the achievable DoF (unless users have global CSI knowledge so that the cooperation based scheme of the previous section can be used). In the next section we present and discuss in details the proposed delivery scheme.

2.6.1 Description of the Scheme

Being the scheme inspired again by [7] and using a cooperation factor L_c bigger than 1, we can focus on a subset $Q \subset [K']$ of size $K'\gamma+1$. Before proceeding with the description, let us define the following useful subsets

$$I_{Q,l_c} = (\mathcal{G}_i(l_c) : \forall i \in Q), \ \forall l_c \in [L_c] \tag{2.21}$$

which contain the l_c-th user of all groups $\mathcal{G}_i, \forall i \in Q$.

Transmission Algorithm

Having in mind how the XOR based scheme of the previous section works, let us fix a given subset Q and let us select one element from Q, say j. Next, we sequentially select all groups $\mathcal{G}_i$ such that $i \in Q \backslash \{j\}$. For each of these groups, say $\mathcal{G}_i$, the algorithm works as described below.

Sequentially, pick user $\mathcal{G}_i(l_c)$ and let him multicast a $L \times 1$ vector to the l_c-th user of the other groups in the subset Q, i.e. to groups $\mathcal{G}_p$ such that $p \in Q \backslash \{i\}$. The transmitted vector from $\mathcal{G}_i(l_c)$ takes the form

$$\mathbf{X}_{\mathcal{G}_i(l_c), I_{Q,l_c} \backslash \{\mathcal{G}_i(l_c)\}} = \begin{bmatrix} \bigoplus\limits_{k \in I_{Q,l_c} \backslash \{\mathcal{G}_i(l_c)\}} W^{d_k}_{Q \backslash \{\mathcal{G}^{-1}(k)\}, i, 1} \\ \vdots \\ \bigoplus\limits_{k \in I_{Q,l_c} \backslash \{\mathcal{G}_i(l_c)\}} W^{d_k}_{Q \backslash \{\mathcal{G}^{-1}(k)\}, i, L} \end{bmatrix} \tag{2.22}$$

where $\mathcal{G}^{-1}(k)$ is a function $f : [K] \to [K']$ obtaining the index of the group user k belongs to. In parallel to the transmission in equation (2.22) a unicast message is transmitted from a user of group $\mathcal{G}_j$, which we denote by t, to one user of group $\mathcal{G}_i$. User t is selected (potentially even randomly) from the group $\mathcal{G}_j$ among the users not beneficing from transmission (2.22). The intended user for the unicast transmission is selected to be the subsequent user, in group $\mathcal{G}_i$, to the user transmitting the multicast vector, i.e. the user beneficing of the unicast transmission is $\mathcal{G}_i(1 + l_c \mod L_c)$. The transmitted unicast vector is the following:

$$\mathbf{X}_{t, \mathcal{G}_i(1+l_c \mod L_c)} = \begin{bmatrix} W^{d_{\mathcal{G}_i(1+l_c \mod L_c)}}_{Q \backslash \{i\}, j, 1} \\ \vdots \\ W^{d_{\mathcal{G}_i(1+l_c \mod L_c)}}_{Q \backslash \{i\}, j, L} \end{bmatrix} \tag{2.23}$$

A pseudo-code of the transmission scheme is given in algorithm 6.

```
1  procedure DELIVERY($W^1, ..., W^N,\ d_1, ..., d_K$)
2  for $Q \subset [K']\ :\ |Q| = K'\gamma + 1$ do
3      $I_{Q,l_c} = (\mathcal{G}_i(l_c) : \forall i \in Q), \forall l_c \in [L_c]$
4      Select any $j \in Q$
5      for $i \in Q \backslash \{j\}$ do
6          for $l_c \in [L_c]$ do
7              Transmit
8              $\mathbf{X}_{\mathcal{G}_i(l_c), I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}} = \begin{bmatrix} \bigoplus_{k \in I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}} W^{d_k}_{Q\backslash\{\mathcal{G}^{-1}(k)\},i,1} \\ \vdots \\ \bigoplus_{k \in I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}} W^{d_k}_{Q\backslash\{\mathcal{G}^{-1}(k)\},i,L} \end{bmatrix}$
9              In parallel:
10             Select any $t \in \mathcal{G}_j \backslash \{\mathcal{G}_j \cap I_{Q,l_c}\}$
11             Transmit
12             $\mathbf{X}_{t,\mathcal{G}_i(1+l_c \mod L_c)} = \begin{bmatrix} W^{d_{\mathcal{G}_i(1+l_c \mod L_c)}}_{Q\backslash\{i\},j,1} \\ \vdots \\ W^{d_{\mathcal{G}_i(1+l_c \mod L_c)}}_{Q\backslash\{i\},j,L} \end{bmatrix}$
13         end
14     end
15 end
16 end procedure
```

Algorithm 6: Low subpacketization delivery strategy with local CSI

Decoding phase

In this paragraph we describe the decoding process performed at a given receiving user. Let us denote by r a user belonging to the set $I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}$ for which the transmission $\mathbf{X}_{\mathcal{G}_i(l_c),I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}}$ is intended. User r receives the signal

$$\mathbf{y}_r = \mathbf{H}_{r,\mathcal{G}_i(l_c)} \cdot \mathbf{X}_{\mathcal{G}_i(l_c),I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}} + \mathbf{H}_{r,t} \cdot \mathbf{X}_{t,\mathcal{G}_i(1+l_c \mod L_c)}{}^{5}. \tag{2.24}$$

Step 1 In the first step of the decoding phase, user r creates the signal $\mathbf{H}_{r,t} \cdot \mathbf{X}_{t,\mathcal{G}_i(1+l_c \mod L_c)}$. This is possible thanks to the knowledge of local CSI and because $\mathbf{X}_{t,\mathcal{G}_i(1+l_c \mod L_c)}$ is a vector of subfiles with index $\tau = Q\backslash\{i\}$ which user r has stored in its cache since $r \in \mathcal{G}_p, p \in Q\backslash\{i\}$. User r proceed to subtract the created signal from the received one:

$$\bar{\mathbf{y}}_r = \mathbf{y}_r - \mathbf{H}_{r,t} \cdot \mathbf{X}_{t,\mathcal{G}_i(1+l_c \mod L_c)} = \mathbf{H}_{r,\mathcal{G}_i(l_c)} \cdot \mathbf{X}_{\mathcal{G}_i(l_c),I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}} \tag{2.25}$$

[5] We recall that the noise is omitted because our analysis is done in the high SNR regime.

Step 2 At this point user r applies a ZF decoder to null out the effect of the channel $\mathbf{H}_{r,\mathcal{G}_i(l_c)}$:

$$\mathbf{H}^{-1}_{r,\mathcal{G}_i(l_c)}\bar{\mathbf{y}}_r = \mathbf{X}_{\mathcal{G}_i(l_c),I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}} \tag{2.26}$$

Step 3 In the next step, user r constructs from its cache the vector

$$\begin{bmatrix} \bigoplus\limits_{k\in I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c),r\}} W^{d_k}_{Q\backslash\{\mathcal{G}^{-1}(k)\},i,1} \\ \vdots \\ \bigoplus\limits_{k\in I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c),r\}} W^{d_k}_{Q\backslash\{\mathcal{G}^{-1}(k)\},i,L} \end{bmatrix} \tag{2.27}$$

which it subtracts from $\mathbf{X}_{\mathcal{G}_i(l_c),I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}}$ to finally get with no error its desired subfiles $\{W^{d_r}_{Q\backslash\mathcal{G}^{-1}(r),i,1}\cdots W^{d_r}_{Q\backslash\mathcal{G}^{-1}(r),i,L}\}$.

Let us now briefly describe the decoding process of the user $\mathcal{G}_i(1+l_c \mod L_c)$ which receives the signal

$$\begin{aligned}&\mathbf{y}_{\mathcal{G}_i(1+l_c \mod L_c)} = \\ &\mathbf{H}_{\mathcal{G}_i(1+l_c \mod L_c),\mathcal{G}_i(l_c)}\cdot\mathbf{X}_{\mathcal{G}_i(l_c),I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}} + \mathbf{H}_{\mathcal{G}_i(1+l_c \mod L_c),t}\cdot\mathbf{X}_{t,\mathcal{G}_i(1+l_c \mod L_c)}.\end{aligned} \tag{2.28}$$

User $\mathcal{G}_i(1+l_c \mod L_c)$ creates the signal $\mathbf{H}_{\mathcal{G}_i(1+l_c \mod L_c),\mathcal{G}_i(l_c)}\cdot\mathbf{X}_{\mathcal{G}_i(l_c),I_{Q,l_c}\backslash\{\mathcal{G}_i(l_c)\}}$ thanks to its cache and local CSI. Next, it subtracts the latter from the received signal and finally it applies the ZF decoder $\mathbf{H}^{-1}_{\mathcal{G}_i(1+l_c \mod L_c),t}$ to get its desired subfiles.

2.6.2 Performance Evaluation

The algorithm operates in $\binom{K'}{K'\gamma+1}$ rounds, one for each set Q of size $|Q| = K'\gamma+1$. Fixed the set Q, L_c sets of 2 parallel transmissions occur for each of the $K'\gamma$ elements of the set $Q\backslash\{j\}, j\in Q$. Thus, the total number of (2 parallel) transmissions is

$$\binom{K'}{K'\gamma+1}K'\gamma L_c. \tag{2.29}$$

Since each transmission is of duration $\frac{1}{LK'\gamma\binom{K'}{K'\gamma}}$, the rate achieved by the scheme is

$$R^{low\ CSI}_{L_c}(M) = \frac{\binom{K'}{K'\gamma+1}K'\gamma L_c}{LK'\gamma\binom{K'}{K'\gamma}} \tag{2.30}$$

which after some elementary math boils down to

$$R^{low\ CSI}_{L_c}(M) = \frac{K(1-\gamma)}{L(K'\gamma+1)} \tag{2.31}$$

The analysis under the subpacketization constraint S_{max} is the same as in the subsection 2.5.2 since the subpacketization required by the scheme is the same as the one required by the cooperation based algorithm. Hence, the maximum number of user over which is possible to encode is $\bar{K}_{L_c} = L_c\bar{K}$ and leads to a delay of

$$\bar{R}_{L_c}^{low\ CSI}(M) = \frac{K}{\bar{K}_{L_c}} \frac{\bar{K}_{L_c}(1-\gamma)}{L(\frac{\bar{K}_{L_c}}{L_c}\gamma+1)} = \frac{K(1-\gamma)}{L(\frac{\bar{K}_{L_c}}{L_c}\gamma+1)} = \frac{K(1-\gamma)}{L(\bar{K}\gamma+1)} \tag{2.32}$$

The effective degree of freedom achieved by the scheme is

$$\overline{DoF}(\gamma) = L(\bar{K}\gamma+1). \tag{2.33}$$

2.7 Examples

In order to provide a better understanding of the cache placement and the three proposed delivery schemes, we illustrate a different example for each the proposed schemes and we compare the performance of the schemes in all the three cases. For the sake of brevity, the examples presented below are meant to give the reader only an idea of how the schemes work and thus they are not complete. It is left to the reader to complete them in order to check the correctness.

Example using the XOR based scheme

Consider a network with $K = 8$ users, each equipped with $L = 2$ antennas. Each user has got enough memory so that the normalized cache size is $\gamma = \frac{1}{2}$. Without loss of generality, we assume that the requested demand vector is $\mathbf{d} = (1,2,3,4,5,6,7,8)$, i.e. user 1 requests file W^1, user 2 requests file W^2, etc. In the delivery phase we are going to use the XOR based scheme which does not require users' cooperation at the expenses of an extremely high subpacketization. Thus, since $L_c = 1$, the subpacketization is $S_1 = 2 \cdot 4\binom{8}{4} = 560$, while the respective cached content at each user is respectively:

$$\begin{aligned}
Z_1 &= \{W^n_{1234,1,l}, ..., W^n_{1234,4,l}, W^n_{1235,1,l}, ..., W^n_{1678,8,l}\ \forall l \in [2], \forall n \in [8]\} \\
Z_2 &= \{W^n_{1234,1,l}, ..., W^n_{1234,4,l}, W^n_{1235,1,l}, ..., W^n_{2678,8,l}\ \forall l \in [2], \forall n \in [8]\} \\
&\vdots \\
Z_8 &= \{W^n_{1238,1,l}, ..., W^n_{1238,8,l}, W^n_{1248,1,l}, ..., W^n_{5678,8,l}\ \forall l \in [2], \forall n \in [8]\}
\end{aligned} \tag{2.34}$$

In the delivery phase, let us focus on the subset $Q = \{12345\}$ of size $|Q| = K\gamma+1 = 5$. The 5 transmissions associated to this subset are:

$$\mathbf{X}_{1,2345} = \begin{bmatrix} W^2_{1345,1,1} \oplus W^3_{1245,1,1} \oplus W^4_{1235,1,1} \oplus W^5_{1234,1,1} \\ W^2_{1345,1,2} \oplus W^3_{1245,1,2} \oplus W^4_{1235,1,1} \oplus W^5_{1234,1,1} \end{bmatrix} \tag{2.35}$$

$$\mathbf{X}_{2,1345} = \begin{bmatrix} W^1_{2345,2,1} \oplus W^3_{1245,2,1} \oplus W^4_{1235,2,1} \oplus W^5_{1234,2,1} \\ W^1_{2345,2,2} \oplus W^3_{1245,2,2} \oplus W^4_{1235,2,1} \oplus W^5_{1234,2,1} \end{bmatrix} \tag{2.36}$$

$$\mathbf{X}_{3,1245} = \begin{bmatrix} W^1_{2345,3,1} \oplus W^2_{1345,3,1} \oplus W^4_{1235,3,1} \oplus W^5_{1234,3,1} \\ W^1_{2345,3,2} \oplus W^2_{1345,3,2} \oplus W^4_{1235,3,1} \oplus W^5_{1234,3,1} \end{bmatrix} \tag{2.37}$$

$$\mathbf{X}_{4,1235} = \begin{bmatrix} W^1_{2345,4,1} \oplus W^2_{1345,4,1} \oplus W^3_{1245,4,1} \oplus W^5_{1234,4,1} \\ W^1_{2345,4,1} \oplus W^2_{1345,4,2} \oplus W^3_{1245,4,2} \oplus W^5_{1234,4,1} \end{bmatrix} \tag{2.38}$$

$$\mathbf{X}_{5,1234} = \begin{bmatrix} W^1_{2345,5,1} \oplus W^2_{1345,5,1} \oplus W^3_{1245,5,1} \oplus W^4_{1235,5,1} \\ W^1_{2345,5,1} \oplus W^2_{1345,5,2} \oplus W^3_{1245,5,2} \oplus W^4_{1235,5,1} \end{bmatrix} \tag{2.39}$$

Let us have a look at equation (2.35) which illustrates the 2×1 vector transmitted by user 1 and intended for users 2,3,4,5 which will receive and decode the vector using a Zero-Forcing receiver thanks to the knowledge of the channel matrix $\mathbf{H}_{j,1}, j \in \{2,3,4,5\}$. After decoding, each of the intended users can get their desired subfiles thanks to the cached content. For instance, user 2 constructs from its cache the vector

$$\begin{bmatrix} W^3_{1245,1,1} \oplus W^4_{1235,1,1} \oplus W^5_{1234,1,1} \\ W^3_{1245,1,2} \oplus W^4_{1235,1,1} \oplus W^5_{1234,1,1} \end{bmatrix} \tag{2.40}$$

which can subtract from the decoded vector in (2.35) to get the desired subfiles $W^2_{1345,1,1}$ and $W^2_{1345,1,2}$. We remind the reader that the subfiles in (2.40) are placed in the cache of user 4 since their indices τ contain the element 4 (see the content placement strategy). With the same process, users 3,4,5 get their desired subfiles from the decoded vector $\mathbf{X}_{1,2345}$. We leave to the reader to check that after the transmissions associated to the other subsets Q, each user will have obtained all subfiles of the requested file. Since each transmission has normalized duration $\frac{1}{560}$ and the total number of transmissions is $(K\gamma+1)\binom{K}{K\gamma+1} = 5\binom{8}{5} = 280$, the total delivery time is merely $R(M) = \frac{280}{560} = \frac{1}{2}$.

Example using the Cooperation based scheme

Consider the same network of the previous example but we now group users in groups of $L_c = 2$ users. A possible grouping can be the following:

$$\mathcal{G}_1 = \{1\ 5\}, \mathcal{G}_2 = \{2\ 6\}, \mathcal{G}_3 = \{3\ 7\}, \mathcal{G}_4 = \{4\ 8\} \tag{2.41}$$

In the placement phase, all users within the same group cache the same content, thus the subpacketization reduces to $S_2 = 2 \cdot 2\binom{4}{2} = 24$ which is $\frac{560}{24} \approx 23$ times smaller than the subpacketization needed by the XOR based scheme. We now use the generalized placement strategy with $L_c = 2$ to store content at the users' caches which are illustrated below:

$$\begin{aligned} Z_1 = Z_5 &= \{W^n_{12,1,l}, W^n_{12,2,l}, W^n_{13,1,l}, W^n_{13,3,l}, W^n_{14,1,l}, W^n_{14,4,l}, \forall l \in [2], \forall n \in [8]\}, \\ Z_2 = Z_6 &= \{W^n_{12,1,l}, W^n_{12,2,l}, W^n_{23,2,l}, W^n_{23,3,l}, W^n_{24,2,l}, W^n_{24,4,l}, \forall l \in [2], \forall n \in [8]\}, \\ Z_3 = Z_7 &= \{W^n_{13,1,l}, W^n_{13,3,l}, W^n_{23,2,l}, W^n_{23,3,l}, W^n_{34,3,l}, W^n_{34,4,l}, \forall l \in [2], \forall n \in [8]\}, \\ Z_4 = Z_8 &= \{W^n_{14,1,l}, W^n_{14,4,l}, W^n_{24,2,l}, W^n_{24,4,l}, W^n_{34,3,l}, W^n_{34,4,l}, \forall l \in [2], \forall n \in [8]\}. \end{aligned} \tag{2.42}$$

In the delivery phase, we assume again that the demand vector is $\mathbf{d} = (1,2,3,4,5,6,7,8)$. Let us focus on the subset $Q = \{1,2,3\}$. The first transmission associated to this subset is from group $\mathcal{G}_1$ and it is intended for groups $\mathcal{G}_2$ and $\mathcal{G}_3$. Thus, users 1 and 5, which share the same cached content, will act as a distributed transmitter to partially serve users 2,6,3,7. Assuming both users 1 and 5 know the channel matrices $\mathbf{H}_{\mathcal{G}_2\mathcal{G}_1}$ and $\mathbf{H}_{\mathcal{G}_3\mathcal{G}_1}$, the signal they transmit is

$$\mathbf{X}_{\mathcal{G}_1,\mathcal{G}_2\mathcal{G}_3} = \mathbf{H}^{-1}_{\mathcal{G}_2\mathcal{G}_1} \begin{bmatrix} W^2_{13,1,1} \\ W^2_{13,1,2} \\ W^6_{13,1,1} \\ W^6_{13,1,2} \end{bmatrix} + \mathbf{H}^{-1}_{\mathcal{G}_3\mathcal{G}_1} \begin{bmatrix} W^3_{12,1,1} \\ W^3_{12,1,2} \\ W^7_{12,1,1} \\ W^7_{12,1,2} \end{bmatrix} \tag{2.43}$$

The transmission of users 1 ad 5 is received by all users, particularly the ones in group $\mathcal{G}_2$ and $\mathcal{G}_3$ for which the signal in equation (2.43) is intended. Let us focus on user 2 which, thanks to precoding, receives (neglecting noise):

$$\mathbf{y}_2 = \begin{bmatrix} W^2_{13,1,1} \\ W^2_{13,1,2} \end{bmatrix} + \underbrace{E_2 \mathbf{H}_{\mathcal{G}_2\mathcal{G}_1} \mathbf{H}^{-1}_{\mathcal{G}_3\mathcal{G}_1} \begin{bmatrix} W^3_{12,1,1} \\ W^3_{12,1,2} \\ W^7_{12,1,1} \\ W^7_{12,1,2} \end{bmatrix}}_{interference\ term} \tag{2.44}$$

where $E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$.

