Analysis of Complex Big Data Systems and Telecommunication Networks

Aristotle University of Thessaloniki

Sotirios K. Michos

May 2020

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation ().

Abstract

Coded caching is the distribution of content across a telecommunication system using techniques from coding theory in order to create multicasting opportunities among the users receiving the content. This enables a multiplicative improvement over the classic uncoded caching with respect to the transmission rates required in the delivery phase of the content. Since its introduction, coded caching has drawn significant research interest and several different schemes have been proposed over the last years. In this work, we focus on the fundamental case of coded caching with uncoded prefetching, where each user's cache is filled with uncoded content during a prefetching stage and, during a later delivery phase, each user's request must be served in the most efficient way. This important case has recently received a complete information-theoretic characterization. However, reaching the information-theoretic optimality imposes a significant computational imbalance among the users. To this end, we aim at mitigating this imbalance by first performing a complete computational analysis of the two major forms of coded caching with uncoded prefetching, namely *centralized* and *decentralized*, and then proposing a new method for the delivery phase that achieves a significant improvement compared to the state of the art.

Περίληψη

Τα συστήματα κωδικοποιημένης κρυφής μνήμης αναφέρονται στην διανομή περιεχομένου εντός ενός τηλεπικοινωνιακού συστήματος χρησιμοποιώντας τεχνικές από τη θεωρία κωδίκων με σκοπό να δημιουργηθούν ευκαιρίες πολύ-εκπομπής μεταξύ των χρηστών που λαμβάνουν το περιεχόμενο. Αυτό οδηγεί σε μια πολλαπλασιαστική βελτίωση σε σχέση με τα κλασσικά συστήματα μη-κωδικοποιημένης κρυφής μνήμης ως προς το ρυθμό εκπομπής που απαιτείται στη φάση παράδοσης του περιεχομένου. Από την ανάπτυξή του, τα συστήματα κωδικοποιημένης κρυφής μνήμης έχουν προσελκύσει σημαντικό ερευνητικό ενδιαφέρον και έχουν αναπτυχθεί πολλά διαφορετικά σχήματα στη διάρκεια των τελευταίων χρόνων. Σε αυτή την εργασία εστιάζουμε στη θεμελιώδη περίπτωση ενός συστήματος κωδικοποιημένης κρυφής μνήμης με μη-κωδικοποιημένη προανάκτηση, όπου στην κρυφή μνήμη κάθε χρήστη τοποθετείται μη-κωδικοποιημένο περιεχόμενο κατά τη διάρκεια της φάση προανάκτησης και, αργότερα, κατά τη φάση παράδοσης, το αίτημα κάθε χρήστη πρέπει να εξυπηρετηθεί με τον πιο αποδοτικό τρόπο. Αυτή η σημαντική περίπτωση έχει πρόσφατα δεχτεί έναν πλήρη πληροφοριοθεωρητικό χαρακτηρισμό. Όμως, η επίτευξη του πληροφοριοθεωρητικού βέλτιστου εισάγει μια σημαντική υπολογιστική ανισορροπία μεταξύ των χρηστών. Για το σκοπό αυτό, στοχεύουμε στη βελτίωση αυτής της ανισορροπίας αρχικά αναπτύσσοντας μια πλήρη υπολογιστική ανάλυση των δυο βασικών μορφών συστήματος κωδικοποιημένης κρυφής μνήμης, της κεντροποιημένης και της αποκεντροποιημένης μορφής, και στη συνέχεια προτείνοντας μια νέα μέθοδο για τη φάση παράδοσης η οποία πετυχαίνει σημαντική βελτίωση σε σχέση με τις μεθόδους που είναι διαθέσιμες στη βιβλιογραφία.

Dedication

This thesis is dedicated to my family and especially to my brother, whose struggles through life has been an inspiration for me.

Acknowledgements

This dissertation is the product of my research activity at the Department of Electrical and Computer Engineering of Aristotle University of Thessaloniki, during the years 2015-2020. Those five years were filled with a wonderful pursuit of knowledge, learning new and amazing things and having the chance to experience the joy of reaching concrete results after a long process of researching for them. Also, I had the opportunity to meet and work with wonderful people, whose enthusiasm further fueled my research interests and engagement with the process of pursuing a doctorate degree.

Firstly I would like to thank my advisor and mentor, prof. George Karagiannidis, Professor at the Department of Electrical and Computer Engineering of Aristotle University of Thessaloniki and leader of the Wireless Communications Systems Group (WCSG). His guidance, insights as well as vast experience contributed immensely in the formation of the final thesis' contents and results. However, his father-like understanding, patience, and support were of major importance for me and were the ones that allowed me to keep pursuing my doctorate degree, besides some quite important difficulties that I was also facing in my personal and family life. Furthermore, I would like to express my deepest gratitude towards my good friend and collaborator Dr. Vassilis Kapinas, for his supportive presence in my life and the wonderful work I had the opportunity to perform with him.

I would also like to thank the members of my advisory committee, prof. Leonidas Georgiadis, whose contribution and advice were of insurmountable importance and prof. Niovi Pavlidou for all her supportive comments and advice through my PhD preparation.

My best regards extend also to prof. Ioannis Antoniou, for allowing me to attend the Master's Program on Networks and Complexity, whose content contributed to the formation of my thesis' topic, as well as all the extremely talented members of the WCSG group I had the joy and the honor to meet and collaborate with, including the older Dr. Diomidis Michalopoulos, Dr. Athanasios Lioumpas, Dr. Korina Pappi, Dr. Georgia Ntouni, Dr. Alexandros Boulogiorgos and Dr. Dimitris Karas, all of whom are now pursuing very successful careers, as well as the newer ones Vassilis Papanikolaou, Sotiris Tegos, Stelios Trevlakis and Pavlos Bouzinis, with their PhD's topics covering a quite wide domain of discourse, which made the interaction with them quite stimulating and engaging.

I would also like to express my deepest gratitude to my family, my mother Nikoleta, my brother Nektarios as well as my late father Konstantinos whose constant support throughout my studies and life made this dissertation possible.

Last but not least, I would like to thank all my teachers and professors throughout my life that shaped my personality and interests as well as the Aristotle University of Thessaloniki itself, for all the support it provides its students.

Contents

Chapter 1

Introduction

The next generation of 5G communication networks faces a number of challenges imposed by the high requirements with respect to bandwidth and latency as well as the diverse ecosystem of applications and services that drive these requirements and the special characteristics of the physical medium that differ substantially from the ones in the past.

First of all, the use of radio frequencies up to 30GHz or even 60GHz, with the potential expansion to the whole range of mmWave communications (up to 300GHz) [1] introduces an unprecedented locality in the network due to the sheer amount of high atmospheric attenuation and increased fading present in these frequencies. This overhauls the traditional approaches of network design and enables a dramatic amount of frequency reuse, creating the chance for a really consumer-centric network. The classical paradigm of large cells that cover a big area with multiple users no longer applies and tends to be replaced by a large number of small cells each serving a few users in their close vicinity, such as fempto and pico cells and their evolution [2]. As a matter of fact, the actual networks are expected to be a hybrid of these two paradigms, having a multilevel structure of high heterogeneity composed of both small cells reliably serving the low-mobility users in their range with high bandwidth and low latency communications and large cells covering big areas with the goal to serve the high-mobility users. This multilevel structure introduces an additional complexity in the network both with respect to its control as well as the management of content generated and consumed by the individual users [3]. With the additional massive device-to-device communication that is expected to be implemented in these networks, they constitute a complex system in which the main challenge is to handle the big amount of generated or requested data in the most efficient way possible.

Applications such as virtual and augmented reality [4], autonomous vehicles [5] and smart cities [6] as well as the full spectrum of IoT applications [7] in the industry and the commercial sector impose a big data challenge for the network that should be able not only to process and deliver this data but also utilize it for its own optimization. The high bandwidth and ultra-reliable low latency requirement [5] create a need for pushing cloud computing towards the edge of the network so that the core network can offload the corresponding tasks and services keeping them in close proximity to the users, in order to increase the quality of its service.

These developments are expected to create a profound convergence between communication and computing in the form of fog computing enabled communication networks [8] or mobile edge computing [8] networks that will manifest in every abstraction layer of the design, implementation and operation phases of these networks.

During our bibliographic research, caching [9] in general, and coded caching [10] in particular, stood out as major enablers of the above technologies. Caching is the technique of duplicating content in distributed memories across a system with the goal of reducing the traf-

fic load and the service times whenever this content is requested. It is naturally comprised of two phases [10], the placement phase where the content is placed in the system caches, and the delivery phase where content requests are served. In essence, coded caching is the distribution of content across the system using techniques from coding theory in order to create multicasting opportunities among the users receiving the content. As we will explain in more detail in the corresponding chapter, this creates a multiplicative improvement over the classic uncoded caching approach with respect to the transmission rates required in the delivery phase of the content. This makes coded caching an essential candidate in order to harness the increased complexity of these systems and mitigate the big data challenges imposed by their specifications and modes of operation. What is more, there is a vast amount of coded caching variations [11–37] that enable its deployment in a wide range of abstraction levels and scenarios in and both central as well as edge nodes of the network, providing an exceptional asset to the increasingly challenging and complex data management requirements of the system. Further, the great ability of coded caching to be informed and self-adjust its operations based on results coming from big data analytics taking place on the network, make it an essential ingredient in the future 5G supported networks, like the Internet of Things, that are required to show robustness and adaptability, support emergence [38] and self-organization [39] among its parts, and in general, having the full spectrum of traits and behaviours [40] that are present in truly complex systems.

Conventional caching has a long line of research [41–48], where the main goal is to either maximize the hit rate, that is the probability that a requested content is found at the cache memory, or to optimize the placement of contents in the caches based on various criteria, most important of which being their popularity [49, 50].

Since its introduction, coded caching has drawn significant research interest and several different schemes have been proposed over the last years. In this work, we focus on the fundamental case of coded caching with uncoded prefetching, where each user's cache is filled with uncoded content during a prefetching stage and, during a later delivery phase, each user's request must be served in the most efficient way. This important case has recently received a complete information-theoretic characterization. However, reaching the information-theoretic optimality imposes a significant computational imbalance among the users. To this end, we aim at mitigating this imbalance by first performing a complete computational analysis of the two major forms of coded caching with uncoded prefetching, namely *centralized* and *decentralized*, and then proposing a new method for the delivery phase that achieves a significant improvement compared to the state of the art.

In their seminal paper [10] Maddah-Ali and Niesen proposed the use of coding in the placement and delivery phases in order to create simultaneous multicasting opportunities among the users that enabled a multiplicative gain in the transmission rates over conventional caching. One of the most striking characteristics of their approach is that it can offer this significant gain, even for scenarios where the content popularity is unknown or it is considered uniform. Due to its advantages, coded caching has attracted considerable research interest. More specifically, further research on this topic has been mainly focused on the investigation of its information-theoretic limits [51–56], as well as exploring the various forms it can take such as decentralized caching [11], online caching [12], hierarchical caching [13–15], D2D caching [16], caching with non-uniform demands [17–20], cache-aided interference channels [21–24] and others [25–37].

After a number of publications [10, 21, 51, 57, 58] that explored the optimal information-theoretic rate-memory tradeoff in coded caching with uncoded prefetching, an exact characterization for the cases of centralized and decentralized caching was provided in [59] by Yu et al. Centralized caching is the fundamental paradigm around which all other coded caching

schemes are developed, making any results regarding it being of principal importance. In this sense, [59] offers the potential of immediate improvement of all other relevant schemes of coded caching by extending the insights therein to them, like in [60, 61]. There has also been significant progress towards characterizing the exact rate-memory tradeoff for the case of coded prefetching [27, 52–56, 62, 63] with [64] setting the state of the art to within a factor of 2 with respect to the, as of yet unknown, optimal.

Along with these developments, research also turned towards investigating and mitigating the implementation challenges and limitations of wireless caching. To this direction, two major issues have been identified. The first issue is the combinatorial explosion that happens in coded caching, where the number of subfiles the files are broken into increases exponentially with respect to the number of users which quickly makes the subfile size corresponding to any finite file size fall below the single bit level [65]. To this end, several finite-file packetization schemes have been proposed [20, 61, 66–74] that try to contain this explosion while keeping the multiplicative gains of the original methods. Another important issue is the increased computational complexity of the information-theoretically optimal caching methods for the specific set of users called "non-leaders" [59]. In particular, their methods achieve information-theoretic optimality by utilizing the so-called commonality in user requests, which is the fact that many users may be requesting the same content. Among these users, one is arbitrarily called leader and the rest non-leaders. The authors of [59] realized that the computational manipulations a non-leader needs to perform impose a significant extra burden on them in comparison to the leaders and ask for a more efficient decoding scheme. Also, in [59] a motivating example is provided which shows that the computations performed by a non-leader can be potentially reduced. Nevertheless, to the best of the author's knowledge, an exact characterization of the computational cost of the existing methods, as well as a computationally improved method, has not been provided in the existing literature.

In this work, we aim to develop analytic expressions to precisely characterize the computational cost of coded caching with uncoded prefetching as well as to propose an improvement over this coded caching scheme with respect to the required computational resources.

In particular, we derive the computational cost of both centralized and decentralized caching with uncoded prefetching from a leader's and a non-leader's aspect as well as a system-wide point of view. With the computational manipulations being readily translatable to energy consumption, the same expressions provide a characterization of the schemes' energy demands.

Furthermore, we introduce a novel algorithm that directly improves the computational complexity of the state of the art by utilizing a shortcut in the computations performed by a non-leader. We apply this improvement to both centralized and decentralized versions of coded caching and observe some significant computational improvements, especially in the second more realistic case. Due to the multi-parameter dependence of the exact analytic expressions for the computational cost, we show that there is a particular tractable averaging procedure that facilities a meaningful comparison between the methods.

The remainder of this text is organized as follows. Section 2 describes the system model of centralized and decentralized caching with uncoded prefetching along with some major forms of coded caching and describes all the relevant concepts that we will utilize throughout the text. In section 3 we perform a computational analysis of centralized and decentralized caching, providing a complete characterization of the computational costs involved with the different parts of the system and observe that decentralized caching has a surprisingly simpler characterization when compared to centralized caching. We proceed with the development of our proposed method in section 4 for both centralized and decentralized caching and a comparison of its various aspects with the state of the art in section 5 using two cases, one small user case and one

large user case. The text closes with Section 6 containing our conclusions and some comments on future work.

Chapter 2

System Model and Preliminaries

2.1 Centralized Caching

A centralized caching system comprises of a server and K connected users through a shared, error-free channel. The server contains a library of N files $W_1, W_2, \ldots, W_N$ each of size F bits. Also, each user has an amount of cache memory equal to MF bits. Fig.2.1 displays, the archetypal centralized caching system [59].

The system operates in two phases, a placement and a delivery phase. During the placement phase, the users have free access to the library in order to fill their caches, without performing any coding to the content. During the delivery phase, each user makes a demand for a specific file to the server which, being the only one having access to the library, must deliver the requested content as efficiently as possible by utilizing the shared nature of the channel.

For the delivery phase, also called the prefetching phase, the authors in [59] utilize a scheme called symmetric batch prefetching. In the same paper, it is proved that this prefetching scheme, along with the proposed delivery method therein, constitutes an information-theoretically optimal way of delivering the content. Thus, for the rest of this work, will call the method presented in [59] the *information-theoretic optimal delivery method* or *ITODM* for short.

The key parameter of this algorithm is a quantity t equal to the ratio of the total cache memory among the users over the size of the library:

$$t = \frac{MK}{N}. \tag{2.1}$$

When this parameter is an integer, $t \in \{0, 1, \ldots, K\}$, the symmetric batch prefetching considers all the sets comprised of t users:

$$\mathcal{A}_t = \{A_t \in 2^{[K]} : |A_t| = t\}. \tag{2.2}$$

Here, $[K] = \{1, 2, ..., K\}$ is the user set, with each user represented by its unique index, $2^{[K]}$ is the user powerset, that is, all the possible user subsets and $|\cdot|$ is the cardinality of a set. The case where t is not an integer is typically handled through memory sharing. An exposition of how memory sharing can be applied is given in the section for hierarchical caching, where it is an integral part of the scheme.

If we call the contents of $\mathcal{A}_t$ "t-subsets", is it easy to see that there are

$$|\mathcal{A}_t| = \binom{K}{t} \tag{2.3}$$

t-subsets, equal to the number of ways we can choose t users out of K.

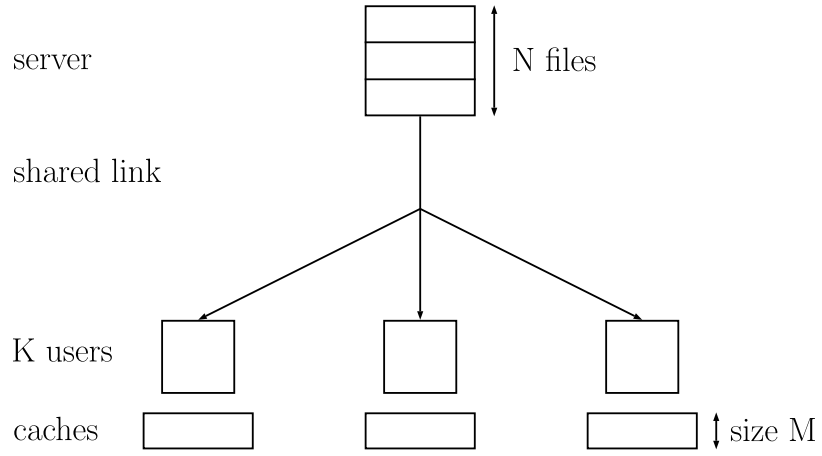


Figure 2.1: Centralized Caching System [59]. The server contains N files and connects through a shared, error-free channel to K users each having a cache size corresponding to M files.

Then, symmetric batch prefetching breaks up each file in $|\mathcal{A}_t|$ subfiles of size $F/\binom{K}{t}$ bit, with each subfile corresponding to a different t-subset, and sends each subfile to the users in that corresponding t-subset (for simplicity we assume that F is divisible by $\binom{K}{t}$, if not we can virtually expand the end of each file by appending a number zeros to suite this condition). By expressing the library as an $N \times F$ bit matrix, the matrix is broken into $|\mathcal{A}_t|$ columns of size $FN/\binom{K}{t}$ bit, each corresponding to at different t-subset $A_t \in \mathcal{A}_t$. Each column is then sent to all the users contained in its t-subset. Since there are $\binom{K-1}{t-1}$ subsets containing a user (as many as the $(t-1)$-subsets not containing that user), each user receives $\binom{K-1}{t-1}/\binom{K}{t}FN = MF$ bits fully filling up their cache.

During the delivery phase, each user makes a file request $d_i \in [N]$, where $[N] = \{1, 2, \ldots, N\}$. These requests form the demand vector $\mathbf{d} = (d_1, d_2, \ldots d_K) \in [N]^K$, on which the delivery algorithm operates. The delivery algorithm is based on two key observations. The first is that during the placement phase, all t-subsets $A_t \in \mathcal{A}_t$ receive a particular column of the library bit matrix. That means that for any $(t+1)$-subset $A_{t+1} \in \mathcal{A}_{t+1}$, each user $u \in A_{t+1}$ is requesting a particular line (i.e. a particular subfile) from the column the rest of the users in $A_{t+1}\backslash\{u\}$ have received during the placement phase. We can name this subfile $W_{d_u, A_{t+1}\backslash\{u\}}$. So, all user subfile demands that correspond to a particular $(t+1)$-subset can be served at once by XOR-ing the subfiles and transmitting the result:

$$Y_{A_{t+1}} = \underset{u \in A_{t+1}}{\oplus} W_{d_u, A_{t+1}\backslash\{u\}}. \tag{2.4}$$

The second observation is that when the number $N_e(\mathbf{d})$ of different files requested in $\mathbf{d}$ is less than the number K of users, not all transmissions $Y_{A_{t+1}}$ for all $(t+1)$-subsets $A_{t+1} \in \mathcal{A}_{t+1}$ need to take place. In particular, it is the realization that by arbitrarily selecting $N_e(\mathbf{d})$ users $\mathcal{U} \subset [K]$, with distinct requests and calling them leaders, any transmission $Y_{A_{t+1}}$ corresponding to a $(t+1)$-subset A_{t+1} comprised solely of non-leaders is redundant. If we call $\mathcal{A}_{t+1}^{nl}$ the family of all $(t+1)$-subsets comprised solely of non-leaders, the above means that, in order to have some profit from the commonality among the user requests, there must be at least one $(t+1)$-subset in $\mathcal{A}_{t+1}^{nl}$. In other words, we must have:

$$K - N_e(\mathbf{d}) \geq t + 1. \tag{2.5}$$

The converse proof given in [59] shows that this condition is fundamental. Combining these two observations, one reaches the conclusion that among all $\binom{K}{t+1}$ $(t+1)$-subsets, only $\binom{K}{t+1} - \binom{K-N_e(\mathbf{d})}{t+1}$ correspond to actual transmissions. Since each transmission is $F/\binom{K}{t}$ bit long a rate of

$$R(M,N,K) = \frac{\binom{K}{t+1} - \binom{K-N_e(\mathbf{d})}{t+1}}{\binom{K}{t}} \tag{2.6}$$

is achievable. Notice that F is not included in (2.6), since the rate corresponds to transmitted bits per file bit. Also, we should note that although the quantity R is called rate in the literature, it is, in fact, the transmission load on the channel during the delivery phase. In other words, the channel should be able to support a transmission with the rate implied by R or above. In this sense, we are interested in devising delivery schemes that make this rate as small as possible. It has been proven [59] that (2.6) represents the best possible rate for a centralized coded caching system with uncoded prefetching. The term centralized means that during the placement phase, we know the K users that will be requesting a file.