We now notice that the index τ of the subfiles in the interference term contain element 2. Consequently, thanks to the design of the cache placement, all the subfiles in the interference term are stored into the cache of user 2. Moreover, we remind that perfect global CSI is available at each node. As a result, user 2 can reconstruct the interference term and subtract it from $\mathbf{y}_2$. After this deconding phase, user 2 obtains his desired subfiles $\{W^2_{13,1,l}\}^2_{l=1}$. Similarly, users 6,3 and 7 get their desired subfiles from the transmitted vector $\mathbf{X}_{\mathcal{G}_1,\mathcal{G}_2\mathcal{G}_3}$.

Analogously, the other 2 transmissions associated to the above subset $Q = \{1,2,3\}$ are:

$$\mathbf{X}_{\mathcal{G}_2,\mathcal{G}_1\mathcal{G}_3} = \mathbf{H}^{-1}_{\mathcal{G}_1\mathcal{G}_2} \begin{bmatrix} W^1_{23,2,1} \\ W^1_{23,2,2} \\ W^5_{23,2,1} \\ W^5_{23,2,2} \end{bmatrix} + \mathbf{H}^{-1}_{\mathcal{G}_3\mathcal{G}_2} \begin{bmatrix} W^3_{12,2,1} \\ W^3_{12,2,2} \\ W^7_{12,2,1} \\ W^7_{12,2,2} \end{bmatrix} \tag{2.45}$$

$$\mathbf{X}_{\mathcal{G}_3,\mathcal{G}_1\mathcal{G}_2} = \mathbf{H}^{-1}_{\mathcal{G}_1\mathcal{G}_3} \begin{bmatrix} W^1_{23,3,1} \\ W^1_{23,3,2} \\ W^5_{23,3,1} \\ W^5_{23,3,2} \end{bmatrix} + \mathbf{H}^{-1}_{\mathcal{G}_2\mathcal{G}_3} \begin{bmatrix} W^2_{13,3,1} \\ W^2_{13,3,2} \\ W^6_{13,3,1} \\ W^6_{13,3,2} \end{bmatrix} \tag{2.46}$$

Again, users can retrieve their desired subfiles with no interference thanks to their caches, the precoding matrix used and knowledge of CSI. We leave to the reader to create the transmitted vectors for the other subsets Q and to check the complete reception of all subfiles by each user after all transmissions are performed.

The total number of transmissions is $3 \cdot \binom{4}{3} = 12$. Thus, the total delivery time is $R(M) = \frac{12}{24} = \frac{1}{2}$. The delivery time turns out to be the same as the one given by the XOR based scheme. However, if there was a subpacketization constraint (for example the maximum number of user over which we could encode was $\bar{K} = 4$ and not 8) this scheme would achieve a better performance than the XOR based scheme.

Example for the low subpacketization scheme with local CSI

Consider a network with $K = 12$ users with normalized cache size $\gamma = \frac{1}{2}$. Each user is equipped with $L = 2$ antennas. We assume that users have only local CSI. For instance, considering users 1,2,3, user 1 knows channel matrices $\mathbf{H}_{1,2}$ and $\mathbf{H}_{1,3}$ but he doesn't know $\mathbf{H}_{2,3}$. Let us put the 12 users in groups of size $L_c = 3$, so that users within the same group cache the same content. The 4 groups are:

$$\mathcal{G}_1 = \{1,5,9\}, \mathcal{G}_2 = \{2,6,10\}, \mathcal{G}_3 = \{3,7,11\}, G_4 = \{4,8,12\} \tag{2.47}$$

For the sake of brevity, the placement phase is skipped since it is analogous to the one presented in the previous example. In the delivery phase, let us focus as usual on the subset $Q = \{1, 2,3\}$ and let us select one element j from Q, for instance $j = 2$. For this set Q, the algorithm operates in $K'\gamma = 2$ rounds of $L_c = 3$ double transmissions (one unicast in parallel to one multicast). In the first round, the 3 parallel transmissions are:

$$1)\ \mathbf{X}_{1,23} = \begin{bmatrix} W^2_{13,1,1} \oplus W^3_{12,1,1} \\ W^2_{13,1,2} \oplus W^3_{12,1,2} \end{bmatrix} \mathbf{X}_{6,5} = \begin{bmatrix} W^5_{23,2,1} \\ W^5_{23,2,2} \end{bmatrix} \tag{2.48}$$

$$2)\ \mathbf{X}_{5,67} = \begin{bmatrix} W^6_{13,1,1} \oplus W^7_{12,1,1} \\ W^6_{13,1,2} \oplus W^7_{12,1,2} \end{bmatrix} \mathbf{X}_{10,9} = \begin{bmatrix} W^9_{23,2,1} \\ W^9_{23,2,2} \end{bmatrix} \tag{2.49}$$

$$3)\ \mathbf{X}_{9,10\ 11} = \begin{bmatrix} W^{10}_{13,1,1} \oplus W^{11}_{12,1,1} \\ W^{10}_{13,1,2} \oplus W^{11}_{12,1,2} \end{bmatrix} \mathbf{X}_{2,1} = \begin{bmatrix} W^1_{23,2,1} \\ W^1_{23,2,2} \end{bmatrix} \tag{2.50}$$

Let us analyse the first set of 2 parallel signals $\mathbf{X}_{1,23}$ and $\mathbf{X}_{6,5}$ transmitted by user 1 and 6 respectively.

User 2 receives the signal $\mathbf{y}_2 = \mathbf{H}_{2,1} \cdot \mathbf{X}_{1,23} + \mathbf{H}_{2,6} \cdot \mathbf{X}_{6,5}$. Since the subfiles $W^5_{23,2,1}$ and $W^5_{23,2,2}$ are both stored at the user 2's cache and thanks to the knowledge of the channel matrix $\mathbf{H}_{2,6}$, user 2 can reconstruct the signal $\mathbf{H}_{2,6}\mathbf{X}_{6,5}$ and subtract it from $\mathbf{y}_2$.

$$\bar{\mathbf{y}}_2 = \mathbf{y}_2 - \mathbf{H}_{2,6}\mathbf{X}_{6,5} = \mathbf{H}_{2,1}\mathbf{X}_{1,23} \tag{2.51}$$

Next, the Zero Forcing decoder $\mathbf{H}^{-1}_{2,1}$ is applied to the signal $\bar{\mathbf{y}}_2$. At this point user 2 has got cleanly the vector $\mathbf{X}_{1,23}$. It can now subtract from the vector $\mathbf{X}_{1,23}$ the subfiles $W^3_{12,1,1}$ and $W^3_{12,1,2}$ which it has in its cache. At the end of the decoding phase, user 2 gets its desired subfiles $W^2_{13,1,1}$ and $W^2_{13,1,2}$. The decoding procedure is the same for user 3.

Let's now see how user 5 decodes its desired files. It receives the signal $\mathbf{y}_5 = \mathbf{H}_{5,1} \cdot \mathbf{X}_{1,23} + \mathbf{H}_{5,6} \cdot \mathbf{X}_{6,5}$ from which it subtracts $\mathbf{H}_{5,1} \cdot \mathbf{X}_{1,23}$.

$$\bar{\mathbf{y}}_5 = \mathbf{y}_5 - \mathbf{H}_{5,1} \cdot \mathbf{X}_{1,23} = \mathbf{H}_{5,6} \cdot \mathbf{X}_{6,5} \tag{2.52}$$

After that, the ZF receiver $\mathbf{H}^{-1}_{5,6}$ is applied to $\bar{\mathbf{y}}_5$. The desired subfiles $W^5_{23,2,1}$ and $W^5_{23,2,2}$ are then obtained.

The decoding process is the same also for the transmissions in equations 2.49 and 2.50.

In the second round we have:

$$4)\ \mathbf{X}_{3,12} = \begin{bmatrix} W^1_{23,3,1} \oplus W^2_{13,3,1} \\ W^1_{13,3,2} \oplus W^2_{12,3,2} \end{bmatrix}, \mathbf{X}_{6,7} = \begin{bmatrix} W^7_{12,2,1} \\ W^7_{12,2,2} \end{bmatrix} \tag{2.53}$$

$$5)\ \mathbf{X}_{7,56} = \begin{bmatrix} W^5_{23,3,1} \oplus W^6_{13,3,1} \\ W^5_{13,3,2} \oplus W^6_{12,3,2} \end{bmatrix}, \mathbf{X}_{10,11} = \begin{bmatrix} W^{11}_{12,2,1} \\ W^{11}_{12,2,2} \end{bmatrix} \tag{2.54}$$

$$6)\ \mathbf{X}_{11,9\ 10} = \begin{bmatrix} W^9_{23,3,1} \oplus W^{10}_{13,3,1} \\ W^9_{13,3,2} \oplus W^{10}_{12,3,2} \end{bmatrix}, \mathbf{X}_{2,3} = \begin{bmatrix} W^3_{12,2,1} \\ W^3_{12,2,2} \end{bmatrix} \tag{2.55}$$

The decoding for the above transmissions is similar as the previous set of transmissions. The delay for this first 6 sets of parallel transmissions associated to set $Q = \{1,2,3\}$ is $\frac{6}{24}$. Overall, there are $\binom{K'}{K'\gamma+1} = \binom{4}{3} = 4$ sets Q. Hence, the total delivery time needed for all users to be served is $R(M) = 4 \cdot \frac{6}{24} = 1$.
It is important to notice that if we had global CSI knowledge, by using the cooperation based algorithm we would achieve a delay of only $R(M) = \frac{12}{24} = \frac{1}{2}$.

2.8 Summary of Results

In this section we summarize the results obtained for the multi-antennas D2D coded caching problem.

Theorem 1. *In the D2D coded caching framework, where each of the K nodes, equipped with L antennas and having a cache of size M, request a different file from a library of N files ($\gamma \triangleq \frac{M}{N}$), the following theoretical degree of freedom is achievable:*

$$DoF(\gamma) = LK\gamma. \tag{2.56}$$

The DoF given in (2.56) is achieved with the caching and delivery scheme described in section 2.4 which are generalizations of the schemes given in [7]. The availability of L antennas at each user leads to an L-fold gain in the maximum achievable DoF compared to the single-antenna case. For the sake of comparison, we recall that in the broadcast channel, adding more antennas at the base station leads to a gain that is only additive.

In fact, for the MISO broadcast channel the order-optimal degree of freedom is $L + K\gamma$, where L is the number of antennas at the base station [17]. For the XOR based scheme to work, it is enough that each receiving user has local CSI knowledge, i.e. knowledge of the channel between the transmitting user and itself. However, the required subpacketization S_1 is huge and in the finite file size regime it affects the effective achievable $\overline{DoF}$. In fact, under the subpacketization constraint S_{max}, which limits the maximum number of users over which it is possible to encode to the value $\bar{K} < K$, the DoF reduces to

$$\overline{DoF}(\gamma) = L\bar{K}\gamma. \tag{2.57}$$

The above DoF can be much smaller than the theoretical one depending on how severe is the subpacketization constraint.

It turns out that by letting users cooperate the effective DoF under the constraint S_{max} can be boosted by a multiplicative factor at the expenses of global channel state knowledge. We give in the following theorem the main result of this chapter which is obtained by using the placement and delivery scheme described in section 2.3 and 2.5, respectively.

Theorem 2. *For a D2D coded caching problem where the finite size of the files stored at the library constraints the files to be split in no more than S_{max} subfiles, the following effective maximum degree of freedom is achievable:*

$$\overline{DoF}(\gamma) = LL_c\bar{K}\gamma \tag{2.58}$$

where $L_c \in [\frac{K}{2}]$ is the cooperation factor and it is a parameter of choice[6].

The above DoF is achieved thanks to the fact that the achievable scheme, the cooperation based scheme described in section 2.5, requires the subpacketization to be

$$S_{L_c} = L\frac{K}{L_c}\gamma\binom{\frac{K}{L_c}}{\frac{K}{L_c}\gamma} \tag{2.59}$$

which, as already discussed, increases the maximum number of users over which it is possible to encode to $\bar{K}_{L_c} = L_c\bar{K}$.

It is of relevant interest to evaluate the ratio between the subpacketization required by the cooperation based scheme S_{L_c} and the one S_1 required by the XOR based algorithm. This will give us an idea of how much bigger the subfiles will be if the new scheme is used. In fact, from a practical point of view, we recall that the smaller is the size of the subfiles, the higher will be the loss due to the headers of the transmitted packets.

[6] *We assume here that $L_c\bar{K} \leq K$.*

Theorem 3. *The subpacketization required by the novel cooperation-based scheme is exponentially lower with the number of users K than the one required by the legacy scheme, at the cost of increased CSI knowledge. The ratio $\frac{S_{L_c}}{S_1}$ takes the form*

$$\frac{S_{L_c}}{S_1} \approx \frac{\gamma^{\gamma \frac{L_c-1}{L_c} K}}{L_c} \tag{2.60}$$

where the approximation is obtained using $\binom{n}{k} \approx (\frac{n}{k})^k$.

The cooperation based algorithm achieving the effective DoF given in Theorem 2 requires each user to have global CSI knowledge. In fact, fixing in the algorithm the set Q and assuming that group $\mathcal{G}_i$ transmits to the other $K'\gamma$ groups $\mathcal{G}_j,\ \forall j \in Q\backslash\{i\}$, then the users in group $\mathcal{G}_i$ has to know the channel matrices between their group and all the others. A receiving group $\mathcal{G}_r,\ r \in Q\backslash\{i\}$ has to know the channel matrices between the transmitting group $\mathcal{G}_i$ and the other groups $\mathcal{G}_j, j \in Q\backslash\{i, r\}$. However, this could be quite difficult to obtain especially when the channel state changes fast. Thus, we developed a low subpacketization scheme which does not require global CSI knowledge (see section 2.6). This lead us to the next theorem.

Theorem 4. *In the multi-antennas D2D coded caching framework, where a low subpacketization scheme is required and no global CSI is available to the users, the following effective DoF is achievable:*

$$\overline{DoF}(\gamma) = L(\bar{K}\gamma + 1) \tag{2.61}$$

with subpacketization $S_{L_c} = L\frac{K}{L_c}\binom{\frac{K}{L_c}}{\frac{K}{L_c}\gamma}$ and where $\bar{K}$ is the maximum number of users over which it is possible to encode given by the subpacketization constraint in equation (2.8).

The effective degree of freedom achieved with the scheme described in section 2.6 is higher than the one achieved by the XOR based scheme of only an additive factor of L. In fact, we recall that the XOR based scheme achieves $\overline{DoF} = L\bar{K}\gamma$. For the regime of interest, i.e. when $K\gamma$ is high, this extra DoF of L might be negligible. Hence, given the complexity of the low subpakcetization scheme, the XOR based scheme would be preferred when $K\gamma$ is high.

All the above discussion leads us to think that there is a strong fundamental connection between achievable DoF, subpacketization and CSI knowledge.

2.9 Conclusions

In this chapter we have studied the multiple-antennas D2D coded caching problem under the subpacketization constraint. To the best of my knowledge, this is the first time that this problem is investigated. Research has been done for the subpacketization problem for the broadcast channel but not for the D2D setting. Moreover, it is also the first time that multiple antennas at users are considered in the coded caching framework. The enormous gains promised by coded caching are theoretically achievable for infinite file size. In the finite file size regime, the maximum number of subfiles into which we can split the files drastically limits the gains promised by coded caching to much lower values. Given the recent results of Lampiris and Elia [18] for the MISO broadcast channel with cache-enabled users, the objective in this chapter was to explore multiple antennas and users' cooperation to boost the performance under the subpacketization constraint. To this end, we have built a generic cache placement algorithm and 3 different delivery schemes. As a results of the aforementioned schemes, it turns out that having L antennas at each user can boost the theoretical sum degree of freedom of the system by a multiplicative factor of L. However, we have not found any interesting relation between number of antennas at the user terminals and subpacketization. In better words, we have not found any way to exploit the multiple antennas at users to reduce the subpacketization. On the other hand, we can conclude that users cooperation plays a crucial role in reducing the subpacketization and hence in increasing the effective achievable degree of freedom. The gain arising from users cooperation is directly proportional to the number of users which cooperate together (L_c) and there exist a cooperation factor which allows to achieve the theoretical gain promised by coded caching even in the finite file size regime. However, users cooperation strongly rely on a perfect knowledge of all the channels between each pair of users involved in a multicast session. In fact, in a cooperation based algorithm users are grouped in groups of L_c users in order that each group can act as a distributed transmitter and/or receiver. This cooperation is possible only thanks to global CSI knowledge. Finally, this work suggests a fundamental relation between effective achievable degree of freedom, channel state information knowledge and subpacketization.

Chapter 3

D2D Coded Multicasting for Distributed Computing

3.1 Introduction

Emerging applications in fields such us computer vision, augmented reality and recommendation systems are becoming more and more computationally expensive and data-hungry. Nowadays, it is already common to run machine learning and data analysis algorithms in very large distributed systems comprising of relatively small and unreliable computing nodes. This is very common in the Cloud where large clusters of servers are used to execute data-intensive jobs. It is common to refer to a set of nodes coordinating and exchanging messages to execute a given task as distributed computing system. The use of a set of low computationally powerful nodes in place of a single super powerful computing node is motivated by economical reasons (cheaper), feasibility (the aggregated computational power of a big set of nodes may be much higher than the most powerful supercomputer we might be able to build) and flexibility in terms of allocation of resources. Generally speaking, distributed computing exploits the presence of more than one available computing node, in order to allow for faster execution of a computational task.

In this age, distributed computing is only used in clusters of servers interconnected each other via a wired network of switches and routers. However, recent progress in radio technology have provided flexibility and enhanced capabilities in terms of power efficiency and guaranteed rates so that *wireless distributed computing* (WDC) has been envisioned to be feasible. The goal of wireless distributed computing is to reduce per-node and network resource requirements and enable computational intensive applications at wireless nodes not otherwise possible. The main enabling technologies for WDC are software defined radio, cognitive radio and fault tolerant distributed computing algorithms.

The general distributed computing problem considers the job of processing a large dataset and aims to generate Q output results in a distributed fashion across a set of K computing nodes, each being in charge of generating one or more of the Q output results.

The execution of a job in a cluster of computing nodes requires the job to be modelled so that it can be executed in parallel in the available nodes. This effort usually involves dividing the original computational task into different subtasks, and then assigning these subtasks to different nodes which, after some intermediate steps, compute the final task in parallel. While some rare tasks are by nature already parallel and so they can be immediately executed in parallel in a distributed system, most computational problems need to be parallelized, and this usually involves an intermediate preprocessing step and a subsequent information exchange between the nodes. The most well-known distributed models are Dryad [19], CIEL [20] and MapReduce [21]. Our focus will be on the MapReduce model, which is a parallel processing tool that simplifies the parallel execution of tasks, by abstracting the original problem into the following four phases:

1. the *assignment phase*, where a central controller node distribute the dataset, with a given redundant factor for fault tolerance, to many computing nodes. In this phase, the controller decides also which of the Q output results each node will be responsible to compute at the end of the process.

2. the *mapping phase*, where the nodes perform an intermediate computation aiming to "prepare" for parallelization.

3. the *shuffling phase* (or communication phase), where nodes communicate between each other the preprocessed data that is needed to make the process parallel.

4. the *reduce phase*, where nodes work in parallel to provide the final output that each is responsible for.

Some type of tasks that can be parallelized under a MapReduce framework include Sorting [22], Data Analysis [23], Word Counting, and Genome Sequencing [25].

3.1.1 Communication Bottleneck of Distributed Computing

While though MapReduce allows for a simple parallelization of the jobs, it comes with a fundamental bottleneck: the communication bottleneck. The latter bounds the performance of MapReduce, especially as the dataset size becomes larger and larger. Specifically, while having more nodes can speed up computational 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. For example, $50\% - 60\%$ of the overall execution time for some basic tasks including the aforementioned terasort and word counting is spent in the shuffling phase [24].

Delays of each of the MapReduce Phases In particular, consider a setting where there are K computing nodes, operating on a dataset of F elements. The time spent in the assignment phase is usually note considered as part of the overall execution time of a job. In fact, it is also common that the dataset needed for a certain job is already distributed among the nodes according to the underlying file system. In terms of the map phase, let $T_{\text{map}}(F)$ represent the time required for one node to map the entire dataset. Assuming that each node has to map a fraction $\gamma \geq \frac{1}{K}$ of the dataset (which implies that each element of the dataset appear in $t = K\gamma$ different computing nodes), then the map phase has a duration approximately $T_{\text{map}}(\gamma F) = T_{\text{map}}(t\frac{F}{K})$ which generally reduces with K. The reason why the dataset may be replicated more than once is due to the fact that there might be some failures, due to unavailability of resources. Hence, the replication of the dataset is used for fault tolerance at the expenses of storage. Similarly, the final reduce phase enjoys the same decreased delay $T_{\text{red}}(F/K)$, where $T_{\text{red}}(F)$ denotes the time required for a single node to reduce the entire mapped dataset[1].

The problem lies with the communication delay $T_{\text{com}}(F)$. Let T_c denote the time required to transmit the entire mapped dataset, from one node to another without any interference from the other nodes[2]. Since each node already has a fraction γ of the dataset, the delay of the shuffling phase takes the form $T_{\text{com}}(F) = T_c \cdot (1-\gamma)$, which does not decrease with K.

Hence for the basic MapReduce (MR) algorithm — under the traditional assumption that the three phases are performed sequentially — the overall execution time becomes

$$T_{\text{tot}}^{\text{MR}}(F, K) = T_{\text{map}}(t\frac{F}{K}) + T_c \cdot (1 - \frac{t}{K}) + T_{\text{red}}(\frac{F}{K}) \tag{3.1}$$

which shows that, while the joint computational cost $T_{\text{map}}(\gamma F) + T_{\text{red}}(\frac{F}{K})$ of the map and reduce phases can decrease by adding more nodes, the communication time $T_c \cdot (1-\gamma)$ is not reduced and actually increases with the number of nodes. Thus the cost of the shuffling phase emerges as the actual major bottleneck of the entire process.

It is also important to point out that, since T_c accounts for the capacity of the communication link between the nodes, the time spent in the shuffling phase is a critical point of failure which is even more severe in wireless distributed computing.

In order to tackle this problem and make distributed computing scaling with the number of nodes, several optimization methods, acting on different aspects of MapReduce, have been proposed to reduce the communication time. Among others, a method employing distributed cache memories has been proposed in [27], an optimal flow scheduling across network paths has been proposed in [26] and a method combining intermediate results before shuffling can be found in [21]. A new line of research was opened up by Li et al. in [28] wherein the coded multicasting technique used in single-antenna D2D coded caching (see section 1.3) has been proposed to reduce the communication load of distributed computing.

[1]We here assume for simplicity of exposition, uniformity in the amount of mapped data that each node uses in the final reduce phase. We also assume a uniformity in the computational capabilities of each node.

[2]T_c essentially accounts for the ratio between the capacity of the communication link and the dataset size F.

3.1.2 Emergence of Coded MapReduce: exploiting structured redundancy to reduce the communication load

Coded MapReduce [28] is a communication-efficient variation of MapReduce, which modifies the mapping phase, in order to allow for the shuffling phase to employ coded communication. The main idea of the method is to assign and then force each node to map a fraction $\gamma > 1/K$ of the whole dataset (such that each element of the dataset is mapped in $t \triangleq K\gamma$ 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 in the shuffling phase. The packets to be transmitted are not sent one after the other, but are rather combined together into XORs and sent as one. The reason this speedup would work is because the recipients of these packets could use part of their (redundant) mapped packets in order to remove the interfering packets from the received XOR, and acquire their own requested packet. This allowed for communicating (during the shuffling phase) to $t = K\gamma$ nodes at a time, thus reducing the shuffling phase duration, from $T_c \cdot (1-\gamma)$ to $\frac{1}{t}T_c \cdot (1-\gamma) = \frac{1}{K\gamma}T_c \cdot (1-\gamma)$. Hence, the overall execution time given by Coded MapReduce is

$$T_{\text{tot}}^{\text{CMR}}(F, K) = T_{\text{map}}(\gamma F) + T_c \cdot \frac{1-\gamma}{K\gamma} + T_{\text{red}}(\frac{F}{K}) \tag{3.2}$$

Looking at the above equation, it is evident that there is a fundamental trade-off between communication and computation. In fact, equation (3.2) shows that it is possible to reduce the communication load by a multiplicative factor $K\gamma$ at the expense of some extra computation (γ times more) in the map phase. The optimal value of γ required to minimize the overall execution time depends on the communication link capacity and the job itself. This trade-off has been proven to be information theoretical optimal in [29]. Moreover, we notice that if we fix γ the shuffling time and the reduce time decrease with the number of nodes. This behaviour is not experienced by the conventional MapReduce.

The technique we are talking about (Coded MapReduce) is nothing but the same technique used in the coded caching framework for the D2D setting which we have already described in section 1.3. Thus, there exist a one-to-one connection between the communication phase of MapReduce and D2D coded caching. Since Coded MapReduce uses exactly the same communication technique of D2D coded caching, we will omit here its detailed description. However, in the next section we try to make a clear connection between these two only apparently different worlds.

On the Connection Between Coded MapReduce and D2D Coded Caching

The general distributed computing problem considers a job of processing a large dataset to generate Q outputs. The job will be carried out distributively across K computing nodes. For simplicity and to make the parallelism more clear we assume $Q = K$ and thus each of the K nodes will be in charge of computing one of the Q outputs. In the more general case, each node has to compute $\frac{Q}{K}$ outputs. Under the above assumption, this corresponds to say that in the caching problem each of the K users requests a different file from the library. Thus, as in distributed computing a node needs all the mapped data necessary to compute a given output, in the same way a coded caching user requests a certain file.

In the assignment phase of Coded MapReduce, the dataset is split into S subsets, each of which is assigned to a given subset of nodes. Analogously, in a D2D network with K users, in the placement phase each file is split into S subfiles, each of which is cached in the memory of a given subset of users. However, while in distributed computing there is only a single input file (the dataset), in caching there are many of them. In Coded MapReduce, the way the subsets of the dataset are assigned to the computing nodes is exactly the same as the way subfiles are stored in the users' caches.

In the mapping phase, each computing node processes each of the assigned subsets to generate for each of them Q intermediate results. This phase is not present in caching where the stored subfiles at each user are not processed. Despite some computation is performed on each subset, the structured redundancy introduced in the assignment phase is preserved and multicasting opportunities arise. These multicasting opportunities are the same as the ones that arise in the equivalent caching problem thanks to the cache placement. At this point, the communication phase starts using the coding scheme presented in Coded MapReduce, which is actually equivalent to the scheme presented in [7]. Finally, after the communication phase, the reduce phase starts and all nodes compute in parallel the output results they are in charge of. This last phase is not present in D2D coded caching which is only a communication technique. We here provide a toy example to better understand how Coded MapReduce works and its connection with coded caching.

Example Consider $K = 3$ servers that are assigned a dataset of F numbers in the range $[1-3000]$ to perform a sorting algorithm on. In order to make the sorting algorithm parallel we make server 1 in charge of sorting all numbers in the range $C_1 = [1 - 1000]$, server 2 will sort all the numbers in the range $C_2 = [1001 - 2000]$ and server 3 will be in charge of range $C_3 = [2001 - 3000]$. In the map phase each server puts the numbers that it has received during the assignment phase in one of the three ranges C_1, C_2, C_3.

Assuming that we want to reach a gain of $t = 2$, in the assignment phase the dataset is divided into $t\binom{K}{t} = 6$ non-overlapping subsets

$$W_1, W_2, W_3, W_4, W_5, W_6$$

each of size $\frac{F}{6}$ elements. After that, server 1 is assigned subsets W_1, W_2, W_3, W_4, server 2 subsets W_3, W_4, W_5, W_6 and server 3 is assigned subsets W_1, W_2, W_5, W_6. Then, each server maps each of its own subsets into the 3 ranges. Thus, for example the number $2145 \in W_i$ would be placed in the range $C_3 = [2001 - 3000]$. After the server has placed all the numbers of the subset W_i in the correct range, the subset W_i remains divided in three sets of numbers

$$W_i^{C_1}, W_i^{C_2}, W_i^{C_3}$$

where $W_i^{C_1}$ is the set of numbers from W_i which are in the range C_1, $W_i^{C_2}$ is the set of numbers from W_i which are in the range C_2 and $W_i^{C_3}$ is the set of numbers in the range C_3.

For example, after the mapping phase, server 1 will have the following mapped data

$$W_1^{C_1}, W_1^{C_2}, W_1^{C_3}, W_2^{C_1}, W_2^{C_2}, W_2^{C_3}, W_3^{C_1}, W_3^{C_2}, W_3^{C_3}, W_4^{C_1}, W_4^{C_2}, W_4^{C_3}.$$

We can now notice that, since server 1 is in charge of range C_1, it needs to get the mapped data $W_i^{C_1}, i \in \{1,2,3,4,5,6\}$ so that it can sort, in the reduce phase, all the numbers in the range $[1-1000]$. Thus, it needs $W_5^{C_1}$ and $W_6^{C_1}$ which have been computed by servers 2 and 3. A similar discussion can be done for server 2 and 3.

In the shuffling phase, the 3 servers transmit respectively[3]

$$\textit{server } 1 \rightarrow x_1 = W_1^{C_2} \oplus W_3^{C_3} \tag{3.3}$$

$$\textit{server } 2 \rightarrow x_2 = W_4^{C_3} \oplus W_5^{C_1} \tag{3.4}$$

$$\textit{server } 3 \rightarrow x_3 = W_2^{C_2} \oplus W_6^{C_1} \tag{3.5}$$

Each receiving server, thanks to the mapped data that it already has, can successfully decode its desired message by subtracting the interfering one that he has in its memory. At the end of the shuffling phase, all the servers will have all the numbers falling in the range they are in charge of and thus they can proceed, in parallel, to sort them in order to get the final sorted dataset.

3.1.3 Identifying the Subpacketization Bottleneck of Coded 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 subpackets which grows exponentially in K. This problem is intrinsic in these coded multicasting techniques and it has been already highlighted and investigated in the first chapter of this work in the context of coded caching. However, we restate and analyse the problem in the context of distributed computing.

Specifically the fact that the finite-sized dataset can only be divided into a finite number of subpackets, limits the values of parameter t that can be achieved, because the corresponding subpacketization, which need be as high as

$$S = t\binom{K}{t}$$

[3]Here it is assumed that all $W_i^{C_j}$ are equally-sized. However, if this is not the case, zero-padding is applied.

must be kept below some maximum allowable subpacketization $S_{\max}$, which, also, must be much less than the number of elements F in the dataset[4]. If this number $S = t\binom{K}{t}$ exceeds the maximum allowable subpacketization $S_{\max}$, then coded communication is limited to include coding that spans only

$$\bar{K} = \arg\max_{K} \left\{ t\binom{K}{t} \leq S_{\max} \right\} \tag{3.6}$$

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 the theoretical communication gain from coding.

This high subpacketization requirement of the state-of-art Coded MapReduce algorithm, creates the following three problems:

- *limited communication gain.* As already highlighted right above, under the subpacketization constraint S_{max}, the maximum gain is limited to $\bar{t} = \bar{K}\gamma$ since it is possible to encoded only over a maximum number of users $\bar{K}$.
  Moreover, Coded MapReduce assumes that the communication link between the computing node is a shared channel. In fact, it is only when the link connecting the nodes has the multicast property that all the gains promised by coding can be fully achieved. In practice, in a wired setting such us a cluster of nodes in a data center, the computing nodes are connected via a network of switches with a given physical topology (tree topology for example). As a consequence, the cost of multicasting a packet is higher than the cost of unicasting one. In [31] the authors measured that the multicasting time increases logarithmically with the number of nodes in a multicast group, which in Coded MapReduce corresponds to t (t nodes at a time are served in the shuffling phase). Moreover, since the medium is not multicast by itself, a multicast topology (usually a tree topology) has to be built at network layer every time that a multicast message has to be transmitted. This comes with a non negligible cost. Keeping in mind the above discussion, we recall that the higher is the subpacketization the higher is the number of transmissions $(t+1)\binom{K}{t+1}$ required to conclude the shuffling phase. Hence, the more are the transmissions the more are the multicast sessions that has to be created at network layer, each of which session will be used only for a small multicast message. Thus, the higher is the subpacketization the higher is the loss due to the non-broadcast property of the channel. In [30] the authors have implemented Coded MapReduce for a sorting problem which, despite outperforms the conventional MapReduce, achieves gains that are lower than the theoretical ones because of the aforementioned problems. The above discussed problem is not present in wireless distributed computing where the medium is by itself broadcast.

[4]Recall that in D2D coded caching we have the same subpakcetization $S = K\gamma\binom{K}{K\gamma}$

- *high packet overheads.* The second problem caused by the huge subpacketization relates to the fact that as the number of subsets which the dataset is split into increases, the mapped subsets themselves become smaller and smaller. This means that the header that must accompany each XOR transmission of mapped subsets will occupy a significant portion of the transmitted signal. Simply put, the higher the subpacketization, the more the communication load is dominated by header overheads.

- *inefficient XOR creation due to unevenness in the mapped data.* The third problem from high subpacketization is that the resulting small sized subsets and consequently mapped subsets, cause high variations between the sizes of the mapped subsets. This unevenness — which is naturally much more accentuated in smaller packets — can cause substantial additional delays because it forces zero padding (we can only coombine equal-sized bit streams) which wastes communication resources.

This can be better understood by using the sorting example presented before. Instead of assuming that each mapped subset (intermediate value) has equal amount of elements, i.e., instead of assuming that $|W_i^{C_1}| = |W_i^{C_2}| = |W_i^{C_3}| = \frac{1}{3}|W_i| = \frac{F}{18}, i = 1,2,...,6$, (recall that each of the 6 subfiles has size $|W_i| = F/6$) we will assume the following:

- any intermediate value $W_1^{C_3}, W_2^{C_3}, W_3^{C_3}, W_4^{C_3}, W_5^{C_3}, W_6^{C_3}$ with upper index 3 will each occupy a fraction 1/2 of the elements of the respective subset, i.e. $|W_i^{C_3}| = 1/2|W_i| = F/12, i = 1,2,...,6$;
- intermediate values with upper index 1 or 2 $W_1^{C_1}, W_2^{C_1}, W_3^{C_1}, W_4^{C_1}, W_5^{C_1}, W_6^{C_1}$ and $W_1^{C_2}, W_2^{C_2}, W_3^{C_2}, W_4^{C_2}, W_5^{C_2}, W_6^{C_2}$, will only have 1/4 of the elements of their respective subsets, i.e. $|W_i^{C_1}| = 1/4|W_i| = F/24$ and $|W_i^{C_2}| = 1/4|W_i| = F/24,\ i = 1,2,...,6$.

In the case of uncoded MapReduce, the corresponding delay would remain $(1-\gamma)T_c = (1-2/3)T_c = 1/3T_c$ because there are no XORs, and because despite the unevenness, the total amount of information that must be communicated, remains the same. On the other hand, in the case of coded communication, having $|W_i^{C_1}| = |W_i^{C_2}| = F/24 \neq |W_i^{C_3}| = F/12$, in turn means that for every aforementioned XOR that includes some of the $W_i^{C_3}$ elements inside, we would have to perform zero padding. For example, in the case of $W_4^{C_3} \oplus W_5^{C_1}$, we would have to zero pad $W_5^{C_1}$ to double its size, thus wasting resources. Now the three introduced XORs will have sizes $|x_1| = |x_2| = F/12, |x_3| = F/24$, and thus sending all three would require a total delay of $T_c/12 + T_c/12 + T_c/24 = 5T_c/24$.