In the delivery method presented in [59], each omitted transmission $Y_{A_{t+1}}$ corresponding to a non-leader $(t+1)$-subset $A_{t+1} \in \mathcal{A}_{t+1}^{nl}$ is computed from the transmitted signals. This computation is based on two things. First, a set $\mathcal{B}$ formed by the union of the leader set $\mathcal{U}$ and the non-leaders in A_{t+1}:

$$\mathcal{B} = \mathcal{U} \cup A_{t+1}. \tag{2.7}$$

The second thing is the family $\mathcal{V}^F$ of $\mathcal{V}$-subsets of $\mathcal{B}$, where each $\mathcal{V}$-set contains $N_e(\mathbf{d})$ users of $\mathcal{B}$ with distinct demands.

$$\mathcal{V}^F = \left\{ \mathcal{V} \in 2^{\mathcal{B}} : \begin{array}{ll} |\mathcal{V}| = N_e(\mathbf{d}), & \\ \forall u_1 \neq u_2 \in \mathcal{V} & d_{u_1} \neq d_{u_2} \end{array} \right\}, \tag{2.8}$$

where $2^{\mathcal{B}}$ is the powerset of $\mathcal{B}$. These $\mathcal{V}$-sets can be generated by starting with $\mathcal{U}$ and then replacing one or more leaders by one of their non-leaders (having the same request) in $\mathcal{B}$. Having these two things, the authors of [59] prove that the signal $Y_{A_{t+1}}$ can be computed as

$$Y_{A_{t+1}} = \bigoplus_{\mathcal{V} \in \mathcal{V}^F \setminus \{\mathcal{U}\}} Y_{\mathcal{B}\setminus\mathcal{V}}. \tag{2.9}$$

This computation is the main focus of this work. In particular, we propose an alternative expression of equal computational cost that instead of generating $Y_{A_{t+1}}$, it directly gives the file requested by each user in A_{t+1}.

2.2 Decentralized Caching

The decentralized caching, represents the more realistic scenario were the number of users that will be requesting a file during the delivery phase is not known during the placement phase. In other words, the archetypal systems presented in Fig. 2.2 is the same as the one in section 2.1 with the difference that the K users are not known at the placement phase. As it is shown in [59], assuming a system with a library of N files, each of size F bit, a random caching of MF/N bits from each file by each user during the placement phase is enough to guarantee optimality during the delivery phase. We do not mention the total number of users since this does not matter during this phase.

Suppose that during the delivery phase only K users actually make a request $\mathbf{d} = (d_1, \ldots, d_K)$. We can denote the set of these active users as $[K] = \{1, 2, ..., K\}$, representing each one of them

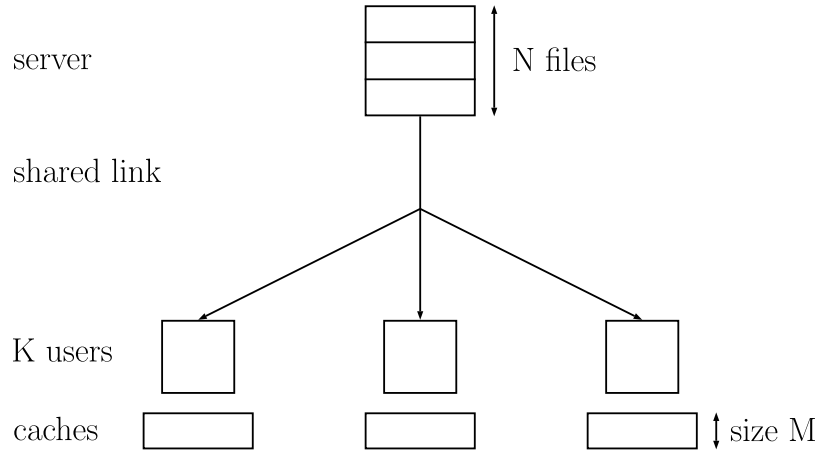


Figure 2.2: Decentralized Caching System [59]. The server contains N files and connects through a shared, error-free channel to K users each having a cache size corresponding to M files. The K users are coming from an even larger pool of users and are unknown during the placement phase.

by a suitable index. Calling the library bits $\mathcal{B}^L$, if we examine the way they were transferred to these users during the random prefetching, we will see that they can be organized (partitioned) into the following classes

$$\mathcal{B}_j^L = \left\{ b \in \mathcal{B}^L : \begin{array}{l} b \text{ is cached by exactly} \\ j \text{ users among } [K] \end{array} \right\}, \tag{2.10}$$

for $j = 0, 1, \ldots, K$. This collection forms a partition since any bit in $\mathcal{B}^L$ can be cached either by no user or exactly 1 user or exactly 2 users etc. up to exactly K users. Each of these classes can be further partitioned if we ask the question which are the particular j users caching a bit of $\mathcal{B}_j^L$. If $A_j \in \mathcal{A}_j$ is any j-subset of the K users, this leads to the following partitioning within each $\mathcal{B}_j^L$:

$$\mathcal{B}_{A_j}^{L,j} = \left\{ b \in \mathcal{B}_j^L : \text{bit } b \text{ is shared among all users in } A_j \right\}. \tag{2.11}$$

We can see now that for each $j \in \{0, \ldots, K\}$, the situation is analogous to the case of centralized caching with a file library $\mathcal{B}_j^L$ being broken to $\binom{K}{j}$ parts, each corresponding to a distinct j-subset and being shared among the j users therein. That means that for any $(j+1)$-subset $A_{j+1} \in \mathcal{A}_{j+1}$ each user $u \in A_{j+1}$ will be requesting the bits of file W_{d_u} shared among the other $u' \in A_{j+1} \backslash \{u\}$ users.

So, for each such $(j+1)$-subset $A_{j+1} \in \mathcal{A}_{j+1}$, the server can transmit the signal

$$Y_{A_{j+1}} = \underset{u \in A_{j+1}}{\oplus} V_{d_u, A_{t+1} \backslash \{u\}} \tag{2.12}$$

to accommodate the users in A_{j+1}. The quantity $V_{d_u, A_{j+1} \backslash \{u\}}$ contains the bits of file W_{d_u} shared exactly by the j users in $A_{t+1} \backslash \{u\}$. A difference from centralized caching is that the size of each $V_{d_u, A_{j+1} \backslash \{u\}}$ in (2.12) will be different due to the prefetching being random and so a padding of the smaller terms with zeros can make the expression computable. Also, we should note that both the server and each user should be aware of the bit positions being cached by the other users in order for the signal in (2.12) to be computable and decodable, so some kind of synchronization between the users and the server should take place.

As in centralized caching, if the number of distinct demands $N_e(\mathbf{d}) \leq K-(j+1)$, the K users can be split to "leaders" and "non-leaders" and any of the $\binom{K-Ne(\mathbf{d})}{j+1}$ signals $Y_{A_{j+1}}$ corresponding to the non-leader $(j+1)$-subsets $A_{j+1} \in A_{j+1}^{nl}$ can be omitted from transmission, being derivable from the ones transmitted.

Since, the probability of a bit being selected by a user in the placement phase is M/N, the probability of a single bit being exclusively selected by the users of a j-subset A_j of $[K]$ (and thus being placed in $\mathcal{B}_{A_j}^{L,j}$) is $P(\mathcal{B}_{A_j}^{L,j}) = (M/N)^j(1-M/N)^{K-j}$ and the probability of a single bit being selected exclusively by any such j subset (and thus being placed in $\mathcal{B}_j^L$) is $P(\mathcal{B}_j^L) = \binom{K}{j}(M/N)^j(1-M/N)^{K-j}$. Iterating over all the F bits in a file W, the number $|\mathcal{B}_{A_j}^{L,j}(W)|$ of bits being placed in $\mathcal{B}_{A_j}^{L,j}$ and the number of bits $|\mathcal{B}_j^L(W)|$ being placed in $\mathcal{B}_j^L$ will be binomial random variables $\mathcal{B}(n,p)$ with parameters $n=F$ and $p=P(\mathcal{B}_{A_j}^{L,j})$ or $p=P(\mathcal{B}_j^L)$, respectively.

We can now use Bernoulli's theorem [75] which states that if k is a binomial random variable $\mathcal{B}(n,p)$ then for any $\epsilon > 0$

$$P\left(\left|\frac{k}{n}-p\right| \leq \epsilon\right) > 1 - \frac{pq}{n\epsilon^2} \geq 1 - \frac{p}{n\epsilon^2}. \tag{2.13}$$

Bernoulli's theorem states that we can get k/n as close to p as we would like with probability as close to one as we would like, as long as n is made high enough. In other words, if we decompose the random variable k as $k = np + e$, the ratio e/n can be made arbitrarily small with probability arbitrarily close to one, as long as n is sufficiently high. In simple words, we can write that asymptotically $k = np + o(n)$ almost surely.

Using the above, we can write that asymptotically almost surely (a.a.s) we will have:

$$\begin{aligned} |\mathcal{B}_j^L(W)| &= \binom{K}{j}\left(\frac{M}{N}\right)^j\left(1-\frac{M}{N}\right)^{K-j}F + o(F), \\ |\mathcal{B}_{A_j}^{L,j}(W)| &= \left(\frac{M}{N}\right)^j\left(1-\frac{M}{N}\right)^{K-j}F + o(F). \end{aligned} \tag{2.14}$$

which are the expressions presented in [59] with the first being $\binom{K}{j}$ times bigger than the second, as expected. So, we see that asymptotically, for any j the delivery phase of decentralized caching can be implemented by the same techniques used in centralized caching.

Since the bits comprising $V_{d_u,A_{t+1}\setminus\{u\}}$ in (2.12) are the ones contained in $\mathcal{B}_{A_{j+1}\setminus\{u\}}^{L,j}(W_{du})$, each transmission $Y_{A_{j+1}}$ will be $|\mathcal{B}_{A_j}^{L,j}(W)|$ bits long (a.a.s.) as given by (2.14). Multiplying this with the number $\binom{K}{j+1} - \binom{K-N_e(\mathbf{d})}{j+1}$ of actual transmissions, summing over all j and dividing by F gives a rate of

$$R(M,N,K) = \frac{N-M}{N}\left(1-\left(1-\frac{M}{N}\right)^{N_e(\mathbf{d})}\right). \tag{2.15}$$

So, we observe that, even though the prefetching is random and the delivery phase transmissions are of varying size, in the limit of large file sizes F an order arises which allows us to explicitly compute the transmission load per file bit. We will utilize this order to compute the computational benefits of our proposed decoding method for decentralized caching.

2.3 Hierarchical Caching

Many real systems and networks are comprised of many layers of abstraction and a high degree of heterogeneity expressed as a hierarchy of nodes ranging from central, core-wise nodes to

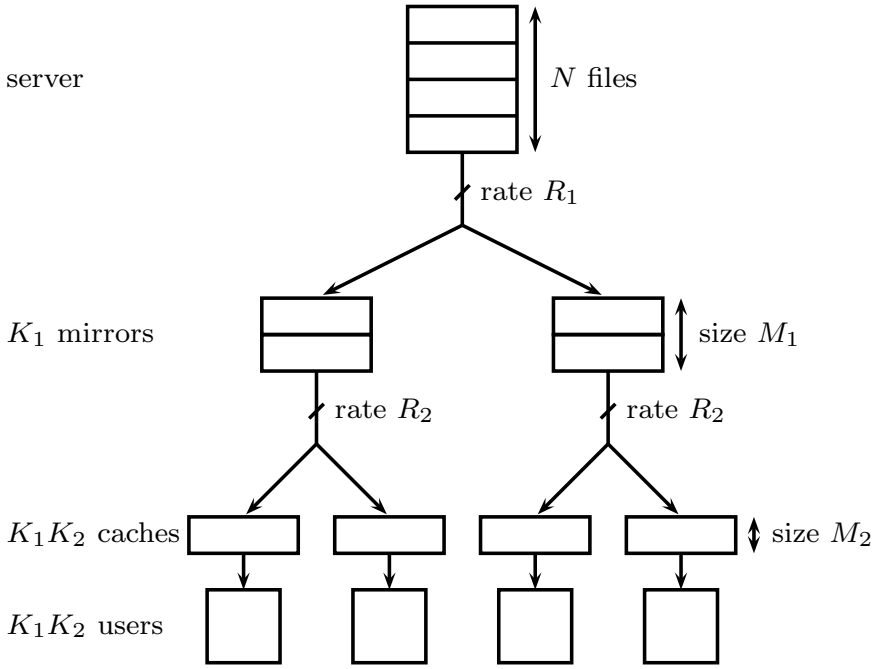


Figure 2.3: Hierarchical Caching System [14]. The system comprises of two levels of caching. The server contains N files of size F bit and connects through a shared, error-free channel to K_1 intermediate nodes (mirrors) with cache memory size M_1F bit. Each mirror then serves K_2 users, each having a cache memory size of M_2F bit, through a another, shared, error-free channel.

peripheral, edge-wise nodes. This creates an opportunity for concurrently utilizing caching in many layers and across different levels of the system. Hierarchical caching [13–15] was developed to account for these arrangements and explores how to optimally distribute and deliver cachable content in heterogeneous networks with a hierarchical structure.

The archetypal hierarchical caching system [14] is displayed in Fig. 2.3. In this system, the server has a library of N files, each of size F bit. The server connects through a shared, error-free channel to K_1 intermediate nodes with cache memory size M_1F bit, called mirrors. Each mirror has a cache memory size of M_1F bit and serves a separate group of K_2 users, each having a cache memory size of M_2F bit, through another shared, error-free channel.

As of the time of this writing, the author is not aware of an optimal caching scheme for this system being available. However, [14] describes a near-optimal placement and delivery method based on decentralized caching.

According to [14], the placement and delivery method is the result of memory sharing between two different schemes. In the first scheme, the placement of content in the mirrors and the users follows the simple random placement approach we described in section 2.2 for decentralized caching. During delivery, every mirror presents to the server the demands of its users in a sequential manner. In the first step, each mirror presents the demand of its first user, in the second the demand of its second user etc. up to the final step presenting the demand of its K_2-th user. In each step, the files are sent to the mirrors using the decentralized delivery scheme also presented in section 2.2, without utilizing the commonality between the demands (sending all transmission corresponding to non-leader). After all the mirrors have acquired and decoded the files requested by their individual users, each mirror uses the decentralized delivery method (without utilizing commonality) to send these files to their users.

The rate R_1^A between the server and the mirrors and the rate R_2^A between each mirror and its users, therefore, are

$$R_1^1 = K_2 \cdot r\left(M_1, N, K_1\right) \tag{2.16}$$

and

$$R_2^1 = r\left(M_2, N, K_2\right) \tag{2.17}$$

where R(M,N,K) is the achievable rate of decentralized caching without utilizing commonality among the demands, which for the typical case of $N > K$ is

$$r(M,N,K) = \begin{cases} K \cdot (1 - M/N) \cdot \min\left\{\frac{N}{KM}\left(1-(1-M/N)^K\right), \frac{N}{K}\right\} & \text{for } 0 < M \leq N \\ \min\{N, K\} & \text{for } M = 0 \\ 0 & \text{for } M > N \end{cases} \tag{2.18}$$

In the second scheme, the cache memory of the mirrors is overlooked and the mirrors act as relays. During the placement phase, the caches of the K_1K_2 users are filled using random placement as before. During the delivery phase, the server receives the demands of all users and transmits all the corresponding codewords according to the decentralized delivery method (without commonality) and each mirror just relays the codeword relevant to its user group.

Thus, the achievable rates R_1^B and R_2^B between the server and the mirrors and each mirror to its users respectively are

$$R_1^2 = r\left(M_2, N, K_1K_2\right), \tag{2.19}$$

and

$$R_2^2 = r\left(M_2, N, K_2\right). \tag{2.20}$$

One important remark that we should make here is the dependence of hierarchical caching on decentralized caching. As we explained before, the quantity r in the above equations is the rate of the decentralized caching scheme used. Thus, any improvement of the decentralized caching leads to a direct improvement of hierarchical caching. As a matter of fact, the utilization of commonality between user requests, a technique not available to the authors of [14], enhances the rate from that of (2.18) to the one given by (2.15). Should the authors of [14] used centralized instead of decentralized caching, the utilization of commonality would lead to a similar enhancement of the achievable rates. As we discussed in the corresponding sections, the utilization of commonality leads to the information-theoretically optimal scheme for centralized or decentralized caching. However, at the time of writing, it is not known whether this enhancement leads hierarchical caching to a similar optimality or just an improvement.

Completing the description of the general hierarchical caching scheme, a memory sharing between the two schemes is performed. In particular, each file in the library and the cache memory of each user is separated into two parts. In the first part, we have

$$\begin{aligned} F^1 &= \alpha F, \\ M_1^1 &= \frac{M_1 F}{F^1} = \frac{M_1}{\alpha}, \\ M_2^1 &= \frac{\beta M_2 F}{F^1} = \frac{\beta M_2}{\alpha}, \end{aligned} \tag{2.21}$$

for some $(\alpha, \beta) \in [0,1]^2$ and in the second part

$$\begin{aligned} F^2 &= (1-\alpha)F, \\ M_2^2 &= \frac{(1-\beta)M_2F}{F^2} = \frac{(1-\beta).M_2}{(1-\alpha)}. \end{aligned} \tag{2.22}$$

Then, in each part, we use the corresponding caching scheme to deliver the data. Note that since the cache memory of the mirrors is utilized only in the first scheme, it is completely dedicated to the first part leading to a higher number M_1^1 of cachable part-1 files ($M_1^1 > M_1$).

This memory sharing makes the achievable rates equal to

$$\begin{aligned} R_1^1 &= \alpha K_2 \cdot r\left(M_1^1, N, K_1\right) = \alpha K_2 \cdot r\left(\frac{M_1}{\alpha}, N, K_1\right), \\ R_2^1 &= \alpha \cdot r\left(M_2^1, N, K_2\right) = \alpha \cdot r\left(\frac{\beta M_2}{\alpha}, N, K_2\right). \end{aligned} \tag{2.23}$$

The selection of α and β then is a matter of optimization. The authors of [14] outline three regimes

$$\begin{aligned} &\text{I) } M_1 + M_2K_2 \geq N \text{ and } 0 \leq M_1 \leq N/4 \\ &\text{II) } M_1 + M_2K_2 < N, \\ &\text{III) } M_1 + M_2K_2 \geq N \text{ and } N/4 < M_1 \leq N, \end{aligned} \tag{2.24}$$

and propose the values

$$(\alpha^\star, \beta^\star) = \begin{cases} \left(\frac{M_1}{N}, \frac{M_1}{N}\right) & \text{in regime I,} \\ \left(\frac{M_1}{M_1+M_2K_2}, 0\right) & \text{in regime II,} \\ \left(\frac{M_1}{N}, \frac{1}{4}\right) & \text{in regime III.} \end{cases} \tag{2.25}$$

The corresponding rate values $R_1(\alpha^\star, \beta^\star)$ and $R_2(\alpha^\star, \beta^\star)$ are proven to be within a constant multiplicative and additive gap with respect to their minimum values (or to corresponding lower bounds to be more exact). This places the above scheme very close to optimality. In particular, it shows that if

$$\mathcal{R}^\star\left(M_1, M_2\right) = \text{closure}\left\{(R_1, R_2) : (M_1, M_2, R_1, R_2) \text{ is feasible }\right\} \tag{2.26}$$

,

is the information-theoretic feasible rate region for a particular choice of M_1 and M_2 and

$$\mathcal{R}_C\left(M_1, M_2\right) = \left\{(R_1(\alpha, \beta), R_2(\alpha, \beta)) : \alpha, \beta \in [0, 1]\right\} + \mathbb{R}_+^2 \tag{2.27}$$

is the achievable rate region using the above scheme then there are constants c_1 and c_2, independent of M_1 and M_2, such that

$$\mathcal{R}_C\left(M_1, M_2\right) \subseteq \mathcal{R}^\star\left(M_1, M_2\right) \subseteq c_1 \cdot \mathcal{R}_C\left(M_1, M_2\right) - c_2 \tag{2.28}$$

We should note that in (2.27) the addition sign is the Minknowski addition between the two sets and it is utilized in order to extend the region achievable by R_1 and R_2 for the various α and β to the larger range values that are also achievable (for example by zero padding).

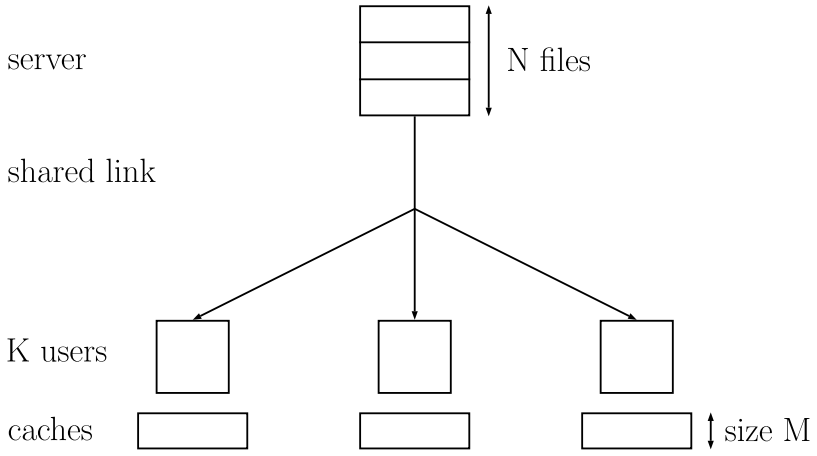


Figure 2.4: Caching with non-Uniform Demands [17]. The server contains N files that follow a specific popularity distribution and connects through a shared, error-free channel to K users each having a cache size corresponding to M files.

2.4 Caching with non-Uniform Demands

As we describe in the introduction, big data analytics on the telecommunication systems' data can reveal crucial information about the system. On such kind of information is the popularity of the various contents that move around the network. Deriving such popularity distributions is crucial to optimize the behavior of the system and properly allocate the available resources. Caching with non-uniform demands was developed with the specific goal to take this information into account and optimally adjust the placement and delivery phases of the caching system in order to minimize the rate of delivery.

There have been many attempts and approaches to characterize caching with non-uniform demands such as [17–20] with varying results. In this text, we will present the method developed in [17] for illustration purposes as well as for highlighting the relation of this kind of caching to the other types.

The archetypal system in Fig. 2.4 is again the one presented in section 2.2. In this system, the server has N files of size F bit with each file having a specific popularity score. Without loss of generality, we assume that the files are named in a decreasing order of popularity, with the first file having popularity p_1, the second $p_2 \leq p_1$, the third $p_3 \leq p_2$ etc. The different scores sum to 1 and thus constitute a probabilistic distribution, called popularity distribution in this context. Each of the users is connected to the server through a shared and error-free channel.

To accommodate for the differences in popularity scores the files as grouped into groups of "similar" popularity. Similar means that the least popular file in each group has no less than half the popularity of the most popular file in the group. For example, the first group $\mathcal{N}_1$ will have all the files from 1 to N_1 such that $p_{N_1} \geq p_1$ and $p_{N_1+1} < p_1$. File $N_l + 1$ will thus be the first file of the second group $\mathcal{N}_2$ and so on forming L groups. The authors of [17] call this N_1,N_2,...,N_L grouping a maximal partition of the files within a popularity factor of 2. Of course, one can use a different factor or leave the choice of groups to be a matter of some optimization technique.

During the placement phase, a fraction of each user's cache memory $M_l F$ is allocated to each group $\mathcal{N}_l$ and the user randomly caches $M_l F/N_l$ bits from each file in the group $\mathcal{N}_1$. Note that the memory factions M_l must sum to M so that each user's cache memory is filled.

In the delivery phase, the users are similarly grouped. All the users requesting a file from group $\mathcal{N}_1$ form the user group $\mathcal{K}_1$, all the users requesting a file from group $\mathcal{N}_2$ form the user group $\mathcal{N}_2$ etc. Since the user choices are random, the group cardinalities K_1, K_2, ... ,K_L will also be random variables. Then, the server uses the decentralized delivery scheme we have presented in section 2.2 to deliver the requested files for each separate group.

The achievable rate of this procedure will be

$$R = \sum_{l=1}^{L} R\left(M_l, N_l, K_l\right), \tag{2.29}$$

where $R\left(M_l, N_l, K_l\right)$ is the peak rate of the decentralized delivery method used for each group (see the end of this section for a discussion on its actual value).

Due to the randomness of the user groups, the figure of merit in this scheme cannot be the one in (2.29) but rather the expected value of it with respect to the user group cardinalities.

$$\mathbb{E}(R) = \sum_{l=1}^{L} \mathbb{E}\left(R\left(M_l, N_l, K_l\right)\right), \tag{2.30}$$

where

$$\mathbb{E}\left(R\left(M_l, N_l, k\right)\right) = \sum_{k=0}^{K} R\left(M_l, N_l, K_l = k\right) P\left(K_l\right). \tag{2.31}$$

In the above $P(K_l)$ is the probability that the cardinality of the l-th user group is k. This probability depends on the user's random choices which ultimately follow the popularity distribution. The authors of [17] prove that for the information-theoretic optimal rate $R^{\star}\left(M, \mathcal{N}, K, \{p_n\}\right)$ of this scheme holds that

$$\begin{aligned} \frac{1}{cL} \sum_{\ell=1}^{L} \mathbb{E}\left(r\left(M, N_\ell, \mathrm{K}_\ell\right)\right) \leq R^{\star}\left(M, \mathcal{N}, K, \{p_n\}\right) &\leq \min_{\{M_\ell\}: \sum_\ell M_\ell = M} \sum_{\ell=1}^{L} \mathbb{E}\left(r\left(M_\ell, N_\ell, \mathrm{K}_\ell\right)\right) \\ &\leq \sum_{\ell=1}^{L} \mathbb{E}\left(r\left(M/L, N_\ell, \mathrm{K}_\ell\right)\right) \end{aligned} \tag{2.32}$$

In other words, the optimal rate is upper bounded by the expected rate of the proposed scheme minimized among the different options for M_1,...,M_L for the user's cache memory segmentation which is further upper bounded by the expected rate that corresponds to a uniform user cache memory segmentation where $M_1 = ... = M_L = M/L$. Furthermore, there is a constant c independent from the system parameters such that the optimal rate is lower bounded by $1/cL$ the aforementioned upper bound. So we see, that for a specific popularity distribution, the optimum rate is within a constant multiplicative gap of this scheme's achievable rate.

We would like to stress the dependence of this caching scheme to the decentralized caching. The quantity $R\left(M_l, N_l, K_l\right)$ used in the above expressions is the peak rate of the decentralized delivery method. This shows that any improvement of the later directly induces an improvement to the former, in a way similar to hierarchical caching. In particular, since the option to utilize commonality was not available at the time of Niesen's et al. paper, the authors of [17] give the expression

$$r(M,N,K)=\begin{cases} K\cdot(1-M/N)\cdot\min\left\{\frac{N}{KM}\left(1-(1-M/N)^K\right),\frac{N}{K}\right\} & \text{for } 0<M\leq N \\ \min\{N,K\} & \text{for } M=0 \\ 0 & \text{for } M>N \end{cases} \tag{2.33}$$

However, after utilizing commonality among the user requests, this rate can improve to the one we saw in section 2.2. What is more, assuming that the optimal file grouping is used, the average rate cannot be further improved since the utilization of commonality achieves optimality for decentralized caching. However, the optimal caching scheme for non-uniform demands remains an open problem at the time of this writing.

We should also note that the proofs of the previous bounding results have not been formally extended to the utilization of commonality, but there is no obvious reason for them not to hold. Instead, utilizing commonality is expected to provide a tightening of these bounds.

2.5 Device-to-Device Caching

Machine type networks are expected to play an ever-increasing role in 5G communication networks. Device-to-device (D2D) caching [16] was developed in order to enable the benefits of coded caching in such distributed settings as a D2D network. D2D networks are systems where the communication devices engage in direct communication with each other, without depending on a central base station to handle the exchange of information. Such spatial networks can exhibit highly complex behaviors that the network itself must be able to support by self-organizing its operation. However, in order to illustrate the concept of D2D caching, we will base our description in a rather simple setting. Nevertheless, the main ideas that we will describe here can be extended in more general settings where user mobility is also taken into account.

The D2D caching archetypal system we will describe is shown in Fig. 2.5. It comprises of K nodes arranged in the unit square grid and each node is at a distance of $1/\sqrt{K}$ from its horizontal and vertical neighbors and can transmit in a unit disk of radius r. Any nodes within this disk can receive its transmission, as long as they are $(1+\Delta)r$ away from other transmitting nodes. For our simple illustrating example, we assume that $r>\sqrt{2}$ so only one node can transmit at any given time.

As usual, there is a library of N files of size F, and these files will be distributed among the K nodes, each having a cache memory size of MF bit. Again, the key parameter here is the ratio of the total cache memory over the library size $t=\frac{MK}{N}$. Assuming that t is an integer, during the placement phase each file is broken up into $\binom{K}{t}$ subfiles, with each different subfile of each file corresponding to a different subset $A_t\in 2^{[}K]$of t nodes ($|A_t|=t$). If t is not an integer, we can apply memory sharing.

Now, each of the $\binom{K}{t}$ subfiles is further broken into t packets. All the packets of a particular subfile are then placed into the caches of the nodes in the corresponding A_t subset. The procedure is exactly the same as the one we described for the placement phase of centralized caching, with the only difference that each subfile is further broken into t packets.

During the delivery phase, each node requests a particular file. The placement phase has guaranteed that for each set of $t+1$ nodes, $A_{t+1}\in 2^{[}K]$ with $|A_{t+1}|=t+1$, each node has a subfile (with all its t packets) that another node requires and, what is more, all the other $t-1$ nodes have the same information as well (the same t packets). So a transmission of a

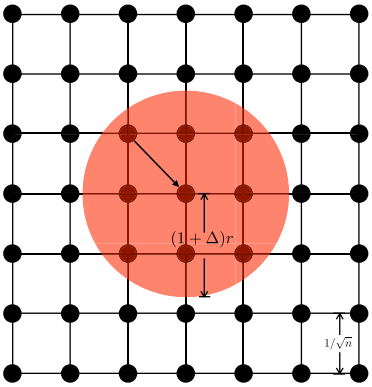


Figure 2.5: Device-to-device (D2D) Caching [16]. The K devices are the nodes in a grid covering the unit square. The distance between neighboring horizontal and vertical nodes is $1/\sqrt{K}$. Each node transmits in a disk of radius r and in order to receive from a transmitting node it must be at least $(1+\Delta)r$ away from other transmitting nodes. In our system $r > \sqrt{2}$ so only one node is allowed to transmit at any given time. The library is comprised of N files, each of size F and each node has a cache memory of MF bit.

suitable xOR among the packets that each node has is enough for all the nodes to acquire their requested files.

To illustrate how these transmission is formed for each A_{t+1}, let us explain the transmission of a particular node $u \in A_{t+1}$. For each other node $v \in A_{t+1}\backslash\{u\}$ with $v \neq u$, the placement phase has guaranteed that all the nodes in $A_{t+1}\backslash\{v\}$ have a subfile that v is requesting (all its t packets). Remembering that we have named all the K nodes using numbers $1, 2, ..., K$ node u can observe its position in the set $A_{t+1}\backslash\{v\}$ arranged in an ascending manner. This position can be one of $1, 2, ..., t$ since there are exactly t nodes present in $A_{t+1}\backslash\{v\}$. So node u, among all packets that user v is requesting, can select the one corresponding to its position. Selecting in this manner one packet for each other node in A_{t+1} and xOR-ing them constitutes its transmission for the particular A_{t+1}.

So we see that the delivery method is, in essence, the same as the one for centralized caching, with the only difference being that for each A_{t+1} instead of having a server making a massive transmission, we have the nodes in A_{t+1} transmitting and exchanging the corresponding packets themselves. The second difference is that in each node transmission, instead of having the node xOR-ing and transmitting a whole subfile, we have it transmit just a packet that corresponds to the $1/t$ of the subfile. In this way, an unfair situation where some nodes transmit a lot of information and some other transmit nothing is avoided. In this arrangement, all t nodes having a requested subfile contribute equally in providing it to the node having requested it by including just their corresponding packet in their xOR-ed transmission.

The rate achieved by this scheme is

$$R(M, N, K) = \frac{N}{M}\left(1 - \frac{M}{N}\right) \tag{2.34}$$

The authors of [16] show that this rate is order optimal by proving that as K, $N \to \infty$ and $t \geq 1$

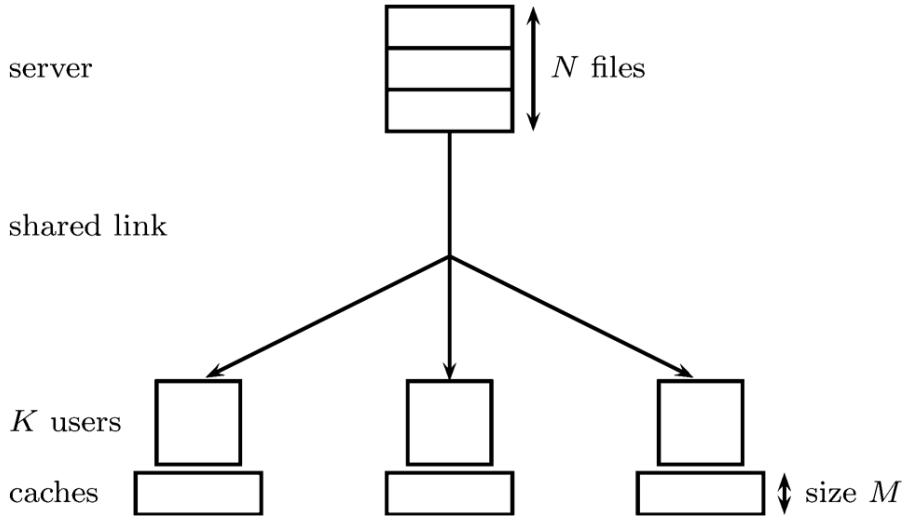


Figure 2.6: Online Caching [12]. The server contains N files of size F that are popular among the system. It communicates through a shared, error-free channel with K users each of cache memory size MF bit. The contents of library N can change over time.

$$\frac{R(M,N,K)}{R^*(M,N,K)} \leq \begin{cases} 4, & t=\omega(1), \frac{1}{2} \leq M = o(m) \\ 8, & n=O(m), t=\Theta(1), \frac{1}{2} \leq M = o(m) \\ 6, & M=\Theta(m) \\ \frac{2}{M}, & n=\omega(m), M<\frac{1}{2} \\ \frac{4}{M}, & n=O(m), n>m, M<\frac{1}{2} \\ 2, & n=O(m), n \leq m, M<\frac{1}{2} \end{cases} \tag{2.35}$$

where $R^*(M,N,K)$ is the information-theoretic optimal rate and O, Θ and ω the common asymptotic notations [76].

We should again note the critical dependence of D2D Caching on the centralized caching scheme showing that any result about the latter is expected to directly affect the former. Although it has not been officially published, since the information that each node ultimately receives is the same as that in a centralized caching, it is expected that the utilizing commonality, so that transmissions corresponding to a non-leader A_{t+1} set do not take place, is expected to further decrease the achievable rate of this method. However, as of the time of this writing, this improvement has not been explored by the literature as well as neither the question of whether the utilization of commonality would lead to an information-theoretically optimal scheme.

2.6 Online Caching

In this section, we present a variance of coded caching tailored to the fact that as time goes by and system traffic is constantly analyzed, different files are recognized as popular in a system. This creates the need for updating the user caches with new content during the system's operation and before a new delivery phase has a chance to take place.

The archetypal system is presented in Fig. 2.6. In this system, the server contains a library of N files of size F that are popular among the system. It communicates with K users, each having a cache memory size of MF. The contents of the library may change as time goes by.

This creates the need for updating the contents of the user's caches without going through a new delivery phase each time the library contents change. The way online caching deals with this situation is two-fold. First, it allows the users to keep in their caches more that N partially cached files. In particular, at any given time the users keep in their cache

$$N' = \alpha N, \tag{2.36}$$

partially cached files, with $\alpha \geq 1$. The exact value can be chosen freely in this interval to optimize the system. The authors of [12] suggest selecting an $\alpha \in (1, 2]$ The initial placement is the same as the one described in decentralized caching, with each user randomly caching MK/N' bits of N' files.

The delivery procedure is again the same as the one we described for decentralized caching without utilizing commonality. The difference in online caching is when a requested file is not partially cached by the users. When this happens, the whole file is transmitted by the server and the users take this opportunity to update their caches. In particular, they discard the MF/N' bits of the least frequently sent file and replace it with an equal amount of bits taken from the transmitted file.

In this way, the users update their caches with new files that become popular or with files that were once popular and have regained their popularity (an $\alpha > 1$ assists in increasing the probability that these files are still cached by the users). This scheme has the additional advantage that if the popularity distribution is constant, the users slowly come to reflect this distribution by caching the most popular files.

The rate achieved by this scheme is the same as that in decentralized caching without utilizing commonality, which for $K \geq N$ is

$$R(M, N, K) = K \cdot (1 - M/N) \cdot \frac{N}{KM} \left(1 - (1 - M/N)^K\right) \tag{2.37}$$

The authors of [12] prove that this achievable rate is within a multiplicative and additive gap with respect to the optimal rate.

$$\frac{1}{12} R(M, N, K) \leq R^\star \leq 2R(M, N, K) + 6 \tag{2.38}$$

Of course, any improvement in decentralized caching, like the utilization of commonality, is expected to improve online caching as well and bring its rate closer to optimality.

Chapter 3

Computational Analysis of ITODM

3.1 Centralized Caching

In this section, we attempt a computational analysis of ITODM presented in [59]. The computational cost will be counted in "instructions" performed or "number of XORs" required to do a calculation. We use the symbols $|\cdot|_c$ and $|\cdot|_s$ to refer to the computational cost and the number of bits, respectively. So for example, if A, B and C are two bit blocks of size $|A|_s = |B|_s = |C|_s = n$ bit, then XOR-ing these three blocks, will require the execution of $|A \oplus B \oplus C|_c = 2n$ instructions (or XORs at the bit level).

Theorem 1. *In a centralized coded caching system with symmetric batch prefetching, the total computational cost for a leader under the ITODM is given by*

$$C_{c,l} = \binom{K-1}{t} \frac{Ft}{\binom{K}{t}}, \tag{3.1}$$

or equivalently

$$C_{c,l} = \left(1 - \frac{M}{N}\right) \frac{MKF}{N}. \tag{3.2}$$

Proof. Under ITODM, all transmissions relevant to a leader $u \in \mathcal{U}$ take place. The number of those transmissions equals the number of $(t+1)$-subsets $A_{t+1} \in \mathcal{A}_{t+1}$ containing the user u. But this is the number of t-subsets $A_t \in \mathcal{A}_t$ that do not contain the user u. So there will be

$$\begin{aligned} |\{A_{t+1} \in \mathcal{A}_{t+1} : u \in A_{t+1}\}| &= |\{A_t \in \mathcal{A}_t : u \notin A_t\}| \\ &= \binom{K-1}{t} \end{aligned} \tag{3.3}$$

transmissions relative to leader u.

Let $Y_{A_{t+1}}$ be one such transmission. In order for u to recover the subfile $W_{d_u, A_{t+1}\backslash\{u\}}$, $Y_{A_{t+1}}$ must be XORed with all the subfiles $W_{d_{u'}, A_{t+1}\backslash\{u'\}}$ requested by the other users $u' \in A_{t+1}\backslash\{u\}$ that are already present in u's cache.

$$W_{d_u, A_{t+1}\backslash\{u\}} = Y_{A_{t+1}} \underset{u' \in A_{t+1}\backslash\{u\}}{\oplus} W_{d_{u'}, A_{t+1}\backslash\{u'\}}. \tag{3.4}$$

There are $|A_{t+1}\backslash\{u\}| = t$ such subfiles so there will be t block-XORs with each block having a size of $|W_{d_u, A_{t+1}\backslash\{u\}}| = F/\binom{K}{t}$ bit. So the computational cost of decoding the subfile $W_{d_u, A_{t+1}\backslash\{u\}}$

will be

$$\begin{aligned} C_c(W_{d_u,A_{t+1}\backslash\{u\}}) &= \left| Y_{A_{t+1}} \underset{u'\in A_{t+1}\backslash\{u\}}{\oplus} W_{d'_u,A_{t+1}\backslash\{u'\}} \right|_c \\ &= \frac{Ft}{\binom{K}{t}}. \end{aligned} \tag{3.5}$$

Since there are $\binom{K-1}{t}$ such subfiles that need decoding, the total computational cost for a leader will be expressed by (3.1) or, substituting the binomial coefficients, by (3.2). □

Finding an expression for the total computational cost of a non-leader $u^{nl} \in [K]\backslash\{\mathcal{U}\}$ is not so straightforward since for different non-leader subsets $A_{t+1} \in \mathcal{A}^{nl}_{t+1}$, the computation of $Y_{A_{t+1}}$ according to (2.9) has different computational cost. Nevertheless, due to the importance of such a quantity,we make an attempt to derive an expression for it starting from the cost of calculating the subfile $W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}$ that is shared among the non-leaders in $A_{t+1}\backslash\{u^{nl}\}$.