Comparing the above to the delay $\frac{1}{3}T_c$ of the uncoded case, we can see that the multiplicative gain in the communication phase due to coded communication is limited to Gain $= (1/3)/(5/24) = 8/5 = 1.6$, instead of the theoretical gain of $t = 2$.

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. As a result, our proposed scheme will eventually allow substantial reductions in the overall execution time for a large class of distributed computing algorithms. This new method is actually the same as the one proposed for the single-antenna D2D coded caching problem in section 2.5. However, for the sake of completeness and in order to better show how to use coded caching techniques in the context of distributed computing, we will present in details the scheme in the next section.

Before describing our solution and its performance, let us first elaborate on the exact channel model.

3.1.4 Channel Model: Distributed Computing in a D2D Setting

In terms of the communication medium, we will focus on the wireless fully-connected setting where the nature of multicasting and the impact of link bottlenecks are clearer. As we will discuss later on though, the ideas here apply directly to the wired case as well.

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 $\mathbf{x} \in C^{L\times 1}$, then the received signal at a receiving node k takes the form

$$y_k = \mathbf{h}_k^T \mathbf{x} + w_k, \quad k = 1, \cdots, K \tag{3.7}$$

where as always $\mathbf{x}$ satisfies a power constraint $E(||\mathbf{x}||^2) < P$, where $\mathbf{h}_k \in C^{L\times 1}$ 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 as we will see later for the wired case, perfect network coding coefficients) at the active receivers and transmitters.

3.2 Description of the Proposed Cooperation Based Scheme

In this section we proceed to describe in details the proposed algorithm, which we will refer to as the Group-based Coded MapReduce (GCMR) algorithm.

We consider a dataset Φ consisting of F elements, and a computational task that asks for $Q \geq K$ output values $u_q = \phi_q(\Phi)$, $q = 1, \cdots, Q$. The general aim is to distribute this task across the K nodes, hence the dataset is split into S disjoint parts W_s, $s = 1, \cdots, S$ ($\cup_{s=1}^{S} W_s = \Phi$). We recall that, as is common in MapReduce, each function ϕ_q is decomposable as

$$\phi_q(\Phi) = r_q(m_q(W_1), \cdots, m_q(W_S)) \tag{3.8}$$

where the *map functions* $\{m_q,\ q \in [Q]\}$ map file W_s into Q intermediate values $W_s^q = m_q(W_s),\ q \in [Q]$, which are used by the reduce function r_q to calculate the desired output value $u_q = r_q(W_1^q, \cdots, W_S^q)$.

We now give the description of the *assignment*, *mapping, shuffling* and *reduce* phases.

3.2.1 Dataset Assignment Phase

We split the K nodes in $[K]$, into $K' \triangleq \frac{K}{L}$ groups

$$\mathcal{G}_i = \{i, i+K', ..., i+(L-1)K'\}, \quad i \in [K'] \tag{3.9}$$

of L nodes per group, and we split the dataset into

$$S = K'\gamma \binom{K'}{K'\gamma} \tag{3.10}$$

parts, 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. At this point, each $s = 1,2, \cdots, S$ is associated to a unique double index τ, σ so that the dataset can be seen as being segmented in $\{W_{\tau,\sigma},\ \tau \subseteq [K'] : |\tau| = K'\gamma,\ \sigma \in \tau\}$. Each node in group $\mathcal{G}_i$ is then assigned the set of subsets

$$\mathcal{M}_{\mathcal{G}_i} = \{W_{\tau,\sigma} : \tau \ni i, \forall \sigma \in \tau\} \tag{3.11}$$

and each of the Q reduce functions r_q is assigned to a given node. For simplicity we assume that $Q = K$[5].

3.2.2 Map Phase

This phase consists of each node k computing the map functions m_q of all files in $\mathcal{M}_{\mathcal{G}_i}, \mathcal{G}_i \ni k$ for all $q \in [Q]$. At the end of the phase, node $k \in \mathcal{G}_i$ has computed the intermediate results $W_{\tau,\sigma}^q = m^q(W_{\tau,\sigma})$ for all $W_{\tau,\sigma} \in \mathcal{M}_{\mathcal{G}_i}$.

3.2.3 Shuffle Phase

Each node $\mathcal{G}_i(j)$ (j-th node of group $\mathcal{G}_i$) must retrieve from the other nodes (except from those in $\mathcal{G}_i$), the intermediate values $\{W_{\tau,\sigma}^{\mathcal{G}_i(j)} : W_{\tau,\sigma} \notin \mathcal{M}_{\mathcal{G}_i}\}$ that it has not computed locally. Each node $\mathcal{G}_i(j)$ will thus create a set of symbols $\{x_{\mathcal{G}_i(j),\mathcal{P}\setminus\{i\}}\}$, intended for all the nodes in groups $\mathcal{G}_j, j \in \mathcal{P} \setminus \{i\}$ for some $\mathcal{P} \subset [K']$ of size $|\mathcal{P}| = K'\gamma + 1$, where of course each symbol $x_{\mathcal{G}_i(j),\mathcal{P}\setminus\{i\}}$ is a function of the intermediate values computed in the map phase. We use

$$\mathbf{x}_{i,\mathcal{P}\setminus\{i\}} \triangleq [x_{1,\mathcal{Q}\setminus\{i\}}, \cdots, x_{|\mathcal{M}_{\mathcal{G}_i}|,\mathcal{P}\setminus\{i\}}]^T$$

[5]When Q is an integer multiple of the number of nodes, the coded multicasting technique used in the shuffling phase is repeated $\frac{Q}{K}$ times.

to denote the vector of symbols that are jointly created by the nodes in $\mathcal{G}_i$ and which are intended for the nodes in $\mathcal{G}_j, j \in \mathcal{P} \setminus \{i\}$. Each symbol is communicated (multicasted) by the corresponding node $\mathcal{G}_i(j)$ to all the other nodes. We proceed to provide the details for transmission and decoding.

Transmission For each subset $\mathcal{P} \subset [K']$ of size $|\mathcal{P}| = K'\gamma + 1$, we sequentially pick all its elements $i \in \mathcal{P}$ so that the nodes in group $\mathcal{G}_i$ act as a single distributed transmitter. These users in $\mathcal{G}_i$ construct the following vector of symbols

$$\mathbf{x}_{i,\mathcal{P}\setminus\{i\}} = \sum_{k' \in \mathcal{P}\setminus\{i\}} \mathbf{H}_{i,k'}^{-1} \left[W_{\mathcal{P}\setminus\{k'\},i}^{\mathcal{G}_{k'}(1)}, \cdots, W_{\mathcal{P}\setminus\{k'\},i}^{\mathcal{G}_{k'}(L)} \right]^T \tag{3.12}$$

where $\mathbf{H}_{i,k'}^{-1}$ is the ZF precoding matrix for the channel $\mathbf{H}_{i,k'} \in C^{L\times L}$ between transmitting group $\mathcal{G}_i$ and receiving group $\mathcal{G}_{k'}$, and where $\{W_{\mathcal{P}\setminus\{k'\},i}^{\mathcal{G}_{k'}(j)}\}_{j=1}^{L}$ is a set of intermediate results desired by the nodes in $\mathcal{G}_{k'}$. Each user $\mathcal{G}_i(j)$ now transmits the j-th element of the constructed vector $\mathbf{x}_{i,\mathcal{P}\setminus\{i\}}$.

Decoding Node $\mathcal{G}_p(j), p \in \mathcal{P} \setminus \{i\}$ receives the signal

$$y_{\mathcal{G}_p(j)} = \mathbf{h}_{\mathcal{G}_p(j)}^T \mathbf{x}_{i,\mathcal{P}\setminus\{i\}} + w_{\mathcal{G}_p(j)} \tag{3.13}$$

and removes out-of-group interference by employing the intermediate values it has computed locally in the map phase. Specifically each node $\mathcal{G}_p(j)$, and all the nodes in $\mathcal{G}_p, p \in \mathcal{P} \setminus \{i\}$, remove from their $y_{\mathcal{G}_p(j)}$ the signal

$$\mathbf{h}_{\mathcal{G}_p(j)}^T \sum_{k' \in \mathcal{P}\setminus\{i,p\}} \mathbf{H}_{i,k'}^{-1} \left[W_{\mathcal{P}\setminus\{k'\},i}^{\mathcal{G}_{k'}(1)}, \cdots, W_{\mathcal{P}\setminus\{k'\},i}^{\mathcal{G}_{k'}(L)} \right]^T \tag{3.14}$$

to stay with a residual signal

$$\mathbf{h}_{\mathcal{G}_p(j)}^T \mathbf{H}_{\mathcal{G}_i \mathcal{G}_p}^{-1} \left[W_{\mathcal{P}\setminus\{p\},i}^{\mathcal{G}_p(1)}, \cdots, W_{\mathcal{P}\setminus\{p\},i}^{\mathcal{G}_p(L)} \right]^T + w_{\mathcal{G}_p(j)}. \tag{3.15}$$

By choosing $\mathbf{H}_{\mathcal{G}_i \mathcal{G}_p}^{-1}$ to be a ZF precoder, removes intra-group interference, thus allowing each node $\mathcal{G}_p(j)$ to receive its desired intermediate value $W_{\mathcal{P}\setminus\{p\},i}^{\mathcal{G}_p(j)}$. The shuffling phase is concluded by going over all the aforementioned sets $\mathcal{P} \subset [K']$.

3.2.4 Reduce Phase

At this point, each node uses the symbols received during the shuffling phase, together with the intermediate mapped results computed locally, in order to construct the inputs $W_1^q, ..., W_S^q$ that are required by the reduce function r_q to calculate the desired output value $u_q = r_q(W_1^q, \cdots, W_S^q)$.

3.2.5 Extension to the Wired Setting

As a last step, we quickly note that the same vector precoding used to separate the users of the same group 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. 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. Figure 3.1 illustrates a wired network of 5 computing nodes connected each other via 2 switches. The switches can perform network coding operations on the incoming signals so that the ”channel” between two sets of nodes can be characterized by a matrix $\mathbf{H}$ of network coding coefficients. The interested reader is encouraged to read [17] to have a better idea of these kind of linear (wired) networks.

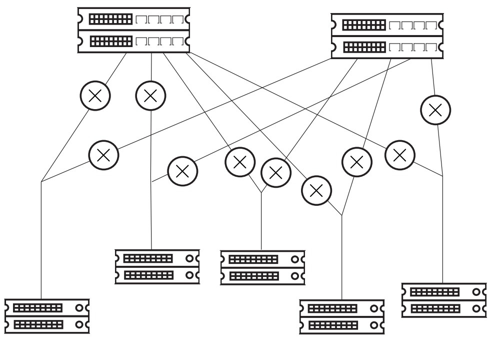

Figure 3.1. Illustration of the wired setting. $\times$ denotes a network coding operation.

3.2.6 Calculation of Shuffling Delay

We now evaluate the performance of the scheme which will be clearly the same as the one achieved by the analogous scheme for D2D coded caching. However, we evaluate it with a different approach. We first see that the subpacketization is equal to

$$S = K'\gamma\binom{K'}{K'\gamma} = \frac{K\gamma}{L}\binom{K/L}{K\gamma/L}. \tag{3.16}$$

Let us assume that $S_{max} \geq S$ in which case the subpacketization constraint is not a problem. In this case, the proposed Group-based Coded MapReduce achieves the same delay as Coded MapReduce, i.e. $T_{com}^{GCMR} = \frac{1-\gamma}{K\gamma}T_c$. To verify the term $K\gamma$, which corresponds to the DoF, we just need to note that during the shuffling phase no subfile is ever sent more than once, and then simply note that the scheme serves a total of $K'\gamma$ groups at a time, thus a total of $K'\gamma L = K\gamma$ nodes at a time. Finally, to justify the term $1 - \gamma$, we just need to recall that, due to the placement redundancy, a fraction γ of all the shuffled data is already at their intended destination.

Lastly when $S_{max} \leq S$, we simply have to recall that we are allowed to encode over $\bar{K}_L = \arg\max_K \{\frac{K\gamma}{L}\binom{K/L}{K\gamma/L} \leq S_{max}\}$ nodes at a time, which yields the desired $\bar{t}_L = \bar{K}_L\gamma$ which is L times higher than the one achieved by Coded MapReduce. Hence, under the subpacketization constraint, the duration of the shuffling delay, if the Group-based Coded MapReduce is used, is

$$T_{com}^{GCMR} = \frac{1-\gamma}{\bar{K}_L\gamma}T_c. \tag{3.17}$$

3.2.7 Example of the Scheme

We here present quickly an example of the scheme. The details will be skipped since we remind that transmission and decoding schemes of the shuffling phase are identical to the cooperation based scheme proposed for the single-antenna D2D coded caching whose a detailed example has been already presented in the second chapter.

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 $K/L = 4$ 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 12 parts as $\{W_{12,1}, W_{12,2}, W_{13,1}, W_{13,3}, W_{14,1}, W_{14,4}, W_{23,2}, W_{23,3}, W_{24,2}, W_{24,4}, W_{34,3}, W_{34,4}\}$, which are distributed to the nodes of group $\mathcal{G}_i$ as follows:

$$\begin{aligned}
\mathcal{M}_{\mathcal{G}_1} =& \{W_{12,1}, W_{12,2}, W_{13,1}, W_{13,3}, W_{14,1}, W_{14,4}\}\\
\mathcal{M}_{\mathcal{G}_2} =& \{W_{12,1}, W_{12,2}, W_{23,2}, W_{23,3}, W_{24,2}, W_{24,4}\}\\
\mathcal{M}_{\mathcal{G}_3} =& \{W_{13,1}, W_{13,3}, W_{23,2}, W_{23,3}, W_{34,3}, W_{34,4}\}\\
\mathcal{M}_{\mathcal{G}_4} =& \{W_{14,1}, W_{14,4}, W_{24,2}, W_{24,4}, W_{34,3}, W_{34,4}\}.
\end{aligned}$$

In the map phase, each file $W_{\tau,\sigma}$ is mapped into $\{W_{\tau,\sigma}^q\}_{q=1}^K$ such that, for example, $W_{\tau,\sigma}^1$ is the output of the first mapping function after acting on $W_{\tau,\sigma}$. Finally the transmissions take the form:

$$
\begin{aligned}
\mathbf{x}_{1,23} =& \mathbf{H}_{1,2}^{-1}\mathbf{W}_{13,1}^{\mathcal{G}_2} + \mathbf{H}_{1,3}^{-1}\mathbf{W}_{12,1}^{\mathcal{G}_3}\\
\mathbf{x}_{1,24} =& \mathbf{H}_{1,2}^{-1}\mathbf{W}_{14,1}^{\mathcal{G}_2} + \mathbf{H}_{1,4}^{-1}\mathbf{W}_{12,1}^{\mathcal{G}_4}\\
\mathbf{x}_{1,34} =& \mathbf{H}_{1,3}^{-1}\mathbf{W}_{14,1}^{\mathcal{G}_3} + \mathbf{H}_{1,4}^{-1}\mathbf{W}_{13,1}^{\mathcal{G}_4}\\
\mathbf{x}_{2,13} =& \mathbf{H}_{2,1}^{-1}\mathbf{W}_{23,2}^{\mathcal{G}_1} + \mathbf{H}_{2,3}^{-1}\mathbf{W}_{12,2}^{\mathcal{G}_3}\\
\mathbf{x}_{2,14} =& \mathbf{H}_{2,1}^{-1}\mathbf{W}_{24,2}^{\mathcal{G}_1} + \mathbf{H}_{2,4}^{-1}\mathbf{W}_{12,2}^{\mathcal{G}_4}\\
\mathbf{x}_{2,34} =& \mathbf{H}_{2,3}^{-1}\mathbf{W}_{24,2}^{\mathcal{G}_3} + \mathbf{H}_{2,4}^{-1}\mathbf{W}_{23,2}^{\mathcal{G}_4}\\
\mathbf{x}_{3,12} =& \mathbf{H}_{3,1}^{-1}\mathbf{W}_{13,3}^{\mathcal{G}_1} + \mathbf{H}_{3,2}^{-1}\mathbf{W}_{13,3}^{\mathcal{G}_2}\\
\mathbf{x}_{3,14} =& \mathbf{H}_{3,1}^{-1}\mathbf{W}_{34,3}^{\mathcal{G}_1} + \mathbf{H}_{3,4}^{-1}\mathbf{W}_{13,3}^{\mathcal{G}_4}\\
\mathbf{x}_{3,24} =& \mathbf{H}_{3,2}^{-1}\mathbf{W}_{34,3}^{\mathcal{G}_2} + \mathbf{H}_{3,4}^{-1}\mathbf{W}_{23,3}^{\mathcal{G}_4}\\
\mathbf{x}_{4,12} =& \mathbf{H}_{4,1}^{-1}\mathbf{W}_{24,4}^{\mathcal{G}_1} + \mathbf{H}_{4,2}^{-1}\mathbf{W}_{14,4}^{\mathcal{G}_2}\\
\mathbf{x}_{4,13} =& \mathbf{H}_{4,1}^{-1}\mathbf{W}_{34,4}^{\mathcal{G}_1} + \mathbf{H}_{4,3}^{-1}\mathbf{W}_{14,4}^{\mathcal{G}_3}\\
\mathbf{x}_{4,23} =& \mathbf{H}_{4,2}^{-1}\mathbf{W}_{34,4}^{\mathcal{G}_2} + \mathbf{H}_{4,3}^{-1}\mathbf{W}_{24,4}^{\mathcal{G}_3},
\end{aligned}
$$

where $\mathbf{W}_{i,\tau}^{\mathcal{G}_g}$ denotes a 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 results have been calculated by each node in $\mathcal{G}_2$ during the map phase. After noting that the precoding matrix $\mathbf{H}_{1,2}^{-1}$ removes intra-group interference, we can conclude that each transmission serves each of the 16 users with one of their desired intermediate results, which in turn implies a 16-fold speedup over the uncoded case.