In order to give an exact expression for this computational cost, we will use some additional terminology introduced in [59]. If A_{t+1} is a non-leader set, then the corresponding $\mathcal{B} = \mathcal{U} \cup A_{t+1}$ is partitioned into sets of users having the same demand. We will call these sets the "tail" of the corresponding leader. So we have the following definition:

Definition 1. *For any non-leader set A_{t+1} and any leader $u \in \mathcal{B} = \mathcal{U} \cup A_{t+1}$ we define the tail of this leader as*

$$\mathcal{B}_u = \{x \in \mathcal{B} : d_u = d_x\}. \tag{3.6}$$

Using this definition, it is easy to see that each $\mathcal{V}$-set in (2.8) is, in fact, a selection of users among the different tails, taking one from each tail. Thus, we have that

$$\mathcal{V}^F \simeq \mathcal{B}_{u_1} \times \mathcal{B}_{u_2} \times \cdots \times \mathcal{B}_{u_{N_e(\mathbf{d})}}. \tag{3.7}$$

The symbol "$\simeq$" is used to show that the two sets are not equal per se, but can be put in a one-to-one correspondence. As a matter of fact, if we change the tuple structure to that of a subset (by dropping the ordering) then the two become equal. We can now state the following theorem.

Theorem 2. *In a centralized coded caching system with symmetric batch prefetching, let A_{t+1} be a set comprised of non-leaders. The computational cost for a non-leader $u^{nl} \in [K]\backslash\{\mathcal{U}\}$ for decoding a specific subfile $W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}$ shared among the other non-leaders $u' \in A_{t+1\backslash\{u\}}$ is*

$$C_{c,nl}\left(W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}\right) = \left(|\mathcal{V}^F| - 2\right)\frac{F}{\binom{K}{t}} + \frac{Ft}{\binom{K}{t}}, \tag{3.8}$$

where $|\mathcal{V}^F| = \prod_{u\in\mathcal{U}} |\mathcal{B}_u|$.

Proof. According to the ITODM, the computation of $W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}$ is done in two parts. The first part is computing the signal $Y_{A_{t+1}}$ from the actual transmissions using (2.9). This computation involves $|\mathcal{V}^F\backslash\{U\}| = |\mathcal{V}^F| - 1$ signals and thus needs $|\mathcal{V}^F| - 2$ block-XORs. Each block has size $|Y_{A_{t+1}}| = F\big/\binom{K}{t}$ bit which leads to the first summand of (3.8).

The second step has to do with decoding the actual subfile $W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}$ from the computed signal $Y_{A_{t+1}}$. This step is the same as the corresponding one for a leader, so its computational cost is given directly from (3.5), leading to the second summand of (3.8). The expression for $|\mathcal{V}^F|$ comes directly from (3.7). □

We can now proceed with our attempt for a total characterization of the computational cost of a non-leader for a particular demand. For this, we will use the following definitions.

Definition 2. *Let* $\mathbf{d} \in [N]^K$ *a demand with* $N_e(\mathbf{d})$ *distinct file requests and* $\mathcal{U}$ *the leader set. We call the set of the non-leaders requesting the same file as leader* $u_i \in \mathcal{U}$ *the pure tail of this leader.*

$$Q_{u_i} = \{u \in [K] \backslash \mathcal{U} : l(u) = u_i\}, \tag{3.9}$$

where $l(u)$ *signifies the leader of* u*.*

For the rest of this section, without loss of generality (w.l.o.g.) we assume that the non-leader we are interested in belongs in Q_{u_1}, that is $l(u^{nl}) = u_1$. This will simplify our expressions while still covering the general case with a simple renaming of the users. It also means that while the cardinality of the other pure tails can be between 0 and $K - N_e(\mathbf{d}) - 1$, the cardinality of Q_{u_1} is between 1 and $K - N_e(\mathbf{d})$.

Definition 3. *For demand* $\mathbf{d} \in [N]^K$ *and leader set* $\mathcal{U}$*, we call* $\mathcal{W}(u^{nl})$ *the set of all non-leader* $(t+1)$*-sets that contain user* u^{nl}*.*

$$\mathcal{W}(u^{nl}) = \{A_{t+1} \in \mathcal{A}^{nl}_{t+1} : u^{nl} \in A_{t+1}\}. \tag{3.10}$$

It is easy to see that $|\mathcal{W}(u^{nl})| = \binom{K-N_e(\mathbf{d})-1}{t}$ because when forming a set $A_{t+1} \in \mathcal{W}(u^{nl})$ we get to choose only t among $K - N_e(\mathbf{d}) - 1$ non-leaders since user u^{nl} is always selected by definition. We can understand that after transmission has taken place, the non-leader u^{nl} is missing exactly the subfiles that correspond to the sets in $\mathcal{W}(u^{nl})$.

Definition 4. *Let* $A_{t+1} \in \mathcal{W}(u^{nl})$*. We call the vector* $(|\mathcal{B}_{u_1}|, \ldots, |\mathcal{B}_{u_{N_e(\mathbf{d})}}|)$ *the profile of* A_{t+1}*.*

Since $l(u^{nl}) = 1$, we have $|\mathcal{B}_{u_1}| \in \{2, \ldots, |Q_{u_1}| + 1\}$ while for the rest we have $|\mathcal{B}_{u_i}| \in \{1, \ldots, |Q_{u_i}| + 1\}$.

It is easy to see that all subfiles $W_{d_{u^{nl}, A_{t+1} \backslash \{u^{nl}\}}}$ coming from sets A_{t+1} that have the same profile have the same computational costs given by (3.8). The following theorem gives the number of non-leader sets $A_{t+1} \in \mathcal{W}(u^{nl})$ that have a particular profile.

Theorem 3. *Let* $(k_1, \ldots, k_{N_e}(\mathbf{d})) \in \{0, \ldots, |Q_{u_1}| - 1\} \times \{0, \ldots, |Q_{u_i}|\}^{N_e(\mathbf{d})-1}$ *such that* $k_1 + \cdots + k_{N_e(\mathbf{d})} = t$*. The number of* $A_{t+1} \in \mathcal{W}(u^{nl})$ *having profile* $(k_1 + 2, k_2 + 1, \ldots, k_{N_e}(\mathbf{d}) + 1)$ *is*

$$\binom{|Q_{u_1}| - 1}{k_1} \binom{|Q_{u_2}|}{k_2} \cdots \binom{|Q_{u_{N_e(\mathbf{d})}}|}{k_{N_e}(\mathbf{d})}. \tag{3.11}$$

Proof. For $i \neq 1$, since the pure tail of leader u_i has $|Q_{u_i}|$ elements, there are $\binom{|Q_{u_i}|}{k_i}$ ways to choose k_i of them. For $i = 1$, user u^{nl} is always chosen by definition. That leaves us with the rest $|Q_{u_1}| - 1$ non-leaders among which to choose k_1 which can be done in $\binom{|Q_{u_1}|-1}{k_i}$ different ways. Thus, the total number of non-leader sets $A_{t+1} \in \mathcal{W}(u^{nl})$ that we can form with the particular profile is given by the product of the above binomial coefficients. □

Now we are in a position to calculate the total computational cost for the non-leader u^{nl} needed for deriving the non-transmitted subfiles from the transmitted ones by summing (3.8) over all $A_{t+1} \in \mathcal{W}(u^{nl})$, in effect proving the following theorem.

Theorem 4. *Suppose a centralized caching scheme with symmetric batch prefetching and a demand* $\mathbf{d}$ *with* $N_e(\mathbf{d})$ *leaders whose pure tail sizes are* $|Q_{u_i}|$, $i \in \{1, \dots, N_e(\mathbf{d})\}$. *Under ITODM, the total computational cost of a non-leader* u^{nl} *whose (w.l.o.g.) leader is* u_1 *needed for computing the non-transmitted subfiles from the transmitted ones is*

$$C_{c,nl}^{nt} = S\frac{F}{\binom{K}{t}} + \binom{K - N_e(\mathbf{d}) - 1}{t}\frac{F(t-2)}{\binom{K}{t}}, \tag{3.12}$$

where

$$\begin{aligned} S = \sum_{\substack{(k1,\dots,k_{N_e(\mathbf{d})}) \in R \\ k_1+\dots+k_{N_e(\mathbf{d})}=t}} & \binom{|Q_{u_1}|-1}{k_1}\binom{|Q_{u_2}|}{k_2}\dots\binom{|Q_{u_{N_e(\mathbf{d})}}|}{k_{N_e}(\mathbf{d})} \\ & \times (k_1+2)(k_2+1)\dots(k_{N_e(\mathbf{d})}+1) \end{aligned} \tag{3.13}$$

and

$$R = \{0, \dots, |Q_{u_1}| - 1\} \times \{0, \dots, |Q_{u_i}|\}^{N_e(\mathbf{d})-1}. \tag{3.14}$$

As we expected, the expression for this computational cost is quite complicated due to the many factors affecting the end result. The difficulty lies in calculating the factor S appearing in the above theorem. However, it is possible to further study this term and reach a somewhat simple form.

To do so, we start from (3.13) and do the multiplications among the different k_i giving us the following result

$$\begin{aligned} S = \sum_{l=0}^{N_e(\mathbf{d})} \Bigg(& \sum_{1<i_2<\dots<i_l\le N_e(\mathbf{d})} \sum_{\substack{(k1,\dots,k_{N_e(\mathbf{d})}) \in R \\ k_1+\dots+k_{N_e(\mathbf{d})}=t}} \binom{|Q_{u_1}|-1}{k_1}\dots\binom{|Q_{u_{N_e(\mathbf{d})}}|}{k_{N_e}} k_1 k_{i_1}\dots k_{i_l} \\ & +2 \sum_{1<i_1<\dots<i_l\le N_e(\mathbf{d})} \sum_{\substack{(k1,\dots,k_{N_e(\mathbf{d})}) \in R \\ k_1+\dots+k_{N_e(\mathbf{d})}=t}} \binom{|Q_{u_1}|-1}{k_1}\dots\binom{|Q_{u_{N_e(\mathbf{d})}}|}{k_{N_e}} k_{i_1}\dots k_{i_l} \Bigg). \end{aligned} \tag{3.15}$$

We should note that the set R in this expression can be generalized to the whole $\mathbb{N}^{N_e(\mathbf{d})}$ since there are not any additional tuples in $\mathbb{N}^{N_e(\mathbf{d})}$ that lead to a non-zero summand. Now, using the binomial coefficient property $\binom{n}{k}k = n\binom{n-1}{k-1}$ and the generalized Vandermonde's identity [77], which reads

$$\sum_{\substack{(k_1,\dots,k_p)\in\mathbb{N}^p \\ k_1+\dots+k_p=m}} \binom{n_1}{k_1}\dots\binom{n_p}{k_p} = \binom{n_1+\dots+n_p}{m}, \tag{3.16}$$

as well as the fact that the sum of the pure tail sizes is equal to the number on non-leaders, we can reach the following form

$$\begin{aligned} S = \sum_{l=0}^{N_e(\mathbf{d})} \binom{K - N_e(\mathbf{d}) - 1 - l}{t-l} \Bigg(& \sum_{1<i_2<\dots<i_l\le N_e(\mathbf{d})} (|Q_{u_1}|-1)\,|Q_{u_{i_2}}|\dots|Q_{u_{i_l}}| \\ & +2 \sum_{1<i_1<\dots<i_l\le N_e(\mathbf{d})} |Q_{u_{i_1}}|\dots|Q_{u_{i_l}}| \Bigg) \end{aligned} \tag{3.17}$$

This form highlights the fact that for $N_e(\mathbf{d}) > t$ the summation ends at $l = t$. We could have seen this in the previous form of (3.15) if we had noticed that when $N_e(\mathbf{d}) > t$ the condition $k_1 + ... + k_{N_e(\mathbf{d})} = t$ allows only for up to t terms to be non-zero at the same time. However, this is easier to see in (3.17). Also, since $(|Q_{u_1-1}) + |Q_{u_2}| + \cdots + |Q_{u_{N_e(\mathbf{d})}}| = K - N_e(\mathbf{d}) - 1$, only up to $K - N_e(\mathbf{d}) - 1$ terms of the above sum can be non-zero at the same time. That means that if $N_e(\mathbf{d}) > K - N_e(\mathbf{d}) - 1$, the summation stops at $l = K - N_e(\mathbf{d}) - 1$.

This allows us to replace $N_e(\mathbf{d})$ in the sum with $\min(t, N_e(\mathbf{d}))$ and reach (3.18).

$$S = \sum_{l=0}^{\min(t,N_e(\mathbf{d}))} \binom{K - N_e(\mathbf{d}) - 1 - l}{t - l} \Bigg(\sum_{1<i_2<\cdots<i_l\leq N_e(\mathbf{d})} (|Q_{u_1}| - 1)\,|Q_{u_{i_2}}| \ldots |Q_{u_{i_l}}| + 2 \sum_{1<i_1<\cdots<i_l\leq N_e(\mathbf{d})} |Q_{u_{i_1}}| \ldots |Q_{u_{i_l}}| \Bigg). \tag{3.18}$$

Now, we can fully characterize the total computational cost of a non-leader. Theorem 4 gives us the computational cost for calculating the untransmitted subfiles. For the transmitted ones, the situation is similar to that of a leader. The cost for decoding each transmitted subfile is given by (3.5). Taking into account that the total transmitted subfiles requested by u^{nl} are $\binom{K-1}{t} - \binom{K-N_e(\mathbf{d})-1}{t}$ (the total subfiles minus the non-transmitted ones) we reach the following theorem.

Theorem 5. *In a centralized caching scheme with symmetric batch prefetching let* $\mathbf{d}$ *be a demand with* $N_e(\mathbf{d})$ *leaders whose pure tail sizes are* $|Q_{u_i}|$, $i \in \{1, \ldots, N_e(\mathbf{d})\}$. *Under ITODM, the total computational cost of a non-leader* u^{nl} *whose (w.l.o.g.) leader is* u_1 *needed for recovering their requested subfiles is*

$$C_{c,nl} = \binom{K-1}{t} \frac{Ft}{\binom{K}{t}} + \left[S - 2\binom{K - N_e(\mathbf{d}) - 1}{t} \right] \frac{F}{\binom{K}{t}}, \tag{3.19}$$

with S *given by either (3.13) or (3.18)*

3.2 Decentralized Caching

For simplicity, we will assume that the choice of leaders and non-leaders is the same among all centralized instances of decentralized caching. This is an assumption that places a big computational burden on the users chosen as non-leaders. Alternate scenarios that seek to distribute the extra computational burden of being a non-leader among the users can modify this assumption and offer a more balanced computational load between the users. However, this assumption provides a solid starting point for such further analyses, which makes it quite important to study.

Extending the previous section analysis to decentralized caching, the exact computational cost for this scheme is given in the three following theorems.

Theorem 6. *In a decentralized coded caching system with symmetric batch prefetching, the total computational cost for a leader under ITODM is*

$$C_{c,l}^{dec} = (K-1)\left(1 - \frac{M}{N}\right)\frac{MF}{N} + o(F). \tag{3.20}$$

Proof. Since decentralized caching is comprised of multiple instances of centralized caching, we can sum up the computational cost of a leader over all these instances. The computational cost of a single instance of centralized caching is given by (3.2) after we substitute F with the equivalent file size given by (2.14).

So the total computational cost for a leader will be

$$\begin{aligned} C_{c,l}^{dec} &= \sum_{j=0}^{K-1} \binom{K-1}{j} \frac{|\mathcal{B}_j^L(W)|j}{\binom{K}{j}} \\ &= \sum_{j=0}^{K-1} \binom{K-1}{j} \left(\frac{M}{N}\right)^j \left(1-\frac{M}{N}\right)^{K-j} jF. \end{aligned} \tag{3.21}$$

Taking the term $1-\frac{M}{N}$ out of the sum and using the formula [77] for the expected value the binomial distribution $\mathcal{B}(K-1, M/N)$ we reach (3.20) after plugging in the term $o(F)$ that we omitted for simplicity. □

Theorem 7. *Suppose a decentralized caching scheme and a demand* $\mathbf{d}$ *with* $N_e(\mathbf{d})$ *leaders whose pure tail sizes are* $|Q_{u_i}|$, $i \in \{1, \ldots, N_e(\mathbf{d})\}$. *Under ITODM, the total computational cost of a non-leader* u^{nl} *whose (w.l.o.g.) leader is* u_1 *needed for deriving the non-transmitted subfiles from the transmitted ones is*

$$\begin{aligned} C_{c,nl}^{dec,nt} = \left(1-\frac{M}{N}\right)^{N_e(\mathbf{d})+1} F \Bigg(\left(\frac{M}{N}(|Q_{u1}|-1)+2\right) \prod_{i=2}^{N_e(\mathbf{d})} \left(\frac{M}{N}|Q_{u_i}|+1\right) \\ + \frac{M}{N}(K-N_e(\mathbf{d})-1)-2 \Bigg) + o(F). \end{aligned} \tag{3.22}$$

Proof. We recall that the decentralized caching is, in fact, broken down to multiple instances of centralized caching, with each instance dealing with the bits shared among j users, with $j \in \{0, 1, \ldots, K\}$. Since for $j \geq K - N_e(\mathbf{d})$, there are no $(j+1)$-sets composed of solely non-leaders, all subfile transmissions take place and there is no computational cost involved with deriving any untransmitted subfiles. Thus, in order to find out the total cost of a non-leader related to the computation of the non-transmitted subfiles, we just have to add the individual computational costs for j up to $K - N_e(\mathbf{d}) - 1$, as given by theorem 4. If $C_{c,nl}^{nt}(j)$ is the computational cost of u^{nl} in the j-th instance of centralized caching, then the total cost for non-leader u^{nl} will be

$$\begin{aligned} C_{c,nl}^{dec,nt} &= \sum_{j=0}^{K-N_e(\mathbf{d})-1} C_{c,nl}^{nt}(j) \\ &= \underbrace{\sum_{j=0}^{K-N_e(\mathbf{d})-1} S(j) \frac{|\mathcal{B}_j^L(W)|}{\binom{K}{j}}}_{S_1} + \underbrace{\sum_{j=0}^{K-N_e(\mathbf{d})-1} \binom{K-N_e(\mathbf{d})-1}{j} \frac{|\mathcal{B}_j^L(W)|j}{\binom{K}{j}}}_{S_2} \\ &\quad - \underbrace{\sum_{j=0}^{K-N_e(\mathbf{d})-1} \binom{K-N_e(\mathbf{d})-1}{j} \frac{|\mathcal{B}_j^L(W)|2}{\binom{K}{j}}}_{S_3}. \end{aligned} \tag{3.23}$$

The above expression comes from (3.19) by replacing the file size F with the equivalent file size $|\mathcal{B}_j^L(W)|$ of the j-th instance of centralized caching given by (2.14) and instead of S using $S(j)$, which is the value of (3.13) for $t = j$. We should note that for $j = 0$ the computational cost is actually zero since there is no derivation happening. The information in the corresponding Y_{A_1} transmissions that take place is readily available to any non-leader that is interested in it. However, we let j in the above expression start from zero because it will help us in our manipulations.