3.3 Main Results

For the sake of comparing the proposed Group-based Coded MapReduce with Coded MapReduce, let us first recall that under the subpacketization constraint $S_{\max}$, the original Coded MapReduce approach achieves communication delay

$$T_{com}^{\mathrm{CMR}} = \frac{1-\gamma}{\bar{t}} T_c \tag{3.18}$$

where

$$\bar{t} = \gamma \cdot \arg\max_{K} \{K\gamma \binom{K}{K\gamma} \leq S_{\max}\}$$

is the maximum achievable effective speedup (due to coding) in the shuffling phase. Assuming for simplicity that $Q = K$ such that each node has one final task, we proceed with the main result.

Theorem 5. *In the K-node distributed computing setting where the dataset can only be split into at most S_{max} identically sized parts, the proposed Group-based Coded MapReduce algorithm with groups of L users, can achieve communication delay*

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

for

$$\bar{t}_L = \gamma \cdot \arg\max_{K} \{ \frac{K\gamma}{L} \binom{K/L}{K\gamma/L} \leq S_{\max} \}.$$

The proof follows directly from the description of the scheme and it is given in subsection 3.2.6.

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 1. *In the subpacketization-constrained regime where $S_{\max} \leq \frac{K\gamma}{L}\binom{K/L}{K\gamma/L}$, the new algorithm here allows for shuffling delay*

$$T_{com}^{GCMR} = \frac{1-\gamma}{\bar{t}_L} T_c = \frac{T_{com}^{CMR}}{L}$$

which is L times smaller than the delay without grouping.

Finally the following also holds.

Corollary 2. *When $S_{max} \geq \frac{K\gamma}{L}\binom{K/L}{K\gamma/L}$, the new algorithm allows for the unconstrained theoretical execution time*

$$T_{tot}^{GCMR} = T_{map}(\gamma F) + \frac{(1-\gamma)}{K\gamma} T_c + T_{red}\left(\frac{F}{K}\right). \tag{3.19}$$

3.4 Conclusions

In this chapter we highlighted the effects of the acute subpacketization required by the optimal Coded MapReduce, thus identifying the subpacketization as the main bottleneck of coded distributed computing. We proposed a way to quantify analytically the performance loss of CMR due to the subpacketization constraint and we provided a novel algorithm that employs node-grouping in the mapping and shuffling phases, to substantially reduce the shuffling-phase delays that had remained large due to the high subpacketization bottleneck of distributed computing. Using node cooperation one, for the first time, can almost infinitely reduce the execution of these types of algorithms as long as there are enough computing nodes, something that previously wasn't possible in uncoded methods and while in coded methods it would reach a performance ceiling due to subpacketization constraints.

Minimal overhead for group-based node cooperation It is interesting to note that the described node cooperation does not require any additional overhead communication of data (dataset entries) between the nodes. The only additional communication-overhead is that of having to exchange CSI between active receiving and transmitting nodes from $K\gamma/L+1$ groups. In static settings, where computing nodes are not moving fast, and in particular in wired settings where the network coding coefficients are fixed and known, the CSI overhead can be very small compared to the volumes of the communicated datasets.

3.4.1 Impact of Reduced Subpacketization on Coded Distributed Computing

The reduced subpacketization comes with a variety of positive ramifications.

Boost of the speedup factor t in the shuffling phase As we have discussed, the much reduced subpacketization allows for a substantial increase in the number of nodes we can encode over, thus potentially yielding an L-fold decrease in the shuffling-phase delay compared to Coded MapReduce.

Reduced packet overheads The second ramification from having fewer subpackets, comes in the form of reduced header overheads that accompany each transmission. Simply put, the fewer the subpackets, the bigger they are, hence the less the communication load is dominated by header overheads which can further slow down the shuffling phase.

Efficient Coded Message Creation by Reducing Unevennes Another positive ramification from our algorithm is that it can reduce the unevenness between the sizes of the mapped data that each subset is mapped into. We have already seen how unevenness can cause substantial additional delays because it forces zero padding which wastes communication resources. Having fewer and thus larger subpackets, averages out these size variations, thus reducing wasteful zero padding. By decreasing subpacketization, we automatically increase the size of these subpackets, thus decreasing with high probability, due to the law of large numbers, the relative unevenness, which in turn allows for higher speedup gains.

Chapter 4

Coded Caching for Networks with Helper Nodes

4.1 Introduction

In the first chapter we provided a brief overview of caching with a special focus on the emerging coded caching with its benefits and open challenging problems. After the first appearance of coded caching in the literature, many lines of research were opened in order to employ this novel multicasting technique in different settings such us D2D, combinatorial networks and hierarchical networks. Future wireless networks will be a combination of a dense deployment of small cells with small coverage and relatively high data rates along with macro cellular base stations with large coverage and smaller data rates. This emerging networks are commonly referred as heterogeneous wireless networks (HetNets) and can be clearly used to provide an architecture for wireless content distribution.

In [32] the authors study the coded caching problem based on an architecture where a library is stored at a main base station and the content of the library is cached at multiple helper nodes. Users, unlike the ones of the usual broadcast channel [5] and D2D setting [7], do not have an isolated cache in their terminal but they can get the content both from the helper nodes and the macro base station. More precisely, they consider the case when users can be at the same time associated to the single-antenna base station and a certain small number of helper nodes. The cost of fetching content from the helper nodes is assumed to be zero. This assumption is motivated by the fact that the data rates provided by small cells can be much higher than the ones provided by macro cells. Moreover, they assume that the same number of users are associated to each helper node and that the file popularity distribution is not uniform but there exists a fixed number of popularity levels, i.e. files are grouped such that all files in the same group have the same popularity. Under the above assumptions, the authors are interested in characterizing the trade-off between the normalized delivery time R required by the macro base station to serve all users and the cache memory M of each helper node for any possible combination of user requests. They propose an order-optimal caching and delivery strategy such that a broadcast message of rate at most R satisfies all said requests. However, they have a gap from optimality of 9909 which suggests that their proposed scheme is suboptimal and there might be still room for improvement in their setting.

In this paper, we study a similar caching problem to the one studied in [32]. We consider an heterogeneous networks with a main multi-antennas base station, Λ helper nodes and K users, each being exclusively connected to a single helper node. However, unlike the setting in [32], we allow different number of users to be associated to the same helper node. We will also assume that all the files of the library stored at the base station have the same probability of being requested. For this problem we will characterize the minimum worst-case delivery time R required by the base station to serve all users. In particular, we develop a lower bound for this delay and a matching algorithm which we will not describe in this document for confidentiality reasons[1]. We discuss the effect of the user profile, namely how the users are distributed among caches, on the performance of the system.

We now describe in details the system model and we formally define the problem of interest. Next, before presenting the main results of this chapter, we will provide some preliminary tool which will be essential to develop the lower bound. The latter and a small discussion about the scheme will be given at the end of this chapter, followed by a conclusion section.

4.2 System Model and Problem Definition

4.2.1 System Model

Consider a base station equipped with a library containing N files $W^1, ..., W^N$ of equal popularity with a sum file size constraint $\frac{1}{N}\sum_{n\in[N]}|W^i| = 1$ where 1 is one unit of file. The base station is connected to $K \in \mathbb{N}$ users through a broadcast link and to $\Lambda \in \mathbb{N}$ helper nodes (caches), where $N \geq K$.

The server is equipped with $N_0 \in \mathbb{N}$ antennas. Each helper node $\lambda \in [\Lambda]$ has a memory of size $M \in \mathbb{N}$ units of file. Each user $k \in [K]$ connects to only one of the helper nodes λ[2]. We denote the number of users connected to cache λ by L_λ where the sum of the users connected to each cache yields the total number of users as

$$\sum_{\lambda\in[\Lambda]} L_\lambda = K. \tag{4.1}$$

We define *user profile* the vector representing the number of users connected to each helper node and we denote such a vector as $\mathbf{L} = (L_1, ..., L_\Lambda)$. Such a system is depicted in Figure 4.1 and an example of a real network is drawn in Figure 4.2.

[1]The reader interested in the details of the achievable scheme is encouraged to contact directly the author of this thesis.

[2]The system model is given for a wireless setting. However, the results are also valid for wired linear networks where users, in groups, have shared caches [17].

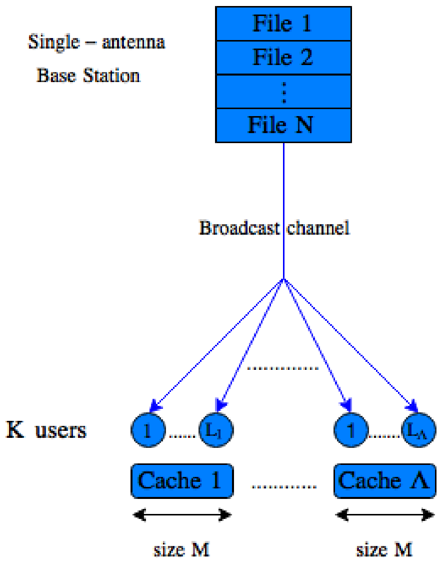

Figure 4.1. Pictorial representation of the addressed problem

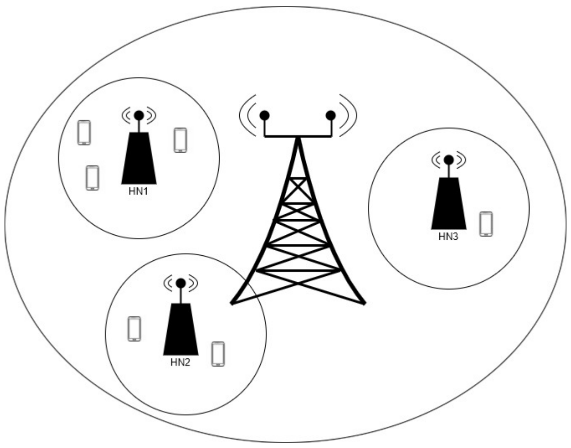

Figure 4.2. System example with $N_0 = 2, K = 6, \Lambda = 3$

Let us now elaborate on the channel model between the base station and the users in the following paragraph.

Channel model The received signal at a receiving user k takes the form

$$y_k = \mathbf{h}_k^T \mathbf{x} + w_k, \quad k = 1, \cdots, K \tag{4.2}$$

where $\mathbf{x}$ is the $N_0 \times 1$ vector of symbols transmitted by the base station that satisfies a power constraint $E(||\mathbf{x}||^2) < P$, where $\mathbf{h}_k \in C^{N_0 \times 1}$ is the (potentially random) fading channel between the N_0 antennas of the base station 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) at the receiving nodes and base station.

4.2.2 Problem Definition

The system operates in two distinct phases: placement phase and delivery phase. In the placement phase during the off-peak hours, helper nodes store content from the library. No coding is applied to the content stored at the helper nodes. This case is commonly referred as the *uncoded cache placement.* Over the course of peak hours in the delivery phase, each user k requests a file W^{d_k} independently, where $d_k \in [N]$.

The files that are requested by K users are given by the following vector $\mathbf{d} = (d_1, \cdots, d_\Lambda)$. For a given user profile, $\mathbf{d}(\mathbf{L}) \overset{\triangle}{=} (\mathbf{d}_{U_1}, \cdots, \mathbf{d}_{U_\Lambda})$ denotes the users' demands where U_λ and $\mathbf{d}_{U_\lambda}$ are the set of users connected to cache λ and the indices of the files requested by those L_λ users, respectively. Each user can download content from its associated helper node at zero cost (and hence no channel model between the helper nodes and users is required). Once the base station becomes aware of the users requests, it transmits a signal of length T through the broadcast link to the K users with a given caching-delivery strategy χ. Then, every user extracts its desired files based on the signal received from the base station and the cached content at its associated helper node. The objective of this work is to characterize the average broadcast rate (delay) T, over all possible user profiles, for the worst-case scenario, i.e. when each user asks a different file. More formally, we define the optimal average broadcast rate as

$$T^* = \min_{\chi} E_{\mathbf{L}} \left[\max_{\mathbf{d}} T(\mathbf{L}, \mathbf{d}, \chi) \right] \tag{4.3}$$

where $T(\mathbf{L}, \mathbf{d}, \chi)$ is the delivery time required by the base station to serve, by using strategy χ, the request $\mathbf{d}$ for a given user profile $\mathbf{L}$ and $E_{\mathbf{L}}[\cdot]$ denotes the expectation over all possible user profiles $\mathbf{L}$. It is important to notice that, from the mathematical formulation of the optimal delay T^*, the base station is aware of the user profile already in the cache placement phase.

Let us now define *profile type* denoted by $\boldsymbol{\mathcal{L}}$ a sorted array in descending order as $\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, \cdots, \mathcal{L}_\Lambda)$, such that $\mathcal{L}_\lambda$ represents the number of users connected to the λ-th most populated cache (helper node). Clearly, the constraint in (4.1) also applies to the profile type. A given user profile is obtained from the profile type via a $\Lambda \times \Lambda$ permutation matrix $\mathbf{P}_\pi$, i.e. $\mathbf{L}^T = \mathbf{P}_\pi \boldsymbol{\mathcal{L}}^T$. There exists $\Lambda!$ user profiles for a given profile type and the set of such profiles is denoted by $S_{\boldsymbol{\mathcal{L}}}$. For example the profile type $\boldsymbol{\mathcal{L}} = (3,2,1)$ means that there are 3 users associated to an helper node, 2 associated to another and 1 user connected to the last helper node. For the aforementioned profile type there exist 6 possible different user profiles. For example the user profiles $\mathbf{L} = (2,1,3)$ and $\mathbf{L} = (3,1,2)$ are two possible profiles in the set $S_{(3,2,1)}$. The optimal average broadcast rate can be rewritten as

$$T^* = \min_{\chi} E_{\boldsymbol{\mathcal{L}}} \left[E_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} \left[\max_{\mathbf{d}} T(\mathbf{L}, \mathbf{d}, \chi) \right] \right] \tag{4.4}$$

where, assuming that each profile $\mathbf{L}$ in the set $S_{\mathcal{L}}$ has the same probability of realization, we have

$$T(\mathcal{L},\chi) = E_{\mathbf{L}\in S_{\mathcal{L}}}\left[\underset{\mathbf{d}}{max}\ T(\mathbf{L},\mathbf{d},\chi)\right] = \frac{1}{|S_{\mathcal{L}}|}\sum_{\mathbf{L}\in S_{\mathcal{L}}}(\underset{\mathbf{d}}{max}\ T(\mathbf{L},\mathbf{d},\chi)). \tag{4.5}$$

4.3 Preliminaries on Index Coding

Before presenting our results, we remind the reader some elements of index coding (IC) [33] which will be fundamental for the derivation of the lower bound on the optimal rate T^*.

Consider a K-receiver MIMO broadcast channel (BC), where the transmitter is equipped with N_0 antennas, and receiver $r_j \in [K]$ is equipped with N_j antennas. Receiver r_j desires message M_j, uniformly distributed over $[2^{T\cdot R_j}]$, which is known by the transmitter. The channel matrices are known to the transmitter and all receivers. Receiver r_j has as side information an ordered set of messages that are desired by other receivers. The transmitter sends a broadcast message X^T to satisfy all users requests, where T is the length of the message. Such a problem can be translated into an index coding problem. An index coding problem can be represented by a directed graph, where the nodes represent the K messages desired by the users and where the side information for each user can be represented by an edge in the directed graph $\mathcal{G} = (\mathcal{V}_{\mathcal{G}}, \mathcal{A}_{\mathcal{G}})$, with $\mathcal{V}_{\mathcal{G}}$ and $\mathcal{A}_{\mathcal{G}}$ being the set of vertices of the graph and being the set of edges, respectively. In a side information graph, there is a directed edge from node i to j if receiver r_j (the receiver that wants message M_j) knows message M_i desired by receiver r_i. In general, for an IC problem receivers and their desired messages coincide, i.e. $i \equiv r_i \equiv M_i$. We denote any vertex-induced subgraph of $\mathcal{G}$ and the degree of freedom of receiver r_j by $\mathcal{J}$ and DoF_j, respectively.

For example, Figure 4.3 represents an index coding problem with 3 nodes, where receiver r_1 knows message M_2, receiver r_3 has both messages M_1 and M_2 has side information and receiver r_2 does not have any message.

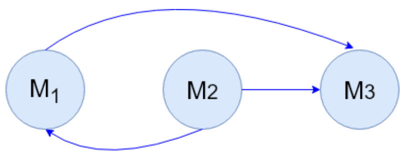

Figure 4.3. Example of an index coding problem with 3 messages

In [34] was recently introduced an upper bound on the DoF region of an IC problem. The following inequality holds for the K-receiver MIMO BC if $(DoF_1, ..., DoF_K)$ is achievable [34, Lemma 1]:

$$\sum_{k\in\mathcal{V}_{\mathcal{J}}} DoF_k \leq \min\{N_0, \sum_{k\in\mathcal{V}_{\mathcal{J}}} N_k\} \tag{4.6}$$

for every acyclic induced subgraph $\mathcal{J}$ of the side information graph $\mathcal{G}$ [3].

Then, the following corollary holds.

Corollary 3. *For an achievable rate tuple $(R_1, ..., R_K)$ of an IC problem with the associated side information graph $\mathcal{G}$, the sum rate is bounded as*

$$\sum_{k\in\mathcal{V}_{\mathcal{J}}} \frac{|M_k|}{T} = \sum_{k\in\mathcal{V}_{\mathcal{J}}} R_k \leq min\{N_0, \sum_{k\in\mathcal{V}_{\mathcal{J}}} N_k\} \tag{4.7}$$

for every acyclic induced subgraph $\mathcal{J}$ of $\mathcal{G}$ where $|M_k|$ and T are the size of the message desired by receiver r_k and the units of time needed to serve all users, respectively.

Proof. The proof of the corollary follows directly from equation (4.3) by simply recalling that T is the high-SNR normalized delay. Consequently, in the high SNR setting of interest, we can directly go from the DoF region to the rate region. The clean connection between DoF and rate has been presented in the context of coded caching in [35],[36] and [37]. □

In the caching problem addressed in this chapter, all users are equipped with a single antenna. Hence, Corollary 3 reduces to

$$\sum_{k\in\mathcal{V}_{\mathcal{J}}} \frac{|M_k|}{T} = \sum_{k\in\mathcal{V}_{\mathcal{J}}} R_k \leq N_0 \tag{4.8}$$

for every acyclic induced subgraph $\mathcal{J}$ of the side information graph $\mathcal{G}$[4].

4.4 Main Results

In this section we are going to present and comment the main results developed in this chapter whose details and proofs will be given in the subsequent section.