We will now examine each of the three summands by itself. We should note that for simplicity, in substituting $|\mathcal{B}_j^L(W)|$, we will omit the term $o(F)$ of (2.14), which nevertheless is always present. Starting with S_1 and using (3.18) for $S(j)$ gives us

$$\begin{aligned} S_1 &= \sum_{j=0}^{K-N_e(\mathbf{d})-1} \sum_{l=0}^{\min(j,N_e(\mathbf{d}))} \binom{K-N_e(\mathbf{d})-1-l}{j-l} \mathcal{Q}(l) \left(\frac{M}{N}\right)^j \left(1-\frac{M}{N}\right)^{K-j} F \\ &= \sum_{\{j,l\}\in\mathcal{R}} \binom{K-N_e(\mathbf{d})-1-l}{j-l} \mathcal{Q}(l) \left(\frac{M}{N}\right)^j \left(1-\frac{M}{N}\right)^{K-j} F, \end{aligned} \tag{3.24}$$

where

$$\begin{aligned} \mathcal{Q}(l) = &\sum_{1<i_2<\cdots<i_l\le N_e(\mathbf{d})} (|Q_{u_1}|-1)\,|Q_{u_{i_2}}|\ldots|Q_{u_{i_l}}| \\ &+2 \sum_{1<i_1<\cdots<i_l\le N_e(\mathbf{d})} |Q_{u_{i_1}}|\ldots|Q_{u_{i_l}}|, \end{aligned} \tag{3.25}$$

and

$$\mathcal{R} = \left\{(j,l)\in\mathbb{N}^2 : \begin{array}{l} j\in\{0,\ldots,K-N_e(\mathbf{d})-1\}, \\ l\in\{0,\ldots,\min(j,N_e(\mathbf{d}))\} \end{array}\right\}. \tag{3.26}$$

Region $\mathcal{R}$ can be equivalently described as

$$\mathcal{R} = \left\{(j,l)\in\mathbb{N}^2 : \begin{array}{l} l\in\{0,\ldots,m\} \\ j\in\{1,\ldots,K-N_e(\mathbf{d})-1\} \end{array}\right\}, \tag{3.27}$$

where

$$m = \min(K-N_e(\mathbf{d})-1, N_e(\mathbf{d})). \tag{3.28}$$

This allows us to exchange the order of summation in (3.24) and get

$$\begin{aligned} S_1 &= \sum_{l=0}^{m} \sum_{j=l}^{K-N_e(\mathbf{d})-1} \binom{K-N_e(\mathbf{d})-1-l}{j-l} \mathcal{Q}(l) \left(\frac{M}{N}\right)^j \left(1-\frac{M}{N}\right)^{K-j} F \\ &\overset{(a)}{=} \sum_{l=0}^{m} \mathcal{Q}(l) \left(\frac{M}{N}\right)^l \left(1-\frac{M}{N}\right)^{N_e(\mathbf{d})+1} F \left\{ \sum_{j'=0}^{K-N_e(\mathbf{d})-1-l} \binom{K-N_e(\mathbf{d})-1-l}{j'} \right. \\ &\qquad \left. \times \left(\frac{M}{N}\right)^{j'} \left(1-\frac{M}{N}\right)^{K-N_e(\mathbf{d})-1-l-j'} \right\} \\ &\overset{(b)}{=} \sum_{l=0}^{m} \mathcal{Q}(l) \left(\frac{M}{N}\right)^l \left(1-\frac{M}{N}\right)^{N_e(\mathbf{d})+1} F. \end{aligned} \tag{3.29}$$

In (a) we make the change $j' = j - l$ and tidy up the term a bit and in (b) we apply the binomial theorem [77]. Plugging in $\mathcal{Q}(l)$ from (3.25), the expression for S_1 becomes

$$S_1 = \left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} F \sum_{l=0}^{m} \left(\sum_{1<i_2<\cdots<i_l\leq N_e(\mathbf{d})} \frac{M}{N}\left(|Q_{u_1}| - 1\right) \frac{M}{N}|Q_{u_{i_2}}| \ldots \frac{M}{N}|Q_{u_{i_l}}| \right. \\ \left. +2 \sum_{1<i_1<\cdots<i_l\leq N_e(\mathbf{d})} \frac{M}{N}|Q_{u_{i_1}}| \ldots \frac{M}{N}|Q_{u_{i_l}}| \right). \tag{3.30}$$

Recovering now the product terms from the sums of this expression gives us

$$S_1 = \left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} F \left(\frac{M}{N}(|Q_{u1}| - 1) + 2\right) \prod_{i=2}^{N_e(\mathbf{d})} \left(\frac{M}{N}|Q_{u_i}| + 1\right). \tag{3.31}$$

The above derivation is straightforward if $m = N_e(\mathbf{d})$. If $m = K - N_e(\mathbf{d}) - 1$ then from

$$\left(|Q_{u_1}| - 1\right) + |Q_{u_2}| + \cdots + |Q_{u_{N_e(\mathbf{d})}}| = K - N_e(\mathbf{d}) - 1 \tag{3.32}$$

we have that only up to $K - N_e(\mathbf{d}) - 1$ summands can be concurrently non-zero. Thus we can extend the sum of (3.30) up to $N_e(\mathbf{d})$ by including the zero valued terms and get (3.31).

The terms S_2 and S_3 are easier to handle. In particular

$$\begin{aligned} S_2 &= \sum_{j=0}^{K-N_e(\mathbf{d})-1} \binom{K - N_e(\mathbf{d}) - 1}{j} \left(\frac{M}{N}\right)^j \left(1 - \frac{M}{N}\right)^{K-j} jF \\ &= \left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} F \sum_{j=0}^{K-N_e(\mathbf{d})-1} \binom{K - N_e(\mathbf{d}) - 1}{j} \left(\frac{M}{N}\right)^j \left(1 - \frac{M}{N}\right)^{K-N_e(\mathbf{d})-1-j} j \\ &= \left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} (K - N_e(\mathbf{d}) - 1)\frac{MF}{N}. \end{aligned} \tag{3.33}$$

In this expression, we have used the formula [77] for the expected value of the binomial distribution $\mathcal{B}\left(K - N_e(\mathbf{d}) - 1, M/N\right)$.

Finally, for the third term, using the binomial theorem, we can easily get

$$\begin{aligned} S_3 &= 2 \sum_{j=0}^{K-N_e(\mathbf{d})-1} \binom{K - N_e(\mathbf{d}) - 1}{j} \left(\frac{M}{N}\right)^j \left(1 - \frac{M}{N}\right)^{K-j} F \\ &= 2\left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} F \sum_{j=0}^{K-N_e(\mathbf{d})-1} \binom{K - N_e(\mathbf{d}) - 1}{j} \left(\frac{M}{N}\right)^j \left(1 - \frac{M}{N}\right)^{K-N_e(\mathbf{d})-1-j} \\ &= 2\left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} F. \end{aligned} \tag{3.34}$$

Plugging back all these terms in (3.23) and re-including the term $o(F)$ that we omitted before, we reach (3.22) that we are trying to prove. □

We should note here that this a surprisingly simple result, especially when compared to the corresponding expression (3.12) for the computational cost in the centralized caching scheme, which is quite more complicated, even if we use the simplified form of (3.18) for the term S. This fact bears the question of whether it could be possible to devise a different but information-theoretically equivalent scheme for decentralized caching that could be more suited as the basic paradigm of coded caching with uncoded prefetching instead of the centralized one.

Theorem 8. *In a decentralized caching scheme let* $\mathbf{d}$ *be a demand with* $N_e(\mathbf{d})$ *leaders whose pure tail sizes are* $|Q_{u_i}|$, $i \in \{1, \dots, N_e(\mathbf{d})\}$. *Under ITODM, the total computational cost of a non-leader* u^{nl} *whose (w.l.o.g.) leader is* u_1 *needed for recovering their requested subfiles is*

$$C^{dec}_{c,nl} = \left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} F \left(\left(\frac{M}{N}(|Q_{u1}|-1)+2\right) \prod_{i=2}^{N_e(\mathbf{d})} \left(\frac{M}{N}|Q_{u_i}|+1\right) - 2 \right) + (K-1)\left(1-\frac{M}{N}\right)\frac{MF}{N} + o(F). \tag{3.35}$$

Proof. In order to derive (3.35) we just have to make some simple observations on (3.23) in order to extend it to the total computational cost of a non-leader. This expression gives the computational cost of u^{nl} in order to extract all the non-transmitted subfiles from the transmitted ones. So in order to get the total computational cost, we have to add to this expression the cost of decoding the transmitted subfiles.

For each $j \in \{0, \dots, K - N_e(\mathbf{d}) - 1\}$ the additional computational cost for decoding the transmitted subfiles is

$$\underbrace{\left[\binom{K-j}{j} - \binom{K-N_e(\mathbf{d})-1}{j}\right]}_{\text{number of transmitted subfiles}} \underbrace{\frac{|\mathcal{B}^L_j(W)|j}{\binom{K}{j}}}_{\substack{\text{cost per} \\ \text{subfile}}}. \tag{3.36}$$

Also for each $j \in \{K - N_e(\mathbf{d}), \dots, K-1\}$ the only computational cost is that of decoding the transmitting subfiles, since there are no $(j+1)$-subsets comprised solely of non-leaders. This additional cost is

$$\underbrace{\binom{K-j}{j}}_{\substack{(j+1)-\text{subsets} \\ \text{involving } u^{nl}}} \underbrace{\frac{|\mathcal{B}^L_j(W)|j}{\binom{K}{j}}}_{\substack{\text{cost per} \\ \text{subfile}}}. \tag{3.37}$$

We should note that for $j = K$, the corresponding bits are present in the caches of all users, and thus there is no computation (or even transmission) taking place. Adding up these additional computational costs in (3.23) we can realize that the total computational cost of u^{nl} will be given by this equation, if we replace the sum S_2 with

$$\begin{aligned} S_2' &= \sum_{j=0}^{K-1} \binom{K-1}{j} \frac{|\mathcal{B}^L_j(W)|j}{\binom{K}{j}} \\ &= \sum_{j=0}^{K-1} \binom{K-1}{j} \left(\frac{M}{N}\right)^j \left(1-\frac{M}{N}\right)^{K-j} jF \\ &= (K-1)\left(1-\frac{M}{N}\right)\frac{MF}{N}. \end{aligned} \tag{3.38}$$

Using this result and following the same operations for the other terms of (3.23) as well as re-positioning the term $o(F)$ that we omitted for simplicity, yields (3.35). □

Chapter 4

Computationally Enhanced Decoding Method

4.1 Method Description for Centralized Caching

In this section, we propose an alternative way for a non-leader $u^{nl} \in [K]\backslash\{\mathcal{U}\}$ to decode the subfile of interest $W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}$ for any t-subset A_{t+1} comprised of non-leaders. Using this method, the non-leader will perform a direct computation of the subfile from previous leader-related transmissions without having to compute the signal $Y_{A_{t+1}}$ as it is normally done in [59]. Moreover, we show that the computational cost of this new method is equal to the cost of computing $Y_{A_{t+1}}$, thus saving the last step of figuring out $W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}$ from $Y_{A_{t+1}}$. We first show the following lemma.

Lemma 1. *Assume a wireless caching system with N files of size F, K users and MF cache size per user where symmetric batch prefetching is used and $t = \frac{MK}{N} \in \{0, 1, \dots K\}$. For any demand $\mathbf{d} = (d_1, \dots, d_K)$ with $N_e(\mathbf{d}) \leq K - t - 1$, any leader set $\mathcal{U} = \{u_1, \dots, u_{N_e(\mathbf{d})}\}$, any $(t+1)$ non-leader subset $A_{t+1} \in \mathcal{A}^{nl}_{t+1}$ and any non-leader $u^{nl} \in A_{t+1}$ we have*

$$\bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} Y_{\mathcal{B}\backslash\mathcal{V}} \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\notin\mathcal{V}}} W_{d_{u^{nl}},\mathcal{B}\backslash(\mathcal{V}\cup\{u^{nl}\})} = 0, \tag{4.1}$$

where $\mathcal{B} = \mathcal{U} \cup A_{t+1}$ and $\mathcal{V}_F$ is the family of $\mathcal{V}$-sets of $\mathcal{B}$, each containing $N_e(\mathbf{d})$ users from $\mathcal{B}$ with each one requesting a different file.

The proof will expand on the one given in [59] for Lemma 1. In addition to the family $\mathcal{V}^F$ containing the $\mathcal{V}$-sets of $\mathcal{B} = \mathcal{U} \cup A_{t+1}$ described in section 2, we will utilize two additional set families defined below, before we move to the actual proof. The first also comes from [59].

Definition 5. *Let $u \in \mathcal{U}$ be any leader. We define $\mathcal{V}^F_u$ to be the family of the sets formed by selecting one user from each tail in $\mathcal{B}$ except $\mathcal{B}_u$. In particular*

$$\mathcal{V}^F_u = \left\{ \mathcal{V} \in 2^{\mathcal{B}\backslash\mathcal{B}_u} : \begin{array}{l} |\mathcal{V}| = N_e(\mathbf{d}) \text{ - } 1, \\ \forall u_1 \neq u_2 \in \mathcal{V}\ d_{u_1} \neq d_{u_2} \end{array} \right\}. \tag{4.2}$$

Definition 6. *Let $u, u' \in \mathcal{U}$ be any two leaders. We define $\mathcal{V}^F_{u,u'}$ to be the family of the sets formed by selecting one user from each tail in $\mathcal{B}$ except $\mathcal{B}_u$ and $\mathcal{B}_{u'}$. In particular*

$$\mathcal{V}^F_{u,u'} = \left\{ \mathcal{V} \in 2^{\mathcal{B}\backslash(\mathcal{B}_u\cup\mathcal{B}_{u'})} : \begin{array}{l} |\mathcal{V}| = N_e(\mathbf{d}) \text{ - } 2, \\ \forall u_1 \neq u_2 \in \mathcal{V}\ d_{u_1} \neq d_{u_2} \end{array} \right\}. \tag{4.3}$$

As in $\mathcal{V}^F$, the sets in $\mathcal{V}_u^F$ and $\mathcal{V}_{u,u'}^F$ can be generated by starting with $\mathcal{U}\backslash\{u\}$ or $\mathcal{U}\backslash\{u,u'\}$ and then replacing one or more leaders by one of their non-leaders (having the same request) in $\mathcal{B}\backslash\mathcal{B}_u$ or $\mathcal{B}\backslash(\mathcal{B}_u \cup \mathcal{B}_{u'})$, respectively.

Proof. We start by studying the term

$$A = \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} Y_{\mathcal{B}\backslash\mathcal{V}} = \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} \bigoplus_{x\in\mathcal{B}\backslash\mathcal{V}} W_{d_x,\mathcal{B}\backslash(\mathcal{V}\cup\{x\})}. \tag{4.4}$$

Since the tails $B_u = \{x \in \mathcal{B} : d_x = d_u\}$ of the leaders $u \in \mathcal{U}$ partition the set $\mathcal{B}$, we have that

$$\begin{aligned} A &= \bigoplus_{u\in\mathcal{U}} \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} \bigoplus_{x\in(\mathcal{B}\cap\mathcal{B}_u)\backslash\mathcal{V}} W_{d_x,\mathcal{B}\backslash(\mathcal{V}\cup\{x\})} \\ &= \bigoplus_{u\in\mathcal{U}} \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} \bigoplus_{x\in\mathcal{B}_u\backslash\mathcal{V}} W_{d_x,\mathcal{B}\backslash(\mathcal{V}\cup\{x\})}. \end{aligned} \tag{4.5}$$

If we define u_0 to be the leader of u^{nl}, so that $u^{nl} \in \mathcal{B}_{u_0}$, then (4.5) can be written as

$$A = \underbrace{\bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} \bigoplus_{x\in\mathcal{B}_{u_0}\backslash\mathcal{V}} W_{d_x,\mathcal{B}\backslash(\mathcal{V}\cup\{x\})}}_{A_1} \underbrace{\bigoplus_{u\in\mathcal{U}\backslash\{u_0\}} \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} \bigoplus_{x\in\mathcal{B}_u\backslash\mathcal{V}} W_{d_x,\mathcal{B}\backslash(\mathcal{V}\cup\{x\})}}_{A_2}. \tag{4.6}$$

Also, the sets of $\mathcal{V}^F$ that contain u^{nl} can be decomposed as follows

$$\left\{\mathcal{V}\in\mathcal{V}^F : u^{nl}\in\mathcal{V}\right\} = \left\{\{u^{nl}\}\cup\mathcal{V}' : \mathcal{V}'\in\mathcal{V}_{u_0}^F\right\}. \tag{4.7}$$

That means that the term A_1 can be written as

$$\begin{aligned} A_1 &= \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} \bigoplus_{x\in\mathcal{B}_{u_0}\backslash\mathcal{V}} W_{d_x,\mathcal{B}\backslash(\mathcal{V}\cup\{x\})} \\ &= \bigoplus_{\mathcal{V}'\in\mathcal{V}_{u_0}^F} \bigoplus_{x\in\mathcal{B}_{u_0}\backslash\left(\{u^{nl}\}\cup\mathcal{V}'\right)} W_{d_x,\mathcal{B}\backslash\left(\{u^{nl}\}\cup\mathcal{V}'\cup\{x\}\right)} \\ &= \bigoplus_{\mathcal{V}'\in\mathcal{V}_{u_0}^F} \bigoplus_{x\in\mathcal{B}_{u_0}\backslash\{u^{nl}\}} W_{d_x,\mathcal{B}\backslash\left(\{x\}\cup\mathcal{V}'\cup\{u^{nl}\}\right)}. \end{aligned} \tag{4.8}$$

We now note that the family

$$\left\{\{x\}\cup\mathcal{V}' : \begin{array}{l}\mathcal{V}'\in\mathcal{V}_{u_0}^F, \\ x\in\mathcal{B}_{u_0}\backslash\{u^{nl}\}\end{array}\right\} = \left\{\mathcal{V}\in\mathcal{V}^F : u^{nl}\notin\mathcal{V}\right\}. \tag{4.9}$$

since x ranges over the whole tail that u^{nl} belongs to but skips u^{nl} itself, and $\mathcal{V}'$ ranges over all the sets in $\mathcal{V}_{u_0}^F$. So the term A_1 gives

$$A_1 = \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\notin\mathcal{V}}} W_{d_{u^{nl}},\mathcal{B}\backslash(\mathcal{V}\cup\{u^{nl}\})}, \tag{4.10}$$

which is the second term in (4.1) canceling it out.

We now focus on the term A_2 of (4.6). The operations we will perform here will follow the more familiar path of the proof in [59]. So, as in [59], we focus on each leader $u \in \mathcal{U}\backslash\{u_0\}$ separately and study the term

$$A_{2,u} = \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in\mathcal{V}}} \bigoplus_{x\in\mathcal{B}_u\backslash\mathcal{V}} W_{d_x,\mathcal{B}\backslash(\mathcal{V}\cup\{x\})}. \tag{4.11}$$

In this term, the sets of $\mathcal{V}^F$ containing u^{nl} can be decomposed as

$$\left\{\mathcal{V}\in\mathcal{V}^F : u^{nl}\in\mathcal{V}\right\} = \left\{\{u^{nl},y\}\cup\mathcal{V}'' : \begin{array}{l} y\in\mathcal{B}_u, \\ \mathcal{V}''\in\mathcal{V}^F_{u_0,u}\end{array}\right\}. \tag{4.12}$$

So $A_{2,u}$ can be written as

$$\begin{aligned} A_{2,u} &= \bigoplus_{\mathcal{V}''\in\mathcal{V}^F_{u_0,u}} \bigoplus_{y\in\mathcal{B}_u} \bigoplus_{x\in\mathcal{B}_u\backslash\left(\{u^{nl},y\}\cup\mathcal{V}''\right)} W_{d_u,\mathcal{B}\backslash\left(\{u^{nl},y\}\cup\mathcal{V}''\cup\{x\}\right)} \\ &= \bigoplus_{\mathcal{V}''\in\mathcal{V}^F_{u_0,u}} \bigoplus_{y\in\mathcal{B}_u} \bigoplus_{x\in\mathcal{B}_u\backslash\{y\}} W_{d_u,\mathcal{B}\backslash\left(\{u^{nl},x,y\}\cup\mathcal{V}''\right)}. \end{aligned} \tag{4.13}$$

We can now observe that the set of ordered pairs

$$\{(x,y) : x\in\mathcal{B}_u\backslash\{y\}, y\in\mathcal{B}_u\} = \left\{(x,y)\in\mathcal{B}_u^2 : x\neq y\right\}, \tag{4.14}$$

so $A_{2,u}$ is

$$A_{2,u} = \bigoplus_{\mathcal{V}''\in\mathcal{V}^F_{u_0,u}} \bigoplus_{\substack{(x,y)\in\mathcal{B}_u^2 \\ x\neq y}} W_{d_u,\mathcal{B}\backslash\left(\{u^{nl},x,y\}\cup\mathcal{V}''\right)}. \tag{4.15}$$

Since, each subfile in the above expression is taken twice, we can deduce that $A_{2,u}=0$ which means that

$$A_2 = \bigoplus_{u\in\mathcal{U}\backslash\{u_0\}} A_{2,u} = 0, \tag{4.16}$$

concluding the proof. □

We can now state the following theorem that provides the means of directly calculating the subfile $W_{d_{u^{nl}},A_{t+1}}$.

Theorem 9. *Assume a wireless caching system with N files of size F, K users and MF cache size per user where symmetric batch prefetching is used and $t=\frac{MK}{N}\in\{0,1,\dots K\}$. For any demand $\mathbf{d}=(d_1,\dots,d_K)$ with $N_e(\mathbf{d})\leq K-t-1$, any leader set $\mathcal{U}=\{u_1,\dots,u_{N_e(\mathbf{d})}\}$, any $(t+1)$ non-leader subset $A_{t+1}\in\mathcal{A}^{nl}_{t+1}$ and any non-leader $u^{nl}\in A_{t+1}$ the requested subfile $W_{d_{u^{nl}},A_{t+1}}$ is given by*

$$W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}} = \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F \\ u^{nl}\in V}} Y_{\mathcal{B}\backslash V} \bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F\backslash\{\mathcal{U}\} \\ u^{nl}\notin V}} W_{d_{u^{nl}},\mathcal{B}\backslash(\mathcal{V}\cup\{u^{nl}\})}. \tag{4.17}$$

Proof. We note that

$$W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}} = W_{d_{u^{nl}},\mathcal{B}\backslash(\mathcal{U}\cup\{u^{nl}\})}, \tag{4.18}$$

which means that the subfile $W_{d_{u^{nl}},A_{t+1}\backslash\{u^{nl}\}}$ is already present in (4.1). Multiplying both sides with this leads to (4.17). □

We should note that all the terms in (4.17) are either transmitted signals or subfiles requested by u^{nl} that are directly computable from the transmitted signals. This allows the non-leader u^{nl} to compute any subfile still missing from his requested file after all the leader-related transmissions have taken place.

4.2 Computational Analysis

In this section, we analyze the computational cost of our proposed decoding method. First, we should note that both our proposed method and ITODM in [59] coincide when decoding the subfiles $W_{d_{u^{nl},A_{t+1}\backslash\{u^{nl}\}}}$ coming from actual transmissions $Y_{A_{t+1}}$ where there is at least one leader in A_{t+1}. So the computational cost for these files is the same and is given by (3.5). For the rest of the subfiles requested by u^{nl} the cost of computing them from previous transmissions and the subfiles collected from these transmissions is given by the following theorem.

Theorem 10. *For any non-leader $u^{nl} \in [K]\backslash\{\mathcal{U}\}$ and any non-leader $(t+1)$-subset A_{t+1}, the cost of computing the subfile $W_{d_{u^{nl},A_{t+1}\backslash\{u^{nl}\}}}$ is given by the expression*

$$C_{c,nl}\left(W_{d_{u^{nl},A_{t+1}\backslash\{u^{nl}\}}}\right) = (|\mathcal{V}_F| - 2)\,\frac{F}{\binom{K}{t}}. \tag{4.19}$$

Proof. In order to find the cost of computing $W_{d_{u^{nl},A_{t+1}\backslash\{u^{nl}\}}}$ we count the number of XORs needed to compute (4.17). From (3.7) we have that

$$\begin{aligned}\left|\left\{\mathcal{V}\in\mathcal{V}^F : u^{nl}\in\mathcal{V}\right\}\right| &= \left|\left\{\mathcal{V}'\in\mathcal{V}^F_{u^{nl}} : u^{nl}\in\mathcal{V}\right\}\right| \\ &= \frac{|\mathcal{V}^F|}{|\mathcal{B}_{u_0}|}\end{aligned}, \tag{4.20}$$

$$\begin{aligned}\left|\left\{\mathcal{V}\in\mathcal{V}^F : u^{nl}\notin\mathcal{V}\right\}\right| &= |\mathcal{V}^F| - \left|\left\{\mathcal{V}'\in\mathcal{V}^F_{u^{nl}} : u^{nl}\in\mathcal{V}\right\}\right| \\ &= \frac{|\mathcal{B}_{u_0}|-1}{|\mathcal{B}_{u_0}|}|\mathcal{V}^F|.\end{aligned} \tag{4.21}$$

So the number of XORs inside each of the two block-XORs of (4.17) will be

$$\left|\bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F\\ u^{nl}\in V}} Y_{\mathcal{B}\backslash V}\right|_c = \left(\frac{|\mathcal{V}^F|}{|\mathcal{B}_{u_0}|}-1\right)\frac{F}{\binom{K}{t}}, \tag{4.22}$$

and

$$\left|\bigoplus_{\substack{\mathcal{V}\in\mathcal{V}_F\backslash\{\mathcal{U}\}\\ u^{nl}\notin V}} W_{d_{u^{nl},\mathcal{B}\backslash\mathcal{V}\cup\{u^{nl}\}}}\right|_c = \left(\frac{|\mathcal{B}_{u_0}|-1}{|\mathcal{B}_{u_0}|}|\mathcal{V}^F|-2\right)\frac{F}{\binom{K}{t}}. \tag{4.23}$$

Summing those up, and taking into account that we have one more block-XOR between these two parts of (4.17), introducing $F/\binom{K}{t}$ additional bit XORs, we reach (4.19). □

Comparing (4.17) to (3.8) we can see that the proposed decoding method requires $Ft/\binom{K}{t}$ fewer XORs than ITODM per non-leader $(t+1)$-subset. Given that the number of non-leader $(t+1)$-subsets $A_{t+1}\in\mathcal{A}^{nl}_{t+1}$ containing a particular non-leader u^{nl} is

$$\begin{aligned}\left|\left\{A_{t+1}\in\mathcal{A}^{nl}_{t+1} : u^{nl}\in A_{t+1}\right\}\right| &= \left|\left\{A_t\in\mathcal{A}^{nl}_t : u^{nl}\notin A_t\right\}\right| \\ &= \binom{K-N_e(\mathbf{d})-1}{t}\end{aligned} \tag{4.24}$$

we can state the following corollary.

Corollary 1. *A non-leader using (4.17) to compute the requested subfiles $W_{d_{u^{nl},A_{t+1}\setminus\{u^{nl}\}}}$ corresponding to all $A_{t+1} \in \mathcal{A}_{t+1}^{nl}$ will have a total saving of*

$$S_{c,nl}(\mathbf{d}) = \binom{K - N_e(\mathbf{d}) - 1}{t} \frac{Ft}{\binom{K}{t}} \tag{4.25}$$

XORS relative to ITODM.

Since there are $K - N_e(\mathbf{d})$ non-leaders, we also get that the total computational savings throughout the whole system will be given by the following corollary.

Corollary 2. *The total computational savings of using (4.17) throughout the system will be*

$$\begin{aligned} S_c(\mathbf{d}) &= \big(K - N_e(\mathbf{d})\big)\binom{K - N_e(\mathbf{d}) - 1}{t} \frac{Ft}{\binom{K}{t}} \\ &= \binom{K - N_e(\mathbf{d})}{t+1} \frac{Ft(t+1)}{\binom{K}{t}} \end{aligned} \tag{4.26}$$

XORS relative to ITODM.

4.3 Extension to decentralized caching

The proposed decoding method finds applicability in any scenario that the centralized coded caching with uncoded prefetching appears in some form. In this section, we show how the previous results extend to the case of decentralized caching with uncoded prefetching and derive the corresponding computational savings per user and throughout the system.

In section 2.2 we have seen that the delivery phase of a decentralized system is comprised of multiple delivery phases that each is equivalent to the delivery phase of a centralized caching system with file size $|\mathcal{B}_{A_j}^{L,j}(W)|$ a.a.s. for $j \in \{0, 1, \ldots, K\}$. Also, for each $j \in \{1, 2, \ldots, K - N_e(\mathbf{d}) - 1\}$ there will be non-leader sets A_{j+1} whose corresponding transmissions will not take place but instead will be computed from the ones that do take place. For each such non-leader set, a non-leader could apply our proposed decoding method using (4.17) and save the number of computational operations given by Corollary 1. Summing up these computational savings for all $j \in \{1, \ldots, K - N_e(\mathbf{d}) - 1\}$ we will yield the total computational savings of a non-leader u^{nl} for the decentralized caching system.

Theorem 11. *In decentralized caching scheme with uncoded prefetching, a non-leader using (4.17) to compute the requested subfiles $V_{d_{u^{nl},A_{t+1}\setminus\{u^{nl}\}}}$ corresponding to all non-leader $(j+1)$-subsets $A_{j+1} \in \mathcal{A}_{j+1}^{nl}$ for all $j \in \{1, 2, \ldots, K - N_e(\mathbf{d}) - 1\}$ will have a total computational saving of*

$$S_{c,nl}^{dec}(\mathbf{d}) = (K - N_e(\mathbf{d}) - 1)\left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} \frac{MF}{N} + o(F) \tag{4.27}$$

XOR operations (a.a.s.).

Proof. This quantity equals the term S_2 in the proof of theorem 7 and is given by (3.33) after repositioning the term $o(F)$ that was stripped away for simplicity. □

Again, since there are $K - N_e(\mathbf{d})$ non-leaders, we have the following corollary.

Corollary 3. *The (a.a.s.) total computational savings of using (4.17) throughout a decentralized caching system with uncoded prefetching will be*

$$S_c^{dec}(\mathbf{d}) = (K - N_e(\mathbf{d}))\,(K - N_e(\mathbf{d}) - 1)\left(1 - \frac{M}{N}\right)^{N_e(\mathbf{d})+1} \frac{MF}{N} + o(F) \tag{4.28}$$

XOR operations relative to ITODM.

Chapter 5

Comparison

5.1 Centralized Caching

In this section, we would like to compare the efficiency of the centralized ITODM scheme with the one we propose in this work. The problem with (3.13, 3.18) is that, although they fully capture the computational cost involved with a particular demand, they implicate quite a big number of parameters, namely K, t, $N_e(\mathbf{d})$ and the sizes of all pure tails $|Q_{u_i}|$ for $u_i \in \mathcal{U}$. This makes any comparison between the two methods using just these expressions overly specific and quite subjective (pending on the parameter choices one opts for) to bare any particular significance. Thus we would like to have a more objective criterion and an expression that would be more representative of the big number of different values S can take.

One approach to do this would be to average all the pure tail sizes out. In other words, we can take into account all the different choices we have for the parameters $|Q_{u_i}|$ and then calculate the average value of S. In order to do so, however, we first need to prove a very useful lemma for our further analysis.

Lemma 2.

$$\mathcal{S} = \sum_{\substack{(q_1,\dots,q_n)\in\mathbb{N}^n \\ q_1+\dots+q_n=N}} q_1 \dots q_k = \binom{N+n-1}{n-1+k} \qquad , k \leq n. \tag{5.1}$$

Proof. Let us start with two sets. The first is the set of n-tuples of non-negative integers whose sum is N that lead to non-zero summands in (5.1)

$$A = \left\{ (q_1, \dots, q_n) \in \mathbb{N}^n : \begin{array}{l} q_1 + \dots + q_n = N, \\ q_1 \geq 1, \dots, q_k \geq 1 \end{array} \right\}. \tag{5.2}$$

The second is the set of $(n+k)$-tuples of non-negative integers whose sum is $N-k$.

$$B = \left\{ (q_1, \dots, q_{n+k}) \in \mathbb{N}^{n+k} : q_1 + \dots q_{n+k} = N - k \right\}. \tag{5.3}$$

We should note that in order for the sum of (5.1) to be non-zero, there must be at least one term $q_{1,0} \dots q_{k,0} \neq 0$. That means $q_{1,0} \geq 1, \dots, q_{k,0} \geq 1$. Since

$$\underbrace{q_{1,0} + \dots + q_{k,0}}_{\geq k} + \underbrace{q_{k+1,0} + \dots + q_{n,0}}_{\geq 0} = N,$$

we get $k \leq N$, which shows that the above definition of set B is always meaningful (in the sense of it always being non-empty).

Now we can define a function $f : B \to A$ such that if

$$b = (q_1, \ldots, q_n, q_{n+1}, \ldots, q_{n+k}) \in B,$$

then

$$f(b) = (q_1 + q_{n+1} + 1, \ldots, q_k + q_{n+k} + 1, q_{k+1}, \ldots, q_n) \in A.$$

It is easy to see that the image $f(b)$ lies within A by a simple addition of its components giving N.

We further show that function f is a surjection. Let

$$a = (q_1, \ldots, q_k, q_{k+1}, \ldots, q_n) \in A.$$

Then, if we take

$$b = (q_1 - 1, \ldots, q_k - 1, q_k, \ldots, q_n, 0, \ldots, 0) \in B,$$

it is easy to see that $f(b) = a$. Because $q_i \geq 1$ for $i \in \{1, \ldots, k\}$ b is guaranteed to belong to B.

Since f is a surjection, we know that the sets

$$f^{-1}(a) = \{b \in B : f(b) = a\} \tag{5.4}$$

are equivalence classes of B. In particular, they form the quotient set with respect to the equivalence relation $b_1 \sim b_2 \Leftrightarrow f(b_1) = f(b_2)$.

Suppose now that

$$a = (q_1^a, \ldots, q_k,^a q_{k+1}^a, \ldots, q_n^a) \in A,$$

and we would like to characterize all the $b \in B$ that belong to $f^{-1}(a)$. The general form of a $b \in B$ is

$$b = (q_1, \ldots, q_k, q_{k+1}, \ldots, q_n, i_1, \ldots, i_k).$$

The expression $f(b) = a$ imposes the following restrictions

$$\begin{cases} q_1 = q_1^a - 1 - i_1 \\ \vdots \\ q_k = q_k^a - 1 - i_k \\ q_{k+1} = q_{k+1}^a \\ \vdots \\ q_n = q_n^a. \end{cases} \tag{5.5}$$

Here we can make three observations. First, that these conditions show that only the quantities i_j for $j \in \{1, \ldots, k\}$ are actually variable. Furthermore, the value of each such i_j can be selected independently from the others from the range $i_j \in \{0, 1, \ldots, q_j^a - 1\}$. The lower end comes from the fact that $i_j \geq 0$ and the upper end from $q_j \geq 0$. Secondly, that different choices lead to different $b \in f^{-1}(a)$ and thirdly that for any $b \in f^{-1}(a)$ there is a unique choice of $i_1, \ldots, i_k$ giving the above form.

These three observations show that there are $|f^{-1}(a)| = q_1^a \ldots q_k^a$ elements in $f^{-1}(a)$, which are the degrees of freedom in the above system of equations.

Now, we can go back to the sum in (5.1) where we can limit the range to the non-zero terms and write

$$\mathcal{S} = \sum_{a=(q_1,\ldots,q_n)\in A} q_1 \ldots q_k = \sum_{a \in A} |f^{-1}(a)| = |B|. \tag{5.6}$$

But, since B is the set of all $(n+k)$-tuples of non-negative integers whose sum is $N-k$ its size is given by [78]

$$|B| = \binom{N+n-1}{n-1+k}. \tag{5.7}$$

completing the proof. □

We are know equipped with the result that will enable us to perform an averaging over the pure tail sizes of (3.18). First of all, we should note that the number of ways we can choose the pure tail sizes equals the number of ways we can distribute the $K - N_e(\mathbf{d}) - 1$ non-leaders into $N_e(\mathbf{d})$ sets. In other words, it is the number of $N_e(\mathbf{d})$-tuples of non-negative integers whose sum is $K - N_e(\mathbf{d}) - 1$. This is a well know combinatorics problem and the answer can be proven [78] to be $\binom{K-2}{N_e(\mathbf{d})}$.

Then we have to find the value of the sum of each product term in (3.18) as the pure tail sizes move over their entire range. In particular, we have to find the following sums

$$\sum_{\substack{\left(|Q_{u_1}|-1,|Q_{u_{i_2}}|,\ldots,|Q_{u_{i_l}}|\right)\in R_q^l \\ |Q_{u_1}|+|Q_{u_{i_2}}|+\cdots+|Q_{u_l}|=K-N_e(\mathbf{d})}} (|Q_{u_1}|-1)\,|Q_{u_{i_2}}|\ldots|Q_{u_{i_l}}| = \sum_{\substack{\left(q_1,\ldots,q_{N_e(\mathbf{d})}\right)\in\mathbb{N}^l \\ q_1+\cdots+q_{N_e(\mathbf{d})}=K-N_e(\mathbf{d})-1}} q_1\ldots q_l, \tag{5.8}$$

$$\sum_{\substack{\left(|Q_{u_{i_1}}|,\ldots,|Q_{u_{i_l}}|\right)\in R_q^l \\ |Q_{u_{i_1}}|+\cdots+|Q_{u_l}|=K-N_e(\mathbf{d})-1}} |Q_{u_{i_1}}|\ldots|Q_{u_{i_l}}| = \sum_{\substack{\left(q_1,\ldots,q_{N_e(\mathbf{d})}\right)\in\mathbb{N}^l \\ q_1+\cdots+q_{N_e(\mathbf{d})}=K-N_e(\mathbf{d})-1}} q_1\ldots q_l, \tag{5.9}$$

where

$$R_q = \{0,\ldots,K-N_e(\mathbf{d})-1\}. \tag{5.10}$$

In these two sums we make the substitutions $q_i = |Q_{u_{i_i}}|$ with the only exception of $q_1 = |Q_{u_1}| - 1$ for (5.8). The generalization from R_q to $\mathbb{N}$ is valid since it does not introduce any additional non-zero summands. So we actually see that for the same l these sums are in effect equal. Their value is given by the formula in Lemma 2 that we proved above which is equal to

$$\sum_{\substack{\left(q_1,\ldots,q_{N_e(\mathbf{d})}\right)\in\mathbb{N}^l \\ q_1+\cdots+q_{N_e(\mathbf{d})}=K-N_e(\mathbf{d})-1}} q_1\ldots q_l = \binom{K-2}{N_e(d)-1+l} \tag{5.11}$$

Thus the only difference between the two sums in (3.18), apart from the multiplication with two in the second sum, is the number of product terms that are being summed. The first, is the sum of products of l terms where the first term is always $|Q_{u_1}| - 1$. Thus the number of these product terms is equal to the ways we can choose $l-1$ things out of $N_e(\mathbf{d}) - 1$, or $\binom{N_e(\mathbf{d})-1}{l-1}$. Similarly, for the second sum, since it is the sum of products of l terms whose selection excludes $|Q_{u_1}|$ their number will be equal to the ways we can select l things out of $N_e(\mathbf{d})-1$ or $\binom{N_e(\mathbf{d})-1}{l}$.

So taking the average and replacing the above while doing some manipulations, we get

$$\begin{aligned}\bar{S} &= \sum_{l=0}^{\min(t,N_e(\mathbf{d}))} \binom{K-N_e(\mathbf{d})-1-l}{t-l} \frac{\binom{K-2}{N_e(\mathbf{d})-1+l}}{\binom{K-2}{N_e(\mathbf{d})-1}} \left(\binom{N_e(\mathbf{d})-1}{l-1} + 2\binom{N_e(\mathbf{d})-1}{l}\right) \\ &= \frac{(N_e(\mathbf{d})-1)!(K-N_e(\mathbf{d})-1)!}{(K-N_e(\mathbf{d})-1-t)!} \sum_{l=0}^{N_e(\mathbf{d})} \frac{\binom{N_e(\mathbf{d})}{l} + \binom{N_e(\mathbf{d})-1}{l}}{(t-l)!(N_e(\mathbf{d})-1+l)!}.\end{aligned} \tag{5.12}$$

Note that in this expression, we choose to use $N_e(\mathbf{d})$ as the upper limit of the sum, instead of $\min(t, N_e(\mathbf{d}))$, because it will make our derivations more natural. This does not change the

end result, as long as the "out of bounds" terms are taken to be zero, as they should. Also, in the above manipulations, we have used the property

$$\binom{N_e(\mathbf{d})-1}{l-1}+\binom{N_e(\mathbf{d})-1}{l}=\binom{N_e(\mathbf{d})}{l}. \tag{5.13}$$

In this handling, some extra care is warranted towards the first and the last terms of the sum to make sure that the zero-valued "out of bounds" terms appearing do not lead to different results.

We now focus on the following sums

$$\bar{S}_1=\sum_{l=0}^{N_e(\mathbf{d})}\frac{\binom{N_e(\mathbf{d})}{l}}{(t-l)!(N_e(\mathbf{d})-1+l)!}, \tag{5.14}$$

$$\bar{S}_2=\sum_{l=0}^{N_e(\mathbf{d})}\frac{\binom{N_e(\mathbf{d})-1}{l}}{(t-l)!(N_e(\mathbf{d})-1+l)!}. \tag{5.15}$$

Expanding the factorials and rearranging the multiplications, we can see that after a suitable pairing of the resulting terms, these sums can be expressed as

$$\begin{aligned}\bar{S}_1&=\frac{N_e(\mathbf{d})!}{t!(2N_e(\mathbf{d})-1)!}\sum_{l=0}^{N_e(\mathbf{d})}\binom{t}{l}\binom{2N_e(\mathbf{d})-1}{N_e(\mathbf{d})-l}\\&=\frac{N_e(\mathbf{d})!}{t!(2N_e(\mathbf{d})-1)!}\binom{2N_e(\mathbf{d})-1+t}{N_e(\mathbf{d})}\\&=\frac{1}{(N_e(\mathbf{d})-1+t)!}\binom{2N_e(\mathbf{d})-1+t}{t},\end{aligned} \tag{5.16}$$

$$\begin{aligned}\bar{S}_2&=\frac{(N_e(\mathbf{d})-1)!}{t!(2N_e(\mathbf{d})-2)!}\sum_{l=0}^{N_e(\mathbf{d})}\binom{t}{l}\binom{2N_e(\mathbf{d})-2}{N_e(\mathbf{d})-1-l}\\&=\frac{(N_e(\mathbf{d})-1)!}{t!(2N_e(\mathbf{d})-2)!}\binom{2N_e(\mathbf{d})-2+t}{N_e(\mathbf{d})-1}\\&=\frac{1}{(N_e(\mathbf{d})-1+t)!}\binom{2N_e(\mathbf{d})-2+t}{t}.\end{aligned} \tag{5.17}$$

In the above, we have used Vandermonde's identity as well as a proper pairing of the terms appearing in the binomial coefficients. Plugging back these results to (5.12) which is the sought after averaged value of S.

$$\bar{S}=\frac{\binom{K-2}{N_e(\mathbf{d})-1+t}}{\binom{K-2}{N_e(\mathbf{d})-1}}\left[\binom{2N_e(\mathbf{d})-1+t}{t}+\binom{2N_e(\mathbf{d})-2+t}{t}\right] \tag{5.18}$$

We must stress here that this expression is the average over all values of S for the different choices of $|Q_{u_i}|$ and leads to the corresponding average of the computational cost of a non-leader with respect to these values. It does not lead to the expected computational cost of a non-leader. The reason for this is that in deriving (5.18) we did not account for the multiplicity of the different requests leading to the same pure tail sizes. Also, such an expected value would require the assumption of a particular scheme by which leaders and non-leaders are assigned during reception. Only then would one be able to compute the expectation with respect to

that particular scheme. We see that seeking for an expected computational cost characterization not only makes the analysis more complicated but also re-introduces a particular level of subjectivism (the choice of the assignment scheme) that we are trying to avoid.