The main contribution for the adressed caching problem is the development of a lower bound on the optimal delay T^*. Inspired by the work in [6] where it was proved the optimality of the MAN scheme for the single-stream broadcast channel already discussed in the first chapter, this bound has been developed by first converting the caching problem into an index coding problem. Then, the index coding bound in equation (4.8) will be heavily used along with some basic graph theory and combinatorial math. The lower bound is given in the following theorem.

[3][34, Lemma 1] is presented for the 3-receiver case only, however its proof remains valid for a system with more users as long as the corresponding subgraph is acyclic.

[4]Here, we are assuming that the number of users K is higher than the number of antennas N_0 at the transmitter, which is the regime of interest.

Theorem 6. *For the adressed caching problem with a library of N files, Λ different caches each of size M and N_0 antennas at the transmitter, the optimal delay for the worst-case demands T^* is lower bounded by*

$$T^* \geq E_{\boldsymbol{\mathcal{L}}} \left[Conv \left(\frac{1}{N_0} \frac{\sum_{r=1}^{\Lambda-i} \mathcal{L}_r \binom{\Lambda-r}{i}}{\binom{\Lambda}{i}} \right) \right] \tag{4.9}$$

where $Conv(f(i))$ is the lower convex envelope defined for the points $\{(i, f(i)) | i \in \{0,1,...,K\}\}$.

The proof of Theorem 6 is given in section 4.5.

Fixed the profile type $\boldsymbol{\mathcal{L}}$, the optimal delay is defined as

$$T^*(\boldsymbol{\mathcal{L}}) = \min_{\chi} E_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} \left[\max_{\mathbf{d}} T(\mathbf{L}, \mathbf{d}, \chi) \right] \tag{4.10}$$

The following corollary comes directly from Theorem 6.

Corollary 4. *For the addressed caching problem, given the profile type $\boldsymbol{\mathcal{L}}$, the optimal worst-case delay is lower bounded by*

$$T^*(\boldsymbol{\mathcal{L}}) \geq Conv \left(\frac{1}{N_0} \frac{\sum_{r=1}^{\Lambda-i} \mathcal{L}_r \binom{\Lambda-r}{i}}{\binom{\Lambda}{i}} \right) \tag{4.11}$$

It is worth recalling that, by the definition of the optimal achievable delay, the base station is aware of the user profile already in the cache placement phase.

Let us now present the results obtained from the developed achievable schemes.

Definition 1. *A profile type $\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, ..., \mathcal{L}_\Lambda)$ is said to be N_0-admissible if the following condition is satisfied:*

$$\mathcal{L}_\lambda = \sum_{j=1}^{P} n_{\lambda,j} \cdot A_j \quad n_{\lambda,j} \in \mathbb{N},\ A_j \in \mathbb{N},\ N_0 \leq A_j < 2N_0 \text{ and } \forall \lambda \in [\Lambda], P \in \mathbb{N}. \tag{4.12}$$

Remark 1. *If a profile type $\boldsymbol{\mathcal{L}}$ is N_0-admissible, then all user profiles $\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}$ are said to be N_0-admissible.*

Remark 2. *Note that all user profiles are 1-admissible.*

For the single-antenna case, i.e. $N_0 = 1$, a scheme has been developed for any user profile $\mathbf{L}$. Such a scheme is a variation of the MAN scheme in [5]. On the other hand, for the multiple-antennas case, extensions of the scheme presented in [18] are developed for the N_0-admissible user profiles. The schemes achieve the same performance for all the user profiles $\mathbf{L}$ of a given profile type $\boldsymbol{\mathcal{L}}$. The description of the schemes is omitted on purpose in this document. However, the achieved performance is given in the following theorems.

Theorem 7. *For the addressed caching problem, where a single-antenna base station has a library of N files, K users request different files from the library, each user having access at zero cost to one of Λ different caching helper nodes with memory M according to user profile $\mathbf{L}$, the following delay is achievable:*

$$T(\mathbf{L}) = \frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \mathcal{L}_r \binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} \tag{4.13}$$

where γ is the helper nodes' cache size normalized by the size of the library $\gamma \triangleq \frac{M}{N}$ and $\boldsymbol{\mathcal{L}} = (\mathcal{L}_1, ..., \mathcal{L}_\Lambda)$ is the profile type of the considered user profile $\mathbf{L}$, i.e. $\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}$.

Proof. The proof follows directly from the here-omitted achievable scheme. □

Corollary 5. *For the addressed problem in the single-antenna case, the following delay is achievable and information theoretical optimal:*

$$T^* = E_{\boldsymbol{\mathcal{L}}}\left[\frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \mathcal{L}_r \binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}}\right] \tag{4.14}$$

Proof. The proof follows straight from Theorem 6 and the average over all profile types of the rate achieved in Theorem 7. The achievable delay and the lower bound match. □

From the above Theorems and Corollaries, it is clear how the achieved delay is affected by the way the users distribute among the caches. In particular, the more skewed is the profile type of a given user profile the higher is time needed by the base station to satisfy all requests. This behaviour is expected since the more skewed is the profile the less are the multicasting opportunities that arise from the cache placement. In fact, it is easy to check that if all users show up in the coverage area of the same helper node and hence they all associate to it, then they cannot benefit of any multicast opportunity because they would have access to the same cached content. As a consequence, only the local caching gain would be obtained. This last statement is easily proved below.

The optimal delay for the user profiles $\mathbf{L}$ such that $L_i = K$ for a certain $i \in [\Lambda]$ and $L_j = 0$ for all $j \in [\Lambda]\backslash\{i\}$ is

$$T^*(\mathbf{L}) = \frac{K\binom{\Lambda-1}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} = K(1-\gamma) \tag{4.15}$$

Figure 4.4 shows the rates achieved for a system with $N_0 = 1$ antennas at the base

station, $K = 30$ users, $N = 30$ files and $\Lambda = 6$ helper nodes for the profile types $\boldsymbol{\mathcal{L}}_1 = (30,0,0,0,0,0)$,$\boldsymbol{\mathcal{L}}_2 = (13,8,4,2,2,1)$,$\boldsymbol{\mathcal{L}}_3 = (8,6,4,4,4,4)$ and $\boldsymbol{\mathcal{L}}_4 = (5,5,5,5,5)$. The impact of the profile is evident and the lowest delay is obtained when users distributes uniformly across the caches, i.e. when the profile type is $\boldsymbol{\mathcal{L}}_4$.

The next theorem states the achievable delay by the developed algorithms for the multi-antennas setting.

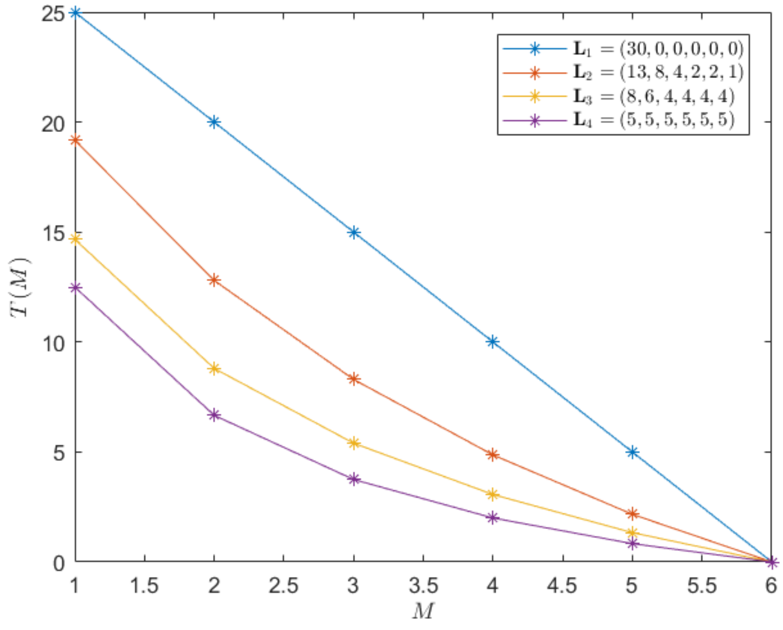

Figure 4.4. Plot of the optimal achievable memory-rate trade-off for different user profiles

Theorem 8. *For the addressed caching problem, for the case when the base station is equipped with N_0 antennas, the following delay is achievable and information theoretical optimal:*

$$T^*(\boldsymbol{\mathcal{L}}) = \frac{1}{N_0} \frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \mathcal{L}_r \binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} \tag{4.16}$$

where $\boldsymbol{\mathcal{L}}$ is N_0-admissible.

Proof. The proof follows directly from the here-omitted achievable schemes and from the matching lower bound in Corollary 4. □

It is important to highlight that the rate/delay given in Theorem 7 and 8 are achieved without the need from the base station to know the user profile already in the cache placement phase. Thus, since the schemes turn out to be optimal then the knowledge of the user profile in the placement phase is not really useful and even with such a knowledge the base station could not do better.

We now state a corollary from Theorem 8 which allows us to make a very important observation.

Corollary 6. *For the adressed caching problem with a library of N files, Λ different caches each of size M, a symmetric user profile, i.e. $L_\lambda = \frac{K}{\Lambda}, \forall \lambda \in [\Lambda]$ and $N_0 \leq \frac{K}{\Lambda}$, the* *optimal worst-case delay is obtained as*

$$T^*(\mathbf{L}) = \frac{K(1-\gamma)}{N_0(\Lambda\gamma+1)} \tag{4.17}$$

.

Proof. Given that $L_\lambda = \frac{K}{\Lambda}, \forall \lambda \in [\Lambda]$ we have

$$T^*(\mathbf{L}) = \frac{1}{N_0}\frac{\sum_{r=1}^{\Lambda-\Lambda\gamma} \frac{K}{\Lambda}\binom{\Lambda-r}{\Lambda\gamma}}{\binom{\Lambda}{\Lambda\gamma}} = \frac{K}{N_0\cdot\Lambda}\frac{\binom{\Lambda}{\Lambda\gamma+1}}{\binom{\Lambda}{\Lambda\gamma}} = \frac{K(1-\gamma)}{N_0(\Lambda\gamma+1)} \tag{4.18}$$

□

The result in Corollary 6 is of relevant interest. We recall that in the MISO broadcast channel where the base station has N_0 antennas and each user has an isolated cache the order-optimal achieved rate is $T = \frac{K(1-\gamma)}{K\gamma+N_0}$ [17]. Thus, whenever there are exactly K caches as the number of users the multiplexing gain of the multiple antennas at the transmitter is additive to the caching gain $K\gamma$. This result is consistent with the optimal rate achieved in the case of single antenna $T^* = \frac{K(1-\gamma)}{K\gamma+1}$. Corollary 6 tells us that when the number of caches is smaller than the number of users $\Lambda < K$ and the number of antennas N_0 is smaller than the number of users associated to a given cache, i.e. $N_0 \leq \frac{K}{\Lambda}$, then the multiplexing gain N_0 is multiplicative to the caching gain $\Lambda\gamma + 1$.

4.5 Proof of the Lower Bound

Before proceeding with the proof of the lower bound given in Theorem 6 we provide an example to highlight the key ideas behind this proof.

4.5.1 Explanatory Example

Assume that in the addressed problem the library has $N = 6$ files and it is stored at a base station with $N_0 = 2$ antennas. The number of users in the system is $K = 6$, there are three caches (helper nodes) $\Lambda = 3$ and users are distributed according to one of the user profiles with profile type $\boldsymbol{\mathcal{L}} = (4,1,1)$. Our goal is to find a lower bound for the optimal delay

$$T^*(\boldsymbol{\mathcal{L}}) = \underset{\chi}{min}\ T(\boldsymbol{\mathcal{L}},\chi) = \underset{\chi}{min}\ E_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}}\left[\underset{\mathbf{d}}{max}\ T(\mathbf{L},\mathbf{d},\chi)\right] \tag{4.19}$$

which can be lower bounded as

$$\begin{aligned}\underset{\chi}{min}\ E_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}}\left[\underset{\mathbf{d}}{max}\ T(\mathbf{L},\mathbf{d},\chi)\right] &\geq \underset{\chi}{min}\ \underset{\mathbf{d}}{max}\ E_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}}\left[T(\mathbf{L},\mathbf{d},\chi)\right] &(4.20)\\ &= \underset{\chi}{min}\ \underset{\mathbf{d}}{max}\ T(\boldsymbol{\mathcal{L}},\mathbf{d},\chi) &(4.21)\end{aligned}$$

where

$$T(\boldsymbol{\mathcal{L}},\mathbf{d},\chi) = \frac{1}{\Lambda!}\sum_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}} T(\mathbf{L},\mathbf{d},\chi) \tag{4.22}$$

Since we are interested in the worst-case delay $\underset{\mathbf{d}}{max}\ T(\mathbf{L}, \mathbf{d}, \chi)$ we will only consider those demand vectors in which all users request different files. We denote the set of such demands by $S_{\mathbf{d}}$. Then, we have

$$\underset{\mathbf{d}}{max}\ T(\boldsymbol{\mathcal{L}}, \mathbf{d}, \chi) = \frac{1}{|S_{\mathbf{d}}|} \sum_{\mathbf{d} \in S_{\mathbf{d}}} T(\boldsymbol{\mathcal{L}}, \mathbf{d}, \chi). \tag{4.23}$$

For example, assume users request files whose index are in the demand vector $\mathbf{d} = (1,2,3,4,5,6)$, which can be rewritten as $\mathbf{d} = (\mathbf{d}_{\mathcal{L}_1}, \mathbf{d}_{\mathcal{L}_2}, \mathbf{d}_{\mathcal{L}_3})$ for the case of three caches, with the corresponding demands $\mathbf{d}_{\mathcal{L}_1} = (1,2,3,4)$, $\mathbf{d}_{\mathcal{L}_2} = (5)$ and $\mathbf{d}_{\mathcal{L}_3} = (6)$. Here $\mathbf{d}_{\mathcal{L}_i}$ denotes the demands of the users connected to the helper node associated to $\mathcal{L}_i$ users. For a fixed user profile $\mathbf{L}$, the demand vector yields $\mathbf{d}(\mathbf{L})^T = \mathbf{P}_\pi \mathbf{d}^T = \mathbf{P}_\pi(\mathbf{d}_{\mathcal{L}_1}, ..., \mathbf{d}_{\mathcal{L}_\lambda})^T \triangleq (\mathbf{d}_{U_1}, ..., \mathbf{d}_{U_\Lambda})^T$, where $\mathbf{P}_\pi$ is a $\Lambda \times \Lambda$ permutation matrix. Thus, in our example, the permutation $\pi = (3,1,2)$ yields $\mathbf{L}^T = \mathbf{P}_\pi \boldsymbol{\mathcal{L}}^T = (1,4,1)$ and the demand vector $\mathbf{d} = (1,2,3,4,5,6)$ has to be read as $\mathbf{d}(\mathbf{L})^T = \mathbf{P}_\pi(\mathbf{d}_{\mathcal{L}_1}, \mathbf{d}_{\mathcal{L}_2}, \mathbf{d}_{\mathcal{L}_3})^T = (\mathbf{d}_{U_1}, \mathbf{d}_{U_2}, \mathbf{d}_{U_3})^T$ with $\mathbf{d}_{U_1} = (6), \mathbf{d}_{U_2} = (1,2,3,4), \mathbf{d}_{U_3} = (5)$. Hence, having the demand $\mathbf{d} = (1,2,3,4,5,6)$ and the user profile $\mathbf{L} = (1,4,1)$ means that the user connected to cache 1 requests file W^6, the four users connected to cache 2 request files W^1, W^2, W^3, W^4 and user connected to helper node 3 requests file W^5.

Each file W^i is divided into $2^\Lambda = 2^3 = 8$ disjoint subfiles, denoted as $W^i_\tau, \tau \in 2^{[3]}$ where τ indicates the helper node the subfile W^i_τ is cached in and the power set of [3] is $2^{[3]} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$. For instance, W^i_{13} is a subfile of file W^i stored in caches 1 and 3.

Imagine a demand vector $\mathbf{d}$ and a user profile obtained from $\boldsymbol{\mathcal{L}}$ via all $\Lambda \times \Lambda$ permutation matrices $\mathbf{P}_\pi$. Similarly to [6], this problem is translated into an index coding problem with a side information graph having $K2^{\Lambda-1} = 6 \cdot 2^2$ nodes. In fact, a user k requesting file W^{d_k} is actually requesting only $2^{\Lambda-1} = 4$ subfiles out of the 8 subfiles into which is file is split. This is because, user k has access to the subfiles cached in the helper node it is connected to. In order to draw the side information graph associated to this problem, for each requested file $W^{\mathbf{d}_{U_\lambda(j)}}$, we draw the 4 subfiles requested by user $U_\lambda(j)$ that are not stored in its associated helper node. Hence, a given user of the caching problem requiring 4 subfiles from the base station, is replaced in the IC problem by 4 different new users, each requesting a different subfile and being connected to the same helper node λ as the original user. This will become more clear with the following example.

Consider the usual demand $\mathbf{d} = (1,2,3,4,5,6)$ and user profile $\mathbf{L} = (1,4,1)$, then we recall that we have $\mathbf{d}_{U_1} = (6), \mathbf{d}_{U_2} = (1,2,3,4), \mathbf{d}_{U_3} = (5)$. For the chosen demand vector and user profile, the nodes of the corresponding side information graph are depicted in Figure 4.5. In each row the subfiles requested from a given coded caching user are drawn. It has to be noticed that all the (drawn) subfiles requested from users connected to the same helper node has the same indices τ. Next, we recall from the preliminaries that in the side information graph there exist a directed edge from a node i to a node j if the user corresponding to node j has the message corresponding to node i. In the index coding problem obtained from the here-considered caching problem, all nodes/messages/new users in a given row have the same side information because they arise from the same coded caching user that is connected to a given cache, thus all the nodes in a row have the same incoming edges. In our considered example, there is an edge from W_1^1 to all nodes $W_0^6, W_2^6, W_3^6, W_{23}^6$ since the caching user requesting file W^6 has access to the subfile W_1^1 that is stored in its associated helper node 1. In Figure 4.5 all the edges are not drawn to avoid that the image becomes not readable since the number of edges is quite large.

Given the side information graph, we can now proceed to develop a lower bound by using equation (4.8).