Nevertheless, one might be interested in the worst-case scheme where whenever a user can be a non-leader, it is done so and then try to calculate the expected computational costs for such a non-leader. This might lead to a useful characterization of the computational cost and could be the object of a future analysis.

In order to compare ITODM with our proposed method, we calculate the relative computational improvement, defined as

$$a_c = \frac{S_{c,nl}(\mathbf{d})}{\bar{C}_{c,nl}}. \tag{5.19}$$

The numerator is the total computational savings coming from using our proposed method for a particular demand $\mathbf{d}$, given in (4.25). The denominator is the average computational cost of the ITODM for the particular K, t and $N_e(\mathbf{d})$. This average computational cost is given by (3.19) if we replace the S term with its average $\bar{S}$ given in (5.18). Note that the numerator also depends only on K, t and $N_e(\mathbf{d})$, making it suitable for our comparison. We do not mention the dependence with respect to F as it is canceled out.

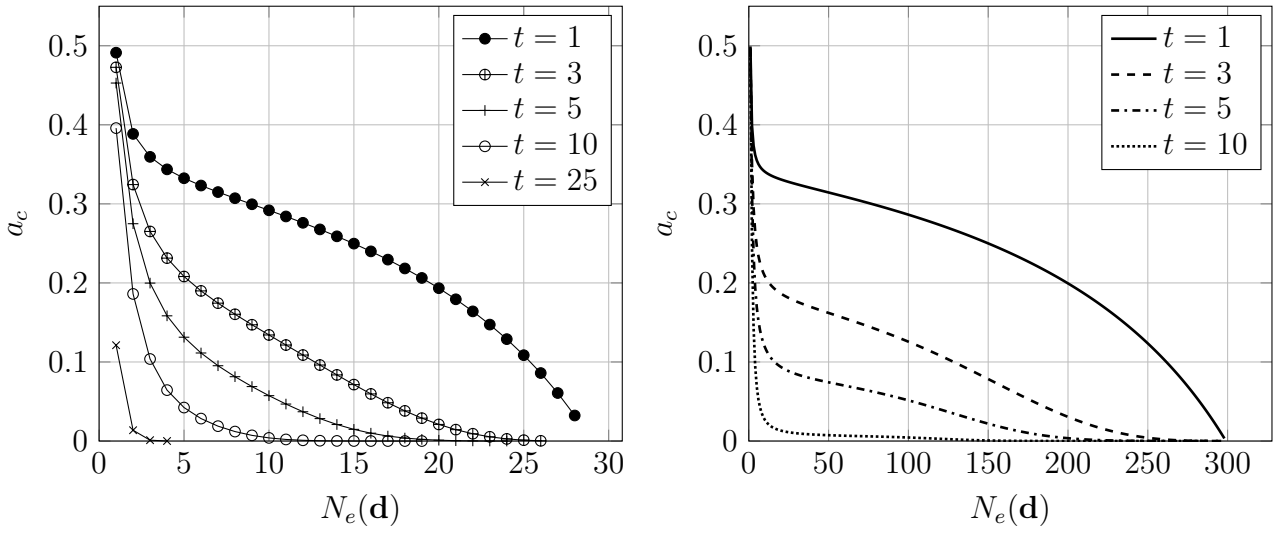
First, we will examine a small user case. Supposing we have $K = 30$ users and plotting a_c against $N_e(\mathbf{d})$ for various values of t, we get the results displayed in Fig. 5.1a. Note that the different plots have different endpoints. That is because, for a specific K and a specific t, $N_e(\mathbf{d})$ is allowed to vary up to $K-t-1$ that is the highest value for which there are non-transmitted subsets and the ITODM can profit from our proposed method.

We observe here a significant computational improvement for most kinds of requests. As the number of users t sharing the same information gets higher and as the number of distinct file requests $N_e(\mathbf{d})$ increases, this improvement becomes lower. Overall, we can see that for the small user case the proposed method can contribute significantly to the reduction of the computational cost of decoding the requested subfiles.

Next, we will examine the other end, which is a large user case. If we set $K = 300$ and plot a_c against $N_e(\mathbf{d})$ for various values of t we will get Fig. 5.1b. The first impression here is similar to the small user case. We have high gains for the smaller values of t and $N_e(\mathbf{d})$ that decrease as these parameters get higher. The important difference is that while in the small user case, the parameter t could get as high as $K-2 = 28$ (otherwise there are no non-transmitted subfiles), in the large user case, the parameter t can get as high as $K-2 = 298$. That means that only for a small region of arrangements, we actually have significant computational gains. This region contains the cases where the total cache memory of the users (KMF) is not much larger than the library size (NF).

So we can conclude that even for a large user set, the proposed computational method yields significant computational gains as long as the total cache memory of the system remains close in scale with the total library size (keeping the parameter t below 10). For larger total user cache sizes, the sheer amount of the transmissions that take place in the derivation of the non-transmitted subfiles is so big, that overshadows any computational benefit their direct computation offered by our method has, compared to ITODM.

Another interesting comparison we can make is to study the improvement in the computational cost related to the derivation of the non-transmitted files from the transmitted ones. The difference is that this quantity does not include the cost for decoding the transmitted subfiles and can give us a more direct glimpse of the improvement in the actual calculation that takes place with respect to this derivation. Also, for the same reason, this quantity can be thought



(a) Small User Case ($K = 30$). (b) Large User Case ($K = 300$).

Figure 5.1: Relative computational improvement of a non-leader when fully decoding their requested file. For the small user case, a significant improvement is observed for most kinds of requests. For the large user case, we have a significant improvement in a region where the aggregate cache memory of the users is close in scale (less than 10 times) to the size of the library.

of as an upper bound for a_c. We can define this relative improvement as

$$a_c^{nt} = \frac{S_{c,nl}(\mathbf{d})}{\bar{C}_{c,nl}^{nt}}. \tag{5.20}$$

Again, the numerator is given by (4.25) and represents the total computational gains of the proposed method and the denominator is the average of the computational cost of deriving the non-transmitted subfiles from the transmitted ones for specific K, t and $N_e(\mathbf{d})$. This quantity is given by (3.12) by replacing the term S with its average $\bar{S}$ given in (5.18).

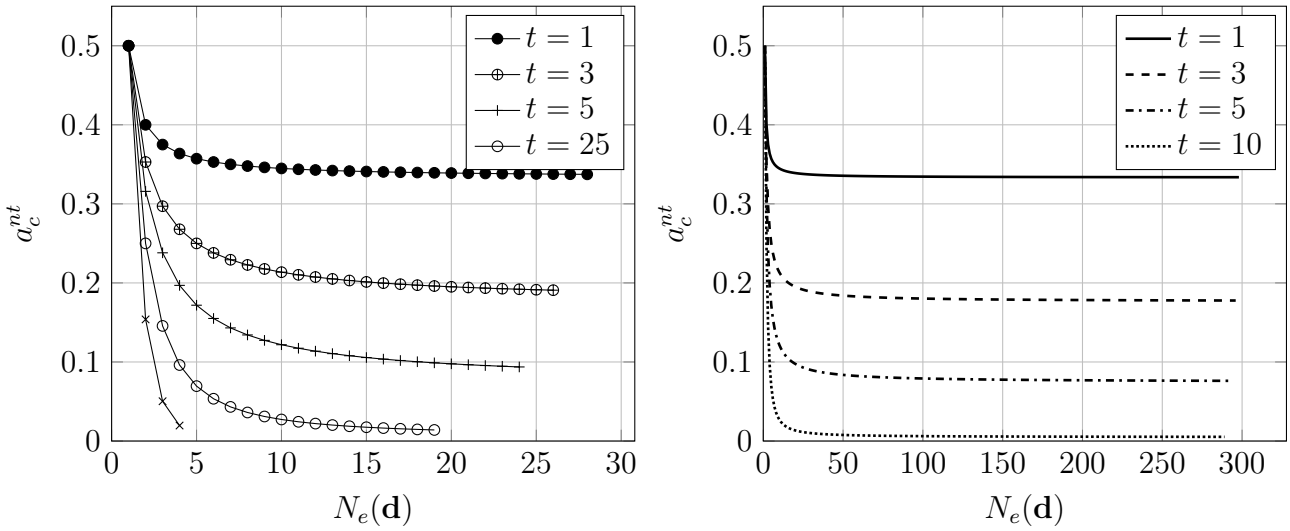
Plotting a_c^{nl} for the small and large user cases we had before, gives us the results presented in Fig. 5.2a and 5.2b. The main characteristics are similar to the previous plots of a_c and the comments we did there apply here as well. However, a major effect that we did not observe in a_c is the presence on an asymptotic behavior.

What these graphs show is that as K gets larger, increasing the number of distinct requests $N_e(\mathbf{d})$ leads to a steady, non-vanishing improvement in the computational cost of deriving the non-transmitted subfiles. We can find an expression for this asymptotic value by letting K go to infinity while replacing $N_e(\mathbf{d})$ with its end value $K - t - 1$. Doing so and using's Stirling's approximation [75] for the factorial leads us, after some straight forward manipulations, to the following expression

$$a_{c,l}^{nt} = \frac{t}{2^{t+1}}, \tag{5.21}$$

which corresponds to the exponential drop in the relative computational improvement we see in these figures and contributing to our understanding of the rapid decrease of the total computational improvement we observed in Fig. 5.1b for the large user case where this asymptotic behavior gets a chance to be fully expressed.

The fact that XOR operations are readily translatable to energy demands allows us to directly translate all the previous computational improvements to corresponding improvements



(a) Small User Case ($K = 30$). (b) Large User Case ($K = 30$).

Figure 5.2: Relative computational improvement of a non-leader when deriving their untransmitted subfiles from the transmitted ones. The major new characteristic we observe here is the presence of an asymptotic behavior having an exponential drop with respect to the parameter t.

in the system's energy profile. In other words, we can view a_c and a_c^{nt} either as relative computational improvements or as relative improvements if energy efficiency. We would like to close this subsection by performing one more comparison that will allow us to appreciate the impact of the proposed method on the energy efficiency of the whole system.

In particular, we will examine the following relative improvement

$$r_c = \frac{S_c(\mathbf{d})}{\bar{C}_{c,t}}. \tag{5.22}$$

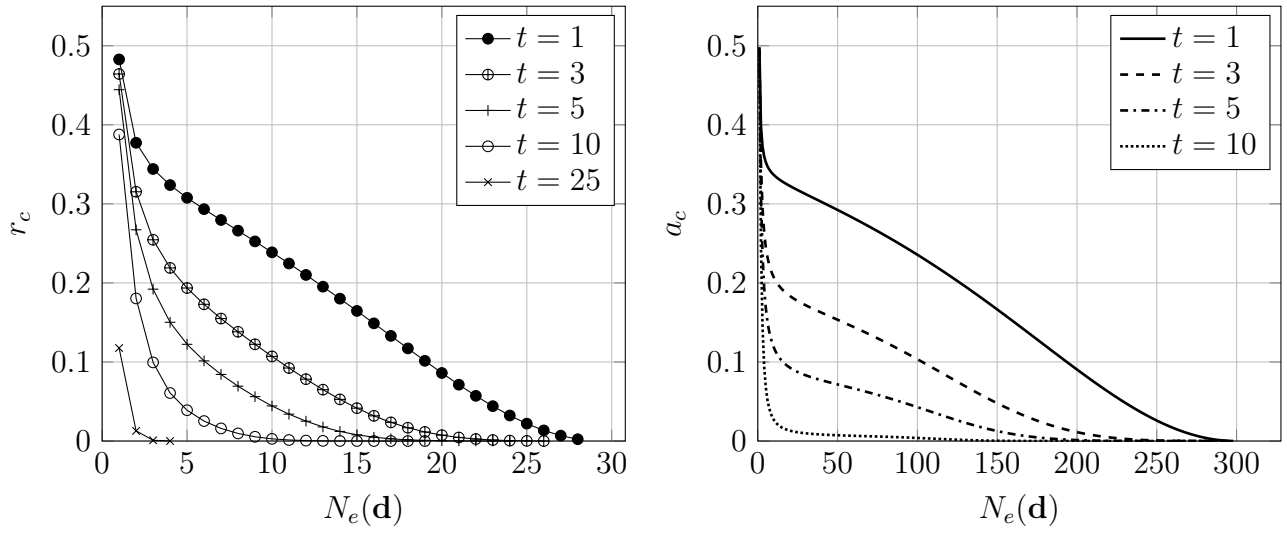
The numerator is the total computational savings coming from the proposed method throughout the system, given by (4.26). The denominator is the total computational cost among all leaders and non-leaders, and is given by

$$\bar{C}_{c,t} = N_e(\mathbf{d})C_{c,l} + (K - N_e(\mathbf{d}))\bar{C}_{c,nl}, \tag{5.23}$$

where $C_{c,l}$ is given by (3.1) or (3.2) and $\bar{C}_{n,nl}$ by (3.19) if we replace the S term with its average $\bar{S}$ given in (5.18). Note that, as we did before, the computational cost for a non-leader is averaged out with respect to the pure tail sizes.

This relative improvement is not very interesting from a computational point of view, as it expresses an improvement among computations that happen in parallel among the different users. However, it is very interesting from the energy consumption point of view, as is expresses the relative improvement in the energy consumed by the users as they perform their decoding task, which is an important figure of merit in the system's energy profile.

Plotting r_c against $N_e(\mathbf{d})$ for the small (K=30) and large (K=300) user cases gives us the results displayed in Fig. 5.3a and 5.3b. We observe that r_c displays a similar behavior as a_c. In particular, in the small user case, the relative energy consumption has a significant improvement for most demands. It becomes lower as the number of users t sharing the same information gets higher and as the number of distinct requests $N_e(\mathbf{d})$ increases. In the large



(a) Small User Case ($K = 30$).

(b) Large User Case ($K = 300$).

Figure 5.3: Relative improvement in the energy consumed by the users during the delivery phase. The behavior here is similar to a_c. For the small user case, we have a significant improvement for most kinds of request while for the large user case, we have a significant improvement in a region where the aggregate cache memory of the users is close in scale (less than 10 times) to the size of the library.

user case, the improvement is more pronounced in the region of small t corresponding to the cases where the total cache memory of the users (KMF) is not much larger than the library size (NF).

This similarity in behavior between r_c and a_c was expected since the system is naturally expected to benefit more when the separate non-leader users benefit more and vice versa, which is another expression of the imbalance in the computational burden between the leaders and the non-leaders.

5.2 Decentralized Caching

Like we did for centralized caching, we would like a more representative expression of the different values $C_{c,nl}^{dec}$ and $C_{c,nl}^{dec,nt}$ can take when the pure tail sizes $|Q_{u_i}|$, with $u_i \in \mathcal{U}$, range among their possible choices for specific K, $N_e(\mathbf{d})$, M, N and F. As we discussed in the previous section one way to do it would be to average out the pure tail sizes in the term

$$S_{dec} = \left(\frac{M}{N}(|Q_{u1}| - 1) + 2\right) \prod_{i=2}^{N_e(\mathbf{d})} \left(\frac{M}{N}|Q_{u_i}| + 1\right). \tag{5.24}$$

To derive this average, first of all, we unfold the product terms by performing the multiplications in (5.24) and get

$$S_{dec} = \sum_{l=0}^{m} \mathcal{Q}(l) \left(\frac{M}{N}\right)^l, \tag{5.25}$$

where $\mathcal{Q}(l)$ is given by (3.25). We can repeat now the reasoning of section 5.1 to acquire the

average of each $\mathcal{Q}(l)$. Doing so and plugging the result back to (5.25) we get

$$\bar{S}_{dec} = \underbrace{\sum_{l=0}^{m} \left(\frac{M}{N}\right)^{l} \frac{\binom{K-2}{N_e(\mathbf{d})-1+l}}{\binom{K-2}{N_e(\mathbf{d})-1}} \binom{N_e(\mathbf{d})}{l}}_{S_a} + \underbrace{\sum_{l=0}^{m} \left(\frac{M}{N}\right)^{l} \frac{\binom{K-2}{N_e(\mathbf{d})-1+l}}{\binom{K-2}{N_e(\mathbf{d})-1}} \binom{N_e(\mathbf{d})-1}{l}}_{S_b}, \tag{5.26}$$

We can now write S_a as

$$S_a = \sum_{l}^{m} \left(\frac{M}{N}\right)^{l} \frac{\binom{N_e(\mathbf{d})}{l}\binom{K-2}{K-N_e(\mathbf{d})-1-l}}{\binom{K-2}{N_e(\mathbf{d})-1}}, \tag{5.27}$$

and recognize the expression as the sum of the probability generating function for the hypergeometric distribution [79], after we substitute M/N by z. Thus we get

$$S_a = {}_2F_1\left(\begin{matrix}-N_e(\mathbf{d}), -K+N_e(\mathbf{d})+1 \\ N_e(\mathbf{d})\end{matrix}; \frac{M}{N}\right). \tag{5.28}$$

Similarly, we can do the same for S_b after we write it in the form

$$S_b = \sum_{l}^{m} \left(\frac{M}{N}\right)^{l} \frac{\binom{N_e(\mathbf{d})-1}{l}\binom{K-2}{K-N_e(\mathbf{d})-1-l}}{\binom{K-2}{N_e(\mathbf{d})-1}}, \tag{5.29}$$

and get

$$S_b = {}_2F_1\left(\begin{matrix}-N_e(\mathbf{d})+1, -K+N_e(\mathbf{d})+1 \\ N_e(\mathbf{d})\end{matrix}; \frac{M}{N}\right). \tag{5.30}$$

Using these results in (5.26) we reach (5.31) which is the average we are looking for.
Doing so will give us the following result:

$$\begin{aligned}\bar{S}_{dec} = {}&{}_2F_1\left(\begin{matrix}-N_e(\mathbf{d}), -K+N_e(\mathbf{d})+1 \\ N_e(\mathbf{d})\end{matrix}; \frac{M}{N}\right) \\ &+ {}_2F_1\left(\begin{matrix}-N_e(\mathbf{d})+1, -K+N_e(\mathbf{d})+1 \\ N_e(\mathbf{d})\end{matrix}; \frac{M}{N}\right).\end{aligned} \tag{5.31}$$

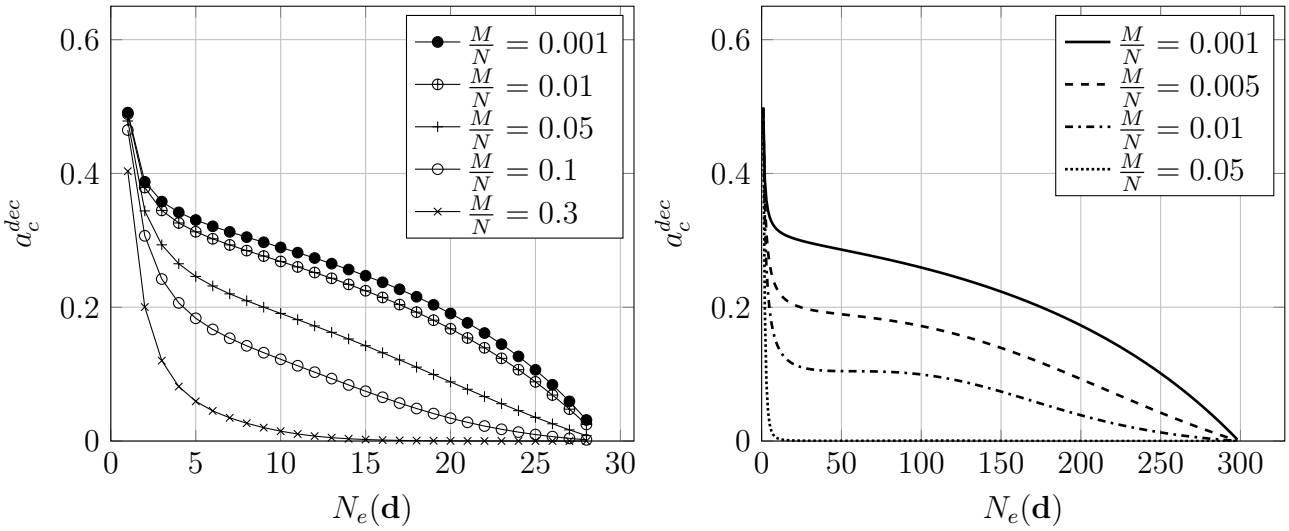
In this expression, ${}_2F_1$ is the Gaussian hypergeometric function [79].

We proceed now to examine the relative computational improvement as we did for centralized caching. Again, we can define this quantity to be

$$a_c^{dec} = \frac{S_{c,nl}^{dec}(\mathbf{d})}{\bar{C}_{c,nl}^{dec}}. \tag{5.32}$$

Here, $S_{c,nl}^{dec}(\mathbf{d})$ is the computational improvement of our proposed method for the decentralized caching, given by (4.27) and $\bar{C}_{c,nl}^{dec}$ the average computational cost of the decentralized ITODM for the particular K, M, N and $N_e(\mathbf{d})$. This quantity is given by (3.35) by replacing the term S_{dec}, as given by (5.24), with its average $\bar{S}_{dec}$ given in (5.31). As before, we do not take the dependence on F into account, as for adequately large values, it is practically canceled out.