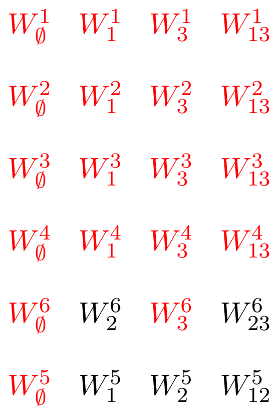
$W_\emptyset^1$ W_1^1 W_3^1 W_{13}^1
$W_\emptyset^2$ W_1^2 W_3^2 W_{13}^2
$W_\emptyset^3$ W_1^3 W_3^3 W_{13}^3
$W_\emptyset^4$ W_1^4 W_3^4 W_{13}^4
$W_\emptyset^6$ W_2^6 W_3^6 W_{23}^6
$W_\emptyset^5$ W_1^5 W_2^5 W_{12}^5

Figure 4.5. Nodes of the side information graph for the demand vector $\mathbf{d}_{U_1} = (6), \mathbf{d}_{U_2} = (1,2,3,4), \mathbf{d}_{U_3} = (5)$

For each permutation vector $\boldsymbol{\sigma} \triangleq (\sigma_1, \sigma_2, \sigma_3)$ of $\{1,2,3\}$, a set of nodes inducing an acyclic subgraph is

$$\{W_{\tau_1}^{\mathbf{d}_{U_{\sigma_1}}(j)}\}_{j=1}^{L_{\sigma_1}} \text{ for all } \tau_1 \subseteq [3] \setminus \{\sigma_1\} \tag{4.24}$$

$$\{W_{\tau_2}^{\mathbf{d}_{U_{\sigma_2}}(j)}\}_{j=1}^{L_{\sigma_2}} \text{ for all } \tau_2 \subseteq [3] \setminus \{\sigma_1, \sigma_2\} \tag{4.25}$$

$$\{W_{\tau_3}^{\mathbf{d}_{U_{\sigma_3}}(j)}\}_{j=1}^{L_{\sigma_3}} \text{ for all } \tau_3 \subseteq [3] \setminus \{\sigma_1, \sigma_2, \sigma_3\} = \emptyset. \tag{4.26}$$

where $\mathbf{d}_{U_\lambda}(j)$ denotes the index of the j-th file requested from helper node λ.

The proof that such a choice of nodes gives an acyclic graph is given in the general proof.

Among all possible permutation vectors $\boldsymbol{\sigma}$, we choose the one with the biggest acyclic subgraph. The choice of the biggest acyclic subgraph provide a tighter lower bound. A different choice of an acyclic subgraph would make the bound looser (see [33, Theorem

3]). It can be easily verified that in this example the permutation vectors (2,1,3) and (2,3,1) both give a maximum acyclic subgraph. We choose the permutation $\boldsymbol{\sigma} = (2,1,3)$. The motivation of this choice is again given in the general proof. The nodes of the acyclic subgraph induced by such a permutation are

$$\{W_{\tau_1}^{\mathbf{d}_{U_2}(j)}\}_{j=1}^{4} = \{W_{\tau_1}^{1}, W_{\tau_1}^{2}, W_{\tau_1}^{3}, W_{\tau_1}^{4}\} \text{ for all } \tau_1 \subseteq [3] \setminus \{2\} \tag{4.27}$$

$$\{W_{\tau_2}^{\mathbf{d}_{U_1}(j)}\}_{j=1}^{1} = \{W_{\tau_2}^{6}\} \text{ for all } \tau_2 \subseteq [3] \setminus \{2,1\} \tag{4.28}$$

$$\{W_{\tau_3}^{\mathbf{d}_{U_3}(j)}\}_{j=1}^{1} = \{W_{\tau_3}^{5}\} \text{ for all } \tau_3 \subseteq [3] \setminus \{2,1,3\} = \emptyset. \tag{4.29}$$

and they are coloured in red in Figure 4.5. For this choice of the acyclic induced subgraph, (4.8) results in the following bound on the delay

$$\begin{aligned} T(\mathbf{L}, \mathbf{d}, \chi) \geq \frac{1}{2} \Big(& |W_{\emptyset}^{1}| + |W_{1}^{1}| + |W_{3}^{1}| + |W_{13}^{1}| + |W_{\emptyset}^{2}| \\ & + |W_{1}^{2}| + |W_{3}^{2}| + |W_{13}^{2}| + |W_{\emptyset}^{3}| + |W_{1}^{3}| + |W_{3}^{3}| \\ & + |W_{13}^{3}| + |W_{\emptyset}^{4}| + |W_{1}^{4}| + |W_{3}^{4}| + |W_{13}^{4}| + |W_{\emptyset}^{5}| \\ & + |W_{\emptyset}^{6}| + |W_{3}^{6}| \Big) \end{aligned} \tag{4.30}$$

A similar lower bound can be built for each of the remaining 5 different user profiles $\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}$. The sum of all gives the lower bound on $T(\boldsymbol{\mathcal{L}}, \mathbf{d}, \chi)$. Subsequently, the procedure is repeated for all other demand vectors $\mathbf{d} \in S\mathbf{d}$ at which point we obtain a lower bound on $\underset{\mathbf{d}}{max}\ T(\boldsymbol{\mathcal{L}}, \mathbf{d}, \chi)$. Overall, we obtain

$$\sum_{\mathbf{d} \in S_{\mathbf{d}}} \sum_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} T(\mathbf{L}, \mathbf{d}, \chi) \geq \frac{1}{2} \sum_{\mathbf{d} \in S_{\mathbf{d}}} \sum_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} \sum_{\lambda=1}^{3} \sum_{j=1}^{L_{\sigma_\lambda}} \sum_{\tau_\lambda \subseteq [3] \setminus \{\sigma_1, \ldots, \sigma_\lambda\}} |W_{\tau_\lambda}^{\mathbf{d}_{U_{\sigma_\lambda}}(j)}| \tag{4.31}$$

By dividing by the total number of sums 6!3! we get

$$\frac{1}{|S_{\mathbf{d}}|} \frac{1}{\Lambda!} \sum_{\mathbf{d} \in S_{\mathbf{d}}} \sum_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} T(\mathbf{L}, \mathbf{d}, \chi) \geq \frac{1}{6!3!} \frac{1}{2} \sum_{\mathbf{d} \in S_{\mathbf{d}}} \sum_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} \sum_{\lambda=1}^{3} \sum_{j=1}^{L_{\sigma_\lambda}} \sum_{\tau_\lambda \subseteq [3] \setminus \{\sigma_1, \ldots, \sigma_\lambda\}} |W_{\tau_\lambda}^{\mathbf{d}_{U_{\sigma_\lambda}}(j)}| \tag{4.32}$$

and hence from equations (4.22) and (4.23)

$$\underset{\mathbf{d}}{max}\ T(\boldsymbol{\mathcal{L}}, \mathbf{d}, \chi) \geq \frac{1}{6!3!} \frac{1}{2} \sum_{\mathbf{d} \in S_{\mathbf{d}}} \sum_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} \sum_{\lambda=1}^{3} \sum_{j=1}^{L_{\sigma_\lambda}} \sum_{\tau_\lambda \subseteq [3] \setminus \{\sigma_1, \ldots, \sigma_\lambda\}} |W_{\tau_\lambda}^{\mathbf{d}_{U_{\sigma_\lambda}}(j)}| \tag{4.33}$$

From equations (4.19),(4.20), (4.21) and (4.33) we obtain

$$\underset{\chi}{min}\ T(\mathcal{L}, \chi) \geq \underset{\chi}{min}\ \frac{1}{2} \sum_{i=0}^{3} \frac{\sum_{r=1}^{3-i} \mathcal{L}_r \binom{3-r}{i}}{6\binom{3}{i}} x_i \tag{4.34}$$

$$\overset{(a)}{=} \underset{\chi}{min}\ \frac{1}{2} Conv\left(\frac{\sum_{r=1}^{3-i} \mathcal{L}_r \binom{3-r}{i}}{\binom{3}{i}} \right) \tag{4.35}$$

$$= \frac{1}{2} Conv\left(\frac{\sum_{r=1}^{3-i} \mathcal{L}_r \binom{3-r}{i}}{\binom{3}{i}} \right). \tag{4.36}$$

where $x_i = \sum_{n\in[N]} \sum_{\tau\subseteq[\Lambda]:|\tau|=i} |W_\tau^n|,\ i \in [\Lambda] \cup \{0\}$ is the total size of the subfiles cached at exactly i helper nodes. The coefficient $\frac{\sum_{r=1}^{3-i} \mathcal{L}_r \binom{3-r}{i}}{6\binom{3}{i}}$ and (a) are proved in the general proof.

4.5.2 General Proof of the Lower Bound

Our goal here is to provide a tight lower bound for the optimal average broadcast rate

$$T^* = \min_{\chi} E_{\boldsymbol{\mathcal{L}}} \left[E_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}} \left[\max_{\mathbf{d}} T(\mathbf{L}, \mathbf{d}, \chi) \right] \right] \tag{4.37}$$

which, as a first step, we can lower bound as

$$\begin{aligned} T^* &\geq E_{\boldsymbol{\mathcal{L}}} \left[\min_{\chi} E_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}} \left[\max_{\mathbf{d}} T(\mathbf{L}, \mathbf{d}, \chi) \right] \right] = E_{\boldsymbol{\mathcal{L}}} \left[T^*(\boldsymbol{\mathcal{L}}) \right] &(4.38)\\ &\geq E_{\boldsymbol{\mathcal{L}}} \left[\min_{\chi} \max_{\mathbf{d}} E_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}} [\ T(\mathbf{L}, \mathbf{d}, \chi)] \right] &(4.39)\\ &= E_{\boldsymbol{\mathcal{L}}} \left[\min_{\chi} \frac{1}{|S_{\mathbf{d}}|} \frac{1}{|S_{\boldsymbol{\mathcal{L}}}|} \sum_{\mathbf{d}\in S_{\mathbf{d}}} \sum_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}} T(\mathbf{L}, \mathbf{d}, \chi) \right] &(4.40)\\ &\geq E_{\boldsymbol{\mathcal{L}}} \left[\min_{\chi} \frac{1}{|S_{\mathbf{d}}|} \frac{1}{|S_{\boldsymbol{\mathcal{L}}}|} \sum_{\mathbf{d}\in S_{\mathbf{d}}} \sum_{\mathbf{L}\in S_{\boldsymbol{\mathcal{L}}}} T^{LB}(\mathbf{L}, \mathbf{d}, \chi) \right] &(4.41) \end{aligned}$$

where $T^{LB}(\mathbf{L}, \mathbf{d}, \chi)$ is a certain lower bound for the delay $T(\mathbf{L}, \mathbf{d}, \chi)$.

Theorem 6 is proved through the translation of the caching problem into an IC problem. Consider the uncoded cache placement, a given profile type $\boldsymbol{\mathcal{L}}$ and a demand vector $\mathbf{d}$. For a $\Lambda \times \Lambda$ permutation matrix $\mathbf{P}_\pi$ applied to the demand vector, we get $\mathbf{d}(\mathbf{L})^T = \mathbf{P}_\pi \mathbf{d}^T$. In total, there are $\Lambda!$ demand vectors $\mathbf{d}(\mathbf{L})$. Given $\mathbf{d}(\mathbf{L})$, we can construct the associated side information graph which will denote by $\mathcal{G}$.

.

In the corresponding side information graph of $\mathbf{d}(\mathbf{L})$, each requested file $W^{\mathbf{d}_{\mathcal{U}_\lambda}(j)}$ for $\lambda \in [\Lambda]$, $j \in [L_\lambda]$ is divided into 2^Λ subfiles $(W_\tau^{\mathbf{d}_{\mathcal{U}_\lambda}(j)},\ \tau \in 2^{[\Lambda]})$ and each of them is requested by a *new* user that is connected to the same helper node λ as the *original* user. Note that the new users have the same side information of the original ones. The subfiles $W_\tau^{\mathbf{d}_{\mathcal{U}_\lambda}(j)},\ \lambda \in \tau$ are not requested by the original user because they can be obtained from the helper node λ. In other words, all users connected to the same cache have access to all files stored in this cache. The total number of IC users connected to a particular helper node λ is $2^{\Lambda-1} L_\lambda$. The side information graph associated to this problem $\mathcal{G}$ has $K \cdot 2^{\Lambda-1}$ nodes.

The bound given by (4.8) is valid for the acyclic induced subgraphs of $\mathcal{G}$. The following lemma states how to find the sets of nodes that induce acyclic subgraphs as an adaptation of [6, Lemma 1].

Lemma 1. *An induced acyclic subgraph $\mathcal{J} \subseteq \mathcal{G}$ of the addressed IC problem contains subfiles $W_{\tau_\lambda}^{\mathbf{d}_{U_{\sigma_\lambda}}(j)}$, $j \in [L_{\sigma_\lambda}]$, $\forall \lambda \in [\Lambda]$ for all $\tau_\lambda \subseteq [\Lambda] \setminus \{\sigma_1, ..., \sigma_\lambda\}$ where $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_\Lambda)$ is a permutation of $[\Lambda]$.*

Proof. It is enough to say that Lemma 1 is an adaptation of [6, Lemma 1] to our setting, where users associated to same helper nodes have access to the same subfiles. Since [6, Lemma 1] gives an acyclic graph, then also our construction results in a graph that is acyclic. □

Lemma 2. *Lemma 1 used with the permutation vector $\boldsymbol{\sigma}$ where $L_{\sigma_1} \geq L_{\sigma_2} \geq ... \geq L_{\sigma_\Lambda}$ results in an acyclic subgraph that is the maximum in terms of number of nodes among all the ones that can be created with Lemma 1.*

Proof. Given a permutation vector $\boldsymbol{\sigma}$, the number of subfiles $W_{\tau_\lambda}^{\mathbf{d}_{U_{\sigma_\lambda}}(j)}$ in the acyclic subgraph such that the index size is $|\tau_\lambda| = i$, $i \in [\Lambda] \cup \{0\}$ is $\sum_{r=1}^{\Lambda-i} L_{\sigma_r} \binom{\Lambda-r}{i}$. The total number of subfiles in the acyclic subgraph is $\sum_{i=0}^{\Lambda} \sum_{r=1}^{\Lambda-i} L_{\sigma_r} \binom{\Lambda-r}{i}$. This number is maximized when the vector $(L_{\sigma_1}, ..., L_{\sigma_\Lambda})$ is sorted in a descending order which coincides with the profile type $\boldsymbol{\mathcal{L}}$.

Note: If several caches have the same number of users connected to them, then there exists more than one permutation vector $\boldsymbol{\sigma}$ through Lemma 1, that results in the maximum acyclic subgraph. Let Σ be the set of permutations $\boldsymbol{\sigma}$ yielding a maximum acyclic subgraph defined as $\Sigma \triangleq \{\boldsymbol{\sigma} : L_{\sigma_1} \geq L_{\sigma_2} \geq ... \geq L_{\sigma_\Lambda}\}$ and let Δ be the set of $\sigma_\lambda, \lambda \in [\Lambda]$ for which the permutations $\boldsymbol{\sigma} \in \Sigma$ differ for the order. Then, among all possible $\boldsymbol{\sigma} \in \Sigma$ the permutation that is chosen is the one where the set Δ appears in ascending order in $\boldsymbol{\sigma}$. □

By using Lemma 1 and Lemma 2 the lower bound $T^{LB}(\mathbf{L}, \mathbf{d}, \chi)$ is obtained by (4.8) and is given below.

$$T^{LB}(\mathbf{L}, \mathbf{d}, \chi) = \frac{1}{N_0} \cdot \sum_{j=1}^{L_{\sigma_1}} \sum_{\tau_1 \subseteq [\Lambda] \setminus \{\sigma_1\}} |W_{\tau_1}^{\mathbf{d}_{U_{\sigma_1}}(j)}| + \cdots + \frac{1}{N_0} \cdot \sum_{j=1}^{L_{\sigma_\Lambda}} \sum_{\tau_\Lambda \subseteq [\Lambda] \setminus \{\sigma_1, ..., \sigma_\Lambda\}} |W_{\tau_\Lambda}^{\mathbf{d}_{U_{\sigma_\Lambda}}(j)}| \tag{4.42}$$

where for clarity we recall that $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_\Lambda)$ is the one chosen via Lemma 2.

Taking the average of (4.42) over all $\Lambda!$ user profiles $\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}$, we get a lower bound on $T(\boldsymbol{\mathcal{L}}, \mathbf{d}, \chi)$. Then, we repeat the same procedure for all $P(N, K)$ demand vectors $\mathbf{d} \in S_{\mathbf{d}}$. Overall, we obtain the following lower bound:

$$\begin{aligned} &\frac{1}{P(N,K)} \frac{1}{\Lambda!} \sum_{\mathbf{d} \in S_{\mathbf{d}}} \sum_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} T(\mathbf{L}, \mathbf{d}, \chi) \\ &\geq \frac{1}{N_0} \frac{1}{P(N,K)} \frac{1}{\Lambda!} \sum_{\mathbf{d} \in S_{\mathbf{d}}} \sum_{\mathbf{L} \in S_{\boldsymbol{\mathcal{L}}}} \sum_{\lambda \in [\Lambda]} \sum_{j=1}^{L_{\sigma_\lambda}} \sum_{\tau_\lambda \subseteq [\Lambda] \setminus \{\sigma_1, ..., \sigma_\lambda\}} |W_{\tau_\lambda}^{\mathbf{d}_{U_{\sigma_\lambda}}(j)}| \end{aligned} \tag{4.43}$$

Equation (4.43) is rewritten as follows

$$\begin{aligned}&\frac{1}{P(N,K)}\frac{1}{\Lambda!}\sum_{\mathbf{d}\in S_{\mathbf{d}}}\sum_{\mathbf{L}\in S_{\mathcal{L}}}T(\mathbf{L},\mathbf{d},\chi)\\ &\geq\frac{1}{N_0}\frac{1}{P(N,K)}\frac{1}{\Lambda!}\sum_{i=0}^{\Lambda}\sum_{n\in[N]}\sum_{\tau\subseteq[\Lambda]:|\tau|=i}\sum_{\mathbf{d}\in S_{\mathbf{d}}}\sum_{\mathbf{L}\in S_{\mathcal{L}}}\mathbb{1}_{\Omega_{\mathbf{d},\mathbf{L}}}(W_\tau^n)|W_\tau^n| \end{aligned}\tag{4.44}$$

where $\Omega_{\mathbf{d},\mathbf{L}}$ is the set of subfiles of the induced maximum acyclic subgraph for the demand vector $\mathbf{d}(\mathbf{L})$ chosen using Lemma 1 and Lemma 2. The indicator function $\mathbb{1}_{\Omega_{\mathbf{d},\mathbf{L}}}(W_\tau^n)$ is 1 if the subfile W_τ^n is chosen as part of the maximum acyclic graph for the demand vector $\mathbf{d}(\mathbf{L})$ and is 0 otherwise. In equation (4.44) we used the fact that the sum of all files in the library can be written in the form

$$N=\sum_{i=0}^{\Lambda}\sum_{n\in[N]}\sum_{\tau\subseteq[\Lambda]:|\tau|=i}|W_\tau^n|\tag{4.45}$$

In (4.44), we redefine $Q_i(W_\tau^n)\triangleq\sum_{\mathbf{d}\in S_{\mathbf{d}}}\sum_{\mathbf{L}\in S_{\mathcal{L}}}\mathbb{1}_{\Omega_{\mathbf{d},\mathbf{L}}}(W_\tau^n)$ which is given by the following Lemma.

Lemma 3. *Subfile W_τ^n, $|\tau|=i$ appears in all $P(N,K)\Lambda!$ constructed lower bounds through Lemma 1 and Lemma 2 $Q_i(W_\tau^n)$ number of times where*

$$\begin{aligned}Q_i(W_\tau^n)=&\binom{N-1}{K-1}\sum_{r=1}^{\Lambda}P(\Lambda-i-1,r-1)\\ &\times(\Lambda-r)!\mathcal{L}_rP(K-1,\mathcal{L}_r-1)(K-\mathcal{L}_r)!(\Lambda-i)\end{aligned}\tag{4.46}$$

Proof. There are $\binom{N-1}{K-1}$ subsets $\Upsilon_m, m\in[\binom{N-1}{K-1}]$ out of $\binom{N}{K}$ unordered subsets of K files from the set $\{W^j, j\in[N]\}$ that contains a certain file W^n and for each Υ_m there exists $K!$ different demand vectors $\mathbf{d}'$. Over all possible demand vectors $\mathbf{d}'(\mathbf{L})$ a subfile $W_\tau^n:|\tau|=i$ appears in the side information graph for each Υ_m the same number of times. For a fixed Υ_m, file W^n is requested by a user connected to any helper node with a certain cardinality $\mathcal{L}_r$. $Q_i(W_\tau^n)$ can be rewritten as

$$Q_i(W_\tau^n)=\binom{N-1}{K-1}\sum_{r=1}^{\Lambda}\sum_{\mathbf{d}'_r}\sum_{\mathbf{L}\in S_{\mathcal{L}}}\mathbb{1}_{\Omega_{\mathbf{d}'_r,\mathbf{L}}}(W_\tau^n)\tag{4.47}$$

where $\mathbf{d}'_r$ denotes the subset of all $K!$ demand vectors arising from Υ_m such that the index $n\in\mathbf{d}_{U_\lambda}:|\mathbf{d}_{U_\lambda}|=\mathcal{L}_r$. The number of chosen maximum acyclic subgraphs containing W_τ^n that arise from all the demand vectors $\mathbf{d}'_r(\mathbf{L})$ is evaluated as follows.

For the demands such that $n\in\mathbf{d}_{\mathcal{L}_r}$, W_τ^n appears in the side information graph $\mathcal{G}$ only if it is requested by a user connected to helper node λ such that $\lambda\notin\tau$ which corresponds to $(\Lambda-i)$ different *available* positions in the demand vector $\mathbf{d}'_r$ for $\mathbf{d}_{\mathcal{L}_\lambda}$, since $|\tau|=i$. Fixed one of the $(\Lambda-i)$ positions occupied by $\mathbf{d}_{\mathcal{L}_r}$, subfile W_τ^n will be picked depending on the position of the other $\mathbf{d}_{\mathcal{L}_j},\forall j\in[\Lambda]\setminus\{r\}$ into $\mathbf{d}$. For these remaining demands

$\mathbf{d}_{\mathcal{L}_j}, \forall j \in [\Lambda] \setminus \{r\}$ there are $P(\Lambda - i - 1, r - 1)(\Lambda - r)!$ possible ways to be placed into $\mathbf{d}$. Fixed the order of $\mathbf{d}_{\mathcal{L}_r}$ in $\mathbf{d}$ and $n \in \mathbf{d}_{U_\lambda} : L_{U_\lambda} = \mathcal{L}_r$, there are $\mathcal{L}_r$ different positions in $\mathbf{d}_{U_\lambda}$ in which we can place n. Then, it remains to fill $\mathcal{L}_r - 1$ positions of $\mathbf{d}_{U_\lambda}$ with $K - 1$ different numbers from the considered set $\Upsilon_m \backslash \{n\}$ and the remaining $K - \mathcal{L}_r$ positions in $\mathbf{d}'_r$ are filled with the remaining $K - \mathcal{L}_r$ numbers.

Therefore, there exist $\mathcal{L}_r P(K-1, \mathcal{L}_r - 1)(K - \mathcal{L}_r)!$ different demand vectors where the subfile W_τ^n will appear in the associated maximum acyclic subgraphs. Overall, we have

$$\begin{aligned} Q_i(W_\tau^n) = &\binom{N-1}{K-1} \sum_{r=1}^{\Lambda} P(\Lambda - i - 1, r - 1) \\ &\times (\Lambda - r)! \mathcal{L}_r P(K-1, \mathcal{L}_r - 1)(K - \mathcal{L}_r)!(\Lambda - i). \end{aligned} \tag{4.48}$$

which completes the proof. □

Since the value of $Q_i(W_\tau^n)$ does not depend on W_τ^n, we can change notation $Q_i \triangleq Q_i(W_\tau^n)$ and with Lemma 3, equation (4.44) becomes

$$\max_{\mathbf{d}} E_{\mathbf{L} \in S_{\mathcal{L}}} [\, T(\mathbf{L}, \mathbf{d}, \chi)] \geq \frac{1}{N_0} \sum_{i=0}^{\Lambda} \sum_{n \in [N]} \sum_{\tau \subseteq [\Lambda] : |\tau| = i} \frac{Q_i}{P(N, K) \Lambda!} |W_\tau^n| \tag{4.49}$$

$$\geq \frac{1}{N_0} \sum_{i=0}^{\Lambda} \frac{Q_i}{P(N, K) \Lambda!} x_i \tag{4.50}$$

where $x_i = \sum_{n \in [N]} \sum_{\tau \subseteq [\Lambda] : |\tau| = i} |W_\tau^n|, \; i \in [\Lambda] \cup \{0\}$ is the total size of the subfiles cached at exactly i helper nodes.

The coefficient of x_i in (4.50) is further simplified as in the following

$$\frac{Q_i}{\Lambda! P(N, K)} = \frac{1}{N} \sum_{r=1}^{\Lambda} \mathcal{L}_r \frac{\binom{\Lambda - r}{i}}{\binom{\Lambda}{i}} \tag{4.51}$$

Proof.

$$\begin{aligned}
&\frac{Q_i}{\Lambda! P(N,K)} \\
&= \frac{(N-1)!(N-K)!}{(K-1)!(N-K)!\Lambda!N!}\sum_{r=1}^{\Lambda}\mathcal{L}_r P(K-1,\mathcal{L}_r-1) \\
&(K-\mathcal{L}_r)!(\Lambda-i)P(\Lambda-i-1,r-1)(\Lambda-r)! \\
&= \frac{1}{(K-1)!\Lambda!N}\sum_{r=1}^{\Lambda}\mathcal{L}_r P(K-1,\mathcal{L}_r-1) \\
&(K-\mathcal{L}_r)!(\Lambda-i)P(\Lambda-i-1,r-1)(\Lambda-r)! \\
&= \frac{1}{(K-1)!\Lambda!N}\sum_{r=1}^{\Lambda}\mathcal{L}_r \\
&\frac{(K-1)!(K-\mathcal{L}_r)!(\Lambda-i)(\Lambda-i-1)!(\Lambda-r)!}{(K-\mathcal{L}_r)!(\Lambda-i-r)!} \\
&= \frac{1}{\Lambda!N}\sum_{r=1}^{\Lambda}\mathcal{L}_r\frac{(K-1)!(\Lambda-i)!(\Lambda-r)!}{(K-1)!(\Lambda-i-r)!} \\
&= \frac{1}{N}\sum_{r=1}^{\Lambda}\mathcal{L}_r\frac{(\Lambda-i)!(\Lambda-r)!i!}{\Lambda!(\Lambda-i-r)!i!} \\
&= \frac{1}{N}\sum_{r=1}^{\Lambda}\mathcal{L}_r\frac{\binom{\Lambda-r}{i}}{\binom{\Lambda}{i}}
\end{aligned} \tag{4.52}$$

□

Combining (4.50) and (4.51), we get

$$T(\boldsymbol{\mathcal{L}},\chi) \geq \frac{1}{N_0}\sum_{i=0}^{\Lambda}\frac{\sum_{r=1}^{\Lambda-i}\mathcal{L}_r\binom{\Lambda-r}{i}}{N\binom{\Lambda}{i}}x_i = \frac{1}{N_0}\sum_{i=0}^{\Lambda}\frac{x_i}{N}c_i \tag{4.53}$$

where $c_i \triangleq \frac{\sum_{r=1}^{\Lambda-i}\mathcal{L}_r\binom{\Lambda-r}{i}}{\binom{\Lambda}{i}}$ is a decreasing sequence whose lower convex envelope, $Conv(c_i)$, is decreasing and convex.

Due to the file size and cache size constraint, the following equalities hold for any caching starategy χ.

$$\sum_{i=0}^{\Lambda}x_i = N \tag{4.54}$$

$$\sum_{i=0}^{\Lambda}ix_i \leq Nt, \quad t\in\{0,1,...,\Lambda\} \tag{4.55}$$

In a similar fashion to [38, Proof of Lemma 2], combined (4.53) with (4.54) and (4.55) using Jensen's inequality and the monotonic decreasing property of $Conv(c_t)$, we obtain

$$T(\boldsymbol{\mathcal{L}}, \chi) \geq \frac{1}{N_0} Conv\left(\frac{\sum_{r=1}^{\Lambda-t} \mathcal{L}_r \binom{\Lambda-r}{t}}{\binom{\Lambda}{t}}\right). \tag{4.56}$$

Consequently,

$$T^*(\boldsymbol{\mathcal{L}}) = \min_{\chi} T(\boldsymbol{\mathcal{L}}, \chi) \geq \min_{\chi} \frac{1}{N_0} Conv\left(\frac{\sum_{r=1}^{\Lambda-t} \mathcal{L}_r \binom{\Lambda-r}{t}}{\binom{\Lambda}{t}}\right) \tag{4.57}$$

$$= \frac{1}{N_0} Conv\left(\frac{\sum_{r=1}^{\Lambda-t} \mathcal{L}_r \binom{\Lambda-r}{t}}{\binom{\Lambda}{t}}\right) \tag{4.58}$$

Finally, from equation (4.38) and (4.58) we have

$$T^* \geq E_{\boldsymbol{\mathcal{L}}}\left[T^*(\boldsymbol{\mathcal{L}})\right] \tag{4.59}$$

$$\geq E_{\boldsymbol{\mathcal{L}}}\left[\frac{1}{N_0} Conv\left(\frac{\sum_{r=1}^{\Lambda-t} \mathcal{L}_r \binom{\Lambda-r}{t}}{\binom{\Lambda}{t}}\right)\right] \tag{4.60}$$

This concludes the proof.

4.6 Conclusions

In this chapter we have studied the coded caching problem for a network comprised of a multi-antennas base station storing the library of files and K users, each one connected to one of $\Lambda < K$ cache-enabled helper nodes. Unlike other works, we did not put any constraint on the number of users connected to each helper node. We assumed that each user can fetch content from the associated helper node at zero cost. Assuming that the system operates in two phases (placement and delivery), our goal was to characterize the normalized delivery time required by the base station to serve all users, each requesting a different file from the library. To this end, we have developed a lower bound on the optimal delay and two different schemes for the single-antenna base station case and the multiple-antennas case, respectively. While the scheme for the single-antenna case can handle any user profile (vector whose elements represent the number of users connected to each helper node), the algorithm proposed for the multiple-antennas case can handle only a wide-but-limited set of user profiles. The performance achieved by the schemes clearly show how the delivery time is strongly affected by the way the users distribute among the helper nodes. It turns out, as expected, that the more "symmetric" is the user profile the lower is the required delay to serve all users. In fact, the lowest delay is experienced when the same number of users is connected to each helper node. Such a symmetric distribution of the users among the caches maximizes the multicast opportunities arising from coded caching. Moreover, for the symmetric user profile, it turns out that (at least for the user profiles handled by the algorithm), the multiplexing gain and the coded caching gain show in a multiplicative way. This result is of relevant interest because when there exist exactly $\Lambda = K$ caches, one for each user, the multiplexing gain is only additive to the coded caching gain. Finally, the delay achieved by the schemes matches with the developed lower bound. As a consequence, the proposed schemes are information theoretical optimal.

Bibliography

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[2] https://media.netflix.com/en/company-blog/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience

[3] L. A. Belady, "A study of replacement algorithms for a virtual-storage computer," in *IBM Systems Journal*, vol. 5, no. 2, pp. 78-101, 1966.

[4] N. Megiddo and D. S. Modha, "Outperforming LRU with an adaptive replacement cache algorithm", *Computer*, vol. 37, no. 4, pp. 58-65, 200.

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

[6] Kai Wan, D. Tuninetti and P. Piantanida, "On the optimality of uncoded cache placement," *2016 IEEE Information Theory Workshop (ITW)*, Cambridge, 2016, pp. 161-165.

[7] M. Ji, G. Caire and A. F. Molisch, "Fundamental Limits of Caching in Wireless D2D Networks," in *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 849-869, Feb. 2016.

[8] M. Ji, G. Caire and A. F. Molisch, "The Throughput-Outage Tradeoff of Wireless One-Hop Caching Networks," in *IEEE Transactions on Information Theory*, vol. 61, no. 12, pp. 6833-6859, Dec. 2015.

[9] F. Arbabjolfaei, B. Bandemer, Y. H. Kim, E. Sasoglu and L. Wang, "On the capacity region for index coding," *2013 IEEE International Symposium on Information Theory*, Istanbul, 2013, pp. 962-966.

[10] R. Pedarsani, M. A. Maddah-Ali and U. Niesen, "Online Coded Caching," in *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 836-845, April 2016.

[11] U. Niesen and M. A. Maddah-Ali, "Coded Caching With Nonuniform Demands," in *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146-1158, Feb. 2017.

[12] J. Zhang, X. Lin and X. Wang, "Coded Caching Under Arbitrary Popularity Distributions," in *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 349-366, Jan. 2018.

[13] M. A. Maddah-Ali and U. Niesen, "Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff," in *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029-1040, Aug. 2015.

[14] Q. Yan, M. Cheng, X. Tang and Q. Chen, "On the Placement Delivery Array Design for Centralized Coded Caching Scheme," in *IEEE Transactions on Information*

Theory, vol. 63, no. 9, pp. 5821-5833, Sept. 2017.

[15] K. Shanmugam, A. M. Tulino and A. G. Dimakis, "Coded caching with linear subpacketization is possible using Ruzsa-SzemÃ©redi graphs," *2017 IEEE International Symposium on Information Theory (ISIT)*, Aachen, 2017, pp. 1237-1241.

[16] C. Shangguan, Y. Zhang, G. Ge, "Centralized coded caching schemes: a hypergraph theoretical approach," *arXiv:1608.03989 [cs.IT]*, Aug 2016.

[17] S. P. Shariatpanahi, S. A. Motahari and B. H. Khalaj, "Multi-Server Coded Caching," in *IEEE Transactions on Information Theory*, vol. 62, no. 12, pp. 7253-7271, Dec. 2016.

[18] E. Lampiris, P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *arXiv:1802.03389 [cs.IT],* Feb 2018.

[19] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly. "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", *Euro. Conf. on Computer Systems (EuroSys)*, 2007.

[20] D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand. "Ciel: a universal execution engine for distributed data-flow computing," in *NSDI*, 2011.

[21] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Sixth USENIX Symposium on Operating System Design and Implementation,* Dec. 2004.

[22] O. O'Malley. "TeraByte Sort on Apache Hadoop", [Online]. Available: http://sortbenchmark.org/YahooHadoop.pdf, 2014.

[23] Kyuseok Shim, "MapReduce Algorithms for Big Data Analysis," *DNIS 2013*, LNCS 7813, pp. 44-48, 2013.

[24] Z. Zhang, L. Cherkasova and B. T. Loo, "Performance Modeling of MapReduce Jobs in Heterogeneous Cloud Environments," *2013 IEEE Sixth International Conference on Cloud Computing*, Santa Clara, CA, 2013, pp. 839-846.

[25] A. McKenna et al., "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data," *Genome research*, 2010.

[26] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication, ser. SIGCOMM '09*, New York, NY, USA: ACM, 2009, pp. 51-62.

[27] S. Zhang, J. Han, Z. Liu, K. Wang and S. Feng, "Accelerating MapReduce with Distributed Memory Cache," *2009 15th International Conference on Parallel and Distributed Systems,* Shenzhen, 2009, pp. 472-478.

[28] S. Li, M. A. Maddah-Ali and A. S. Avestimehr, "Coded MapReduce," *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, 2015, pp. 964-971.

[29] S. Li, M. A. Maddah-Ali, Q. Yu and A. S. Avestimehr, "A Fundamental Tradeoff Between Computation and Communication in Distributed Computing," in *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109-128, Jan. 2018.

[30] S. Li, S. Supittayapornpong, M. A. Maddah-Ali and S. Avestimehr, "Coded TeraSort," *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Lake Buena Vista, FL, 2017, pp. 389-398.

[31] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes," in *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514-1529, March 2018.

[32] J. Hachem, N. Karamchandani and S. N. Diggavi, "Coded Caching for Multi-level Popularity and Access," in *IEEE Transactions on Information Theory,* vol. 63, no. 5, pp. 3108-3141, May 2017.

[33] Z. Bar-Yossef, Y. Birk, T. S. Jayram and T. Kol, "Index Coding With Side Information," in *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1479-1494, March 2011.

[34] B. Asadi, L. Ong and S. J. Johnson, "The DoF Region of the Three-Receiver Gaussian MIMO Broadcast Channel With Receiver Message Side Information," in *IEEE Transactions on Communications,* vol. 65, no. 5, pp. 2000-2010, May 2017.

[35] J. Zhang, F. Engelmann and P. Elia, "Coded caching for reducing CSIT-feedback in wireless communications," *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, 2015, pp. 1099-1105.

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

[37] Y. Cao, M. Tao, F. Xu and K. Liu, "Fundamental Storage-Latency Tradeoff in Cache-Aided MIMO Interference Networks," in *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 5061-5076, Aug. 2017.

[38] Q. Yu, M. A. Maddah-Ali and A. S. Avestimehr, "The Exact Rate-Memory Tradeoff for Caching With Uncoded Prefetching," in *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281-1296, Feb. 2018.