Examining first the small user case ($K = 30$), plotting a_c^{dec} against $N_e(\mathbf{d})$ for different user-over-library (M/N) ratios, we get the results displayed in Fig. 5.4a, where we see some quite different behavior from what we had in centralized caching. What we observe here is



(a) Small User Case ($K = 30$). (b) Large User Case ($K = 300$).

Figure 5.4: Relative computational improvement of a non-leader when fully decoding their requested file. We observe that as long as the individual user caches remain small compared to the total library size, there is a significant computational improvement for practically the whole range of demands.

that as long as the individual user caches MF are small compared to the total library size NF, our proposed method yields significant computational benefits compared to the decentralized ITODM for all kinds of demands.

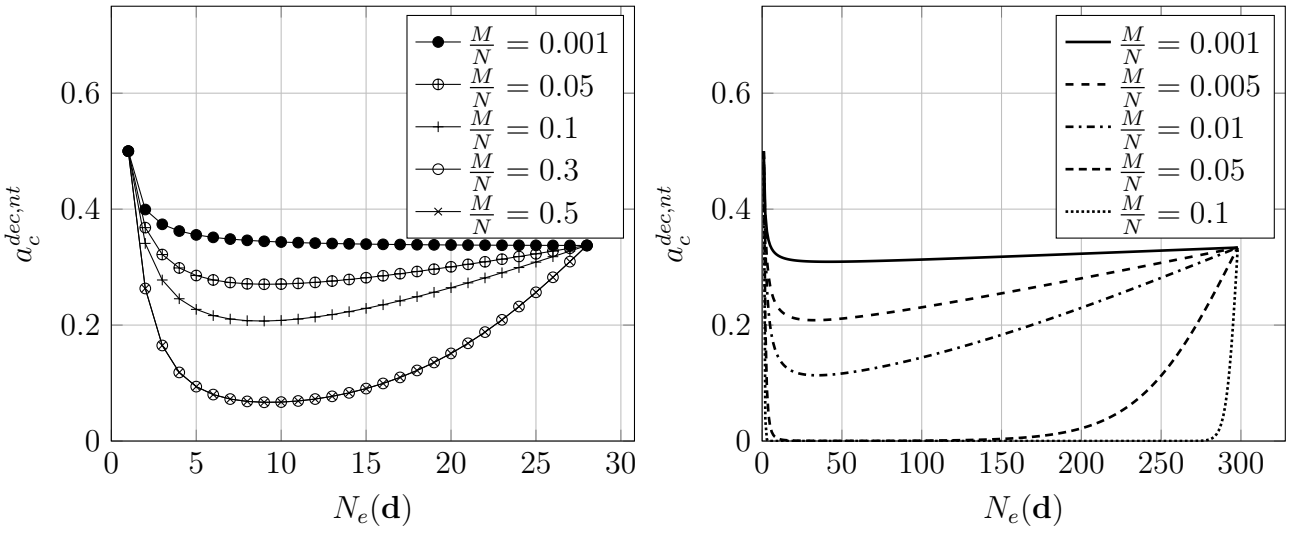
In Fig. 5.4b we plot the a_c^{dec} for the large user case ($K = 300$) and we observe the same behavior. As long as the user cache size MF remains low compared to the total library size NF, our proposed method yields significant computational gains for all kinds of demands. What these results further illustrate is that as the user count K becomes higher, the computational gains slowly decrease. In other words, the range of values for M/N for which our method provides significant computational gains becomes smaller as the number of users increases. However, given that in most typical scenarios, the user caches are quite smaller compared to the total library size of the system, our proposed method still provides significant computational gains practically for the whole range of $N_e(\mathbf{d})$.

As we did in the previous section, we can also compare the computational improvement with respect to the computational cost of deriving the untransmitted subfiles from the transmitted ones. We can define this relative computational improvement as

$$a_c^{dec,nt} = \frac{S_{c,nl}^{dec}(\mathbf{d})}{\bar{C}_{c,nl}^{dec,nt}}. \tag{5.33}$$

The numerator here is the same as that in (5.32) and the denominator is the average computational cost for deriving the transmitted from the untransmitted subfiles for specific values of K, M, N and $N_e(\mathbf{d})$. This quantity is given by (3.22) by replacing the term S_{dec}, as given by (5.24), with its average $\bar{S}_{dec}$ given in (5.31).

Plotting $a_c^{dec,nt}$ for the small user case ($K = 30$) against $N_e(\mathbf{d})$ for different values of the user-to-library (M/N) ratio we get Fig. 5.5a. Again, we observe that we have significant computational gains in the whole range for $N_e(\mathbf{d})$ and we see an important difference. As $N_e(\mathbf{d})$ increased, the relative computational improvement $a_c^{dec,nt}$ starts from value 0.5, decreases and the increases returning to value (asymptotically) equal to 1/3. So, if we are interested in this



(a) Small User Case ($K = 30$). (b) Large User Case ($K = 300$).

Figure 5.5: Relative computational improvement of a non-leader when deriving their untransmitted subfiles from the transmitted ones. We observe that as long as the individual user caches remain small compared to the library size, we have a significant non-vanishing computational gain that asymptotically converges to the same value.

kind of computational improvement, we can say that we have non-vanishing computational gains, as long as the user caches size MF remains small with respect to the total library size NF.

As we can see in Fig. 5.5b, where we plot the same quantity for the large user case ($K = 300$), this significant gain remains for all kinds of user demand, albeit for a somewhat smaller but still quite larger than the typical range of M/N ratio values.

Finally, we would like to examine the relative improvement in energy consumption for the system as a whole, as we did in the previous section for centralized caching. The corresponding relative improvement for decentralized caching will be

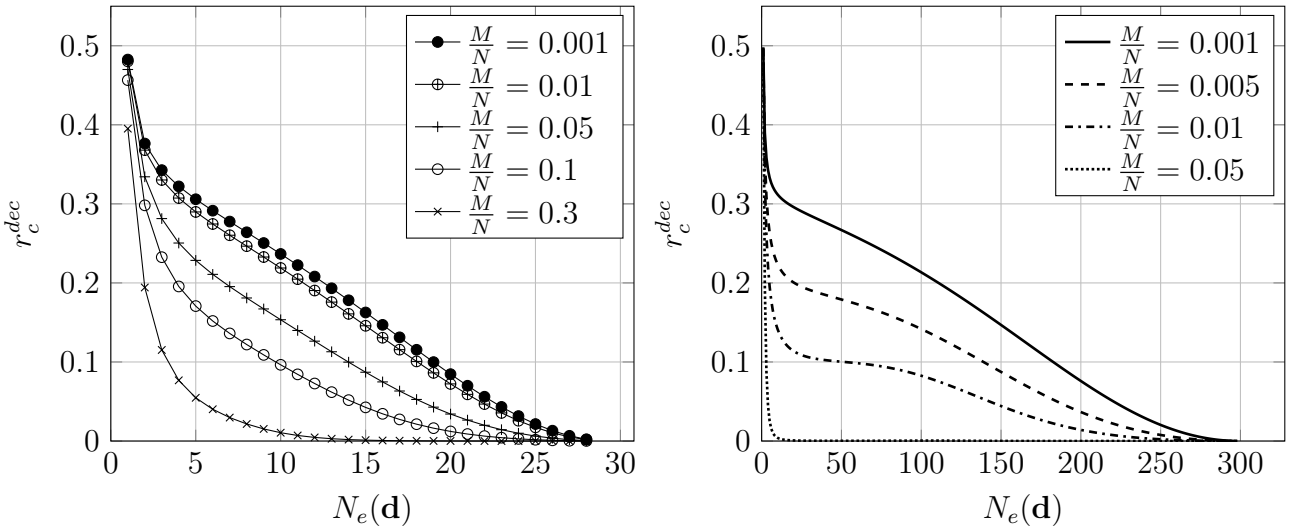
$$r_c^{dec} = \frac{S_c^{dec}(\mathbf{d})}{\bar{C}_{c,t}^{dec}}. \tag{5.34}$$

The numerator is given by (4.28) and represents the total computational savings the proposed method provides to the system. The denominator is the total computational cost among all users and is given by

$$\bar{C}_{c,t}^{dec} = N_e(\mathbf{d}) C_{c,l}^{dec} + (K - N_e(\mathbf{d}))\bar{C}_{c,nl}^{dec}, \tag{5.35}$$

where $C_{c,l}^{dec}$ is given by (3.20) and $\bar{C}_{c,nl}^{dec}$ is given by (3.35) by replacing the term S_{dec}, as given by (5.24), with its average $\bar{S}_{dec}$ given in (5.31). We should note again here that the computational cost for a non-leader is averaged out with respect to the pure tail sizes.

Examining the small ($K = 30$) and large ($K = 300$) user cases by plotting r_c^{dec} against $N_e(\mathbf{d})$ for various user-over-library (M/N) ratios, we get the results displayed in Fig. 5.6a and 5.6b. As with centralized caching, the behavior of r_c^{dec} closely resembles that of a_c^{dec}. In particular, we observe significant improvements in energy consumption among the users for almost all kinds of requests, both in the small as well as the large user case. The only condition is that the user-over-library ratio remains small, with the actual range becoming smaller as the number of users increases.



(a) Small User Case ($K = 30$). (b) Large User Case ($K = 300$).

Figure 5.6: Relative improvement in the energy consumed by the users during the delivery phase. The behavior of r_c^{dec} closely resembles that of a_c^{dec} having significant improvements in energy consumption among the users, as long as the individual cache size of the users remains small in comparison to the library size.

What the analysis of this section shows is that our proposed method provides significant computational and energy-related advantages over the decentralized ITODM for all kinds of system arrangements and demands, as long as the individual user cache size MF is small when compared to the total library size NF. This is a natural condition expected to hold for almost all caching systems, as the available memory in a user device (UE) is typically much smaller than the memory in a content delivery server or in a base station (BS).

Chapter 6

Conclusions

In this work, we have performed a complete computational analysis of the information-theoretic optimal delivery method (ITODM) for centralized and decentralized caching, the two fundamental methods of coded caching, a technology that is expected to play a key role in future 5G networks assisting them in managing their increased complexity and big data challenges. Both methods take advantage of the commonality in the file requests among the users to reduce the telecommunication load down to the information-theoretic optimal level. However, this is done at an exponentially increasing computational cost. Thus, our analysis allowed us to specify the exact amount of this computational cost down to the number of XOR operations required. This is an important figure of merit not only because it gives us an exact expression of the computational needs but is also readily translatable to energy demands both for the individual users and the overall system in general.

Furthermore, we have developed an alternative method for the delivery stage of centralized and decentralized caching that provides significant computational and energy consumption improvements over ITODM. This is achieved by introducing a computational shortcut in the derivation phase of the untransmitted subfiles from the transmitted ones. For centralized caching, the improvements are more pronounced when the number of users is small, or the total cache size among the users is comparable in scale to the total library size. For the more realistic case of decentralized caching, however, we observed significant computational improvements for all kinds of scenarios, as long as the individual user cache size remains small in comparison to the total library size, a condition that naturally happens in such systems.

Due to the principal position of centralized and decentralized caching in the domain of coded caching, any improvement or new results regarding it immediately reverberate outwards to all other kinds of coded caching systems. Thus, future research could extend the results of this work to other coded caching systems and could examine other ways and different aspects of characterizing the computational costs involved. Minimizing the computational burden coming from the utilization of commonality and finding alternative schemes to balance out this cost among non-leader is still an open question and we aspire that this work will push the discussion forward.

Bibliography

[1] L. Wei, R. Q. Hu, Y. Qian, and G. Wu, "Key elements to enable millimeter wave communications for 5g wireless systems," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 136–143, 2014.

[2] B. Romanous, N. Bitar, A. Imran, and H. Refai, "Network densification: Challenges and opportunities in enabling 5g," in *2015 IEEE 20th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, 2015, pp. 129–134.

[3] D. Liu, L. Wang, Y. Chen, M. Elkashlan, K. Wong, R. Schober, and L. Hanzo, "User association in 5g networks: A survey and an outlook," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1018–1044, 2016.

[4] M. Erol-Kantarci and S. Sukhmani, "Caching and computing at the edge for mobile augmented reality and virtual reality (ar/vr) in 5g," in *Ad Hoc Networks*, Y. Zhou and T. Kunz, Eds. Cham: Springer International Publishing, 2018, pp. 169–177.

[5] Chih-Ping Li, Jing Jiang, W. Chen, Tingfang Ji, and J. Smee, "5g ultra-reliable and low-latency systems design," in *2017 European Conference on Networks and Communications (EuCNC)*, 2017, pp. 1–5.

[6] K. E. Skouby and P. Lynggaard, "Smart home and smart city solutions enabled by 5g, iot, aai and cot services," in *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, 2014, pp. 874–878.

[7] S. Li, L. D. Xu, and S. Zhao, "5g internet of things: A survey," *Journal of Industrial Information Integration*, vol. 10, pp. 1 – 9, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2452414X18300037

[8] Y. Zhou, L. Tian, L. Liu, and Y. Qi, "Fog computing enabled future mobile communication networks: A convergence of communication and computing," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 20–27, 2019.

[9] P. Lin, K. S. Khan, Q. Song, and A. Jamalipour, "Caching in heterogeneous ultradense 5g networks: A comprehensive cooperation approach," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 22–32, 2019.

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

[11] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug 2015.

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

[13] J. Hachem, N. Karamchandani, and S. Diggavi, "Multi-level coded caching," in *2014 IEEE International Symposium on Information Theory*, June 2014, pp. 56–60.

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

[15] J. Hachem, N. Karamchandani, and S. Diggavi, "Effect of number of users in multi-level coded caching," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 1701–1705.

[16] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 849–869, Feb 2016.

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

[18] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 349–366, Jan 2018.

[19] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3923–3949, June 2017.

[20] A. Ramakrishnan, C. Westphal, and A. Markopoulou, "An efficient delivery scheme for coded caching," in *2015 27th International Teletraffic Congress*, Sep. 2015, pp. 46–54.

[21] M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 809–813.

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

[23] J. Hachem, U. Niesen, and S. Diggavi, "A layered caching architecture for the interference channel," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 415–419.

[24] J. Hachem, U. Niesen, and S. N. Diggavi, "Degrees of freedom of cache-aided wireless interference networks," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 5359–5380, July 2018.

[25] C. Wang, S. H. Lim, and M. Gastpar, "Information-theoretic caching," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 1776–1780.

[26] ——, "Information-theoretic caching: Sequential coding for computing," *IEEE Transactions on Information Theory*, vol. 62, no. 11, pp. 6393–6406, Nov 2016.

[27] S. H. Lim, C. Wang, and M. Gastpar, "Information-theoretic caching: The multi-user case," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7018–7037, Nov 2017.

[28] R. Timo and M. Wigger, "Joint cache-channel coding over erasure broadcast channels," in *2015 International Symposium on Wireless Communication Systems (ISWCS)*, Aug 2015, pp. 201–205.

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

[30] S. Saeedi Bidokhti, M. Wigger, and R. Timo, "Noisy broadcast networks with receiver caching," *IEEE Transactions on Information Theory*, vol. 64, no. 11, pp. 6996–7016, Nov 2018.

[31] S. S. Bidokhti, M. Wigger, and R. Timo, "An upper bound on the capacity-memory tradeoff of degraded broadcast channels," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sep. 2016, pp. 350–354.

[32] J. Zhang and P. Elia, "Fundamental limits of cache-aided wireless bc: Interplay of coded-caching and csit feedback," *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 3142–3160, May 2017.

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

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

[35] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Edge-facilitated wireless distributed computing," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–7.

[36] ——, "A scalable framework for wireless distributed computing," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2643–2654, Oct 2017.

[37] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 34–40, April 2017.

[38] D. Roca, D. Nemirovsky, M. Nemirovsky, R. Milito, and M. Valero, "Emergent behaviors in the internet of things: The ultimate ultra-large-scale system," *IEEE Micro*, vol. 36, no. 6, pp. 36–44, 2016.

[39] J. Xu, J. Yao, L. Wang, K. Wu, L. Chen, and W. Lou, "Revolution of self-organizing network for 5g mmwave small cell management: From reactive to proactive," *IEEE Wireless Communications*, vol. 25, no. 4, pp. 66–73, 2018.

[40] M. E. Newman, "Complex systems: A survey," *arXiv preprint arXiv:1112.1440*, 2011.

[41] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, no. 2, pp. 202–208, Feb. 1985.

[42] L. W. Dowdy and D. V. Foster, "Comparative models of the file assignment problem," *ACM Comput. Surv.*, vol. 14, no. 2, pp. 287–313, Jun. 1982.

[43] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 6, pp. 1110–1122, Aug 1996.

[44] D. S. P. Dan, Asitand Sitaram, "Dynamic batching policies for an on-demand video server," *Multimedia Systems*, vol. 4, no. 3, pp. 112–121, Jun 1996.

[45] M. R. Korupolu, C. Plaxton, and R. Rajaraman, "Placement algorithms for hierarchical cooperative caching," *Journal of Algorithms*, vol. 38, no. 1, pp. 260 – 302, 2001.

[46] A. Meyerson, K. Munagala, and S. Plotkin, "Web caching using access statistics," in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '01. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001, pp. 354–363.

[47] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411–1429, Aug. 2008.

[48] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–9.

[49] Y. Birk and T. Kol, "Coding on demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2825–2830, June 2006.

[50] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1479–1494, March 2011.

[51] Z. Chen, P. Fan, and K. B. Letaief, "Fundamental limits of caching: improved bounds for users with small buffers," *IET Communications*, vol. 10, no. 17, pp. 2315–2318, 2016.

[52] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 135–139.

[53] S. Sahraei and M. Gastpar, "K users caching two files: An improved achievable rate," in *2016 Annual Conference on Information Science and Systems (CISS)*, March 2016, pp. 620–624.

[54] C. Tian and J. Chen, "Caching and delivery via interference elimination," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1548–1560, March 2018.

[55] M. Mohammadi Amiri and D. Gündüz, "Fundamental limits of coded caching: Improved delivery rate-cache capacity tradeoff," *IEEE Transactions on Communications*, vol. 65, no. 2, pp. 806–815, Feb 2017.

[56] M. M. Amiri, Q. Yang, and D. Gündüz, "Coded caching for a large number of users," in *2016 IEEE Information Theory Workshop (ITW)*, Sep. 2016, pp. 171–175.

[57] A. Sengupta, R. Tandon, and T. C. Clancy, "Improved approximation of storage-rate tradeoff for caching via new outer bounds," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015, pp. 1691–1695.

[58] C. Wang, S. H. Lim, and M. Gastpar, "A new converse bound for coded caching," in *2016 Information Theory and Applications Workshop (ITA)*, Jan 2016, pp. 1–6.

[59] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, Feb 2018.

[60] C. H. H. Suthan, I. Chugh, and P. Krishnan, "An improved secretive coded caching scheme exploiting common demands," in *2017 IEEE Information Theory Workshop (ITW)*, Nov 2017, pp. 66–70.

[61] Kai Wan, D. Tuninetti, and P. Piantanida, "Novel delivery schemes for decentralized coded caching in the finite file size regime," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 1183–1188.

[62] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4388–4413, July 2017.

[63] J. Gómez-Vilardebó, "Fundamental limits of caching: Improved rate-memory tradeoff with coded prefetching," *IEEE Transactions on Communications*, vol. 66, no. 10, pp. 4488–4497, Oct 2018.

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

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

[66] L. Tang and A. Ramamoorthy, "Coded caching schemes with reduced subpacketization from linear block codes," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 3099–3120, April 2018.

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

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

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

[70] G. Vettigli, M. Ji, A. M. Tulino, J. Llorca, and P. Festa, "An efficient coded multicasting scheme preserving the multiplicative caching gain," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2015, pp. 251–256.

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

[72] S. Jin, Y. Cui, H. Liu, and G. Caire, "A new order-optimal decentralized coded caching scheme with good performance in the finite file size regime," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5297–5310, Aug 2019.

[73] S. M. Asghari, Y. Ouyang, A. Nayyar, and A. S. Avestimehr, "Optimal coded multicast in cache networks with arbitrary content placement," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[74] G. Vettigli, M. Ji, K. Shanmugam, J. Llorca, A. M. Tulino, and G. Caire, "Efficient algorithms for coded multicasting in heterogeneous caching networks," *Entropy*, vol. 21, no. 3, 2019.

[75] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, 4th ed. McGraw Hill, 2002.

[76] D. E. Knuth, "Big omicron and big omega and big theta," *SIGACT News*, vol. 8, no. 2, p. 18–24, Apr. 1976.

[77] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[78] R. P. Stanley, *Enumerative Combinatorics: Volume 1*, 2nd ed. New York, NY, USA: Cambridge University Press, 2011.

[79] C. Walck, *Hand-book on statistical distributions for experimentalists*, 1996. [Online]. Available: http://staff.fysik.su.se/~walck/