Coded Caching under Arbitrary Popularity Distributions

Jinbei Zhang[†], Xiaojun Lin[‡], Xinbing Wang[†]

[†]Dept. of Electronic Engineering, Shanghai Jiao Tong University, China

[‡]School of Electrical and Computer Engineering, Purdue University, USA

Email: abelchina@sjtu.edu.cn, linx@purdue.edu, xwang8@sjtu.edu.cn

Abstract

Caching plays an important role in reducing the backbone traffic when serving high-volume multi-media content. Recently, a new class of coded caching schemes have received significant interest because they can exploit coded multi-cast opportunities to further reduce backbone traffic. Without considering file popularity, prior works have characterized the fundamental performance limits of coded caching through a deterministic worst-case analysis. However, when heterogeneous file popularity is taken into account, there remain open questions regarding the fundamental limits of coded caching performance. In this work, for an arbitrary popularity distribution, we first derive a new information-theoretical lower bound on the expected transmission rate of any coded caching schemes. We then show that a simple coded-caching scheme attains an expected transmission rate that is at most a constant factor away from the lower bound (except a small additive term). Unlike other existing studies, the constant factor that we derived is independent of the popularity distribution.

I. Introduction

As the amount of Internet traffic continues to grow, video is expected to dominate 69% of the overall traffic [2], which will greatly stress the underlying communication infrastructure.

An earlier version of this paper has appeared in Information Theory and Applications Workshop, UCSD, Feb. 2015 [1].

Historically, caching has played a significant role in reducing the bandwidth requirement for serving video traffic. By placing contents closer to, or even at the end-users, the bandwidth requirement at the upstream links can be greatly reduced. Most of such studies of caching have focused on the case where uncoded video packets were stored and transmitted (see, e.g., [3–6] and references therein).

Recently, a new class of caching schemes, called coded caching [7–16], have gained significant interest because it can significantly reduce the upstream bandwidth requirement in systems with broadcast/multicast capabilities. Consider K users request contents from one server through a shared communication link with broadcast capability. Each user may request any one of the N files ($N > K$), but each user only has a storage with size $M < N$. In the worst case, each user may request a distinct file. With conventional (uncoded) caching scheme, it is easy to see that the worst-case transmission rate on the upstream link must be $K(1 - \frac{M}{N})$, because each user can only cache $\frac{M}{N}$ fraction of all the contents. [7] refers to this factor $(1 - \frac{M}{N})$ as the *local* caching gain. Unless M is large (compared to N), this local caching gain will not differ significantly from 1 (i.e., the baseline with no-caching). Note that the broadcast capability of the system is not exploited here because each user requests a different file. In contrast, with the coded caching scheme in [7], the worst-case transmission rate at the upstream link is reduced to $K(1 - \frac{M}{N})\frac{1}{1+KM/N}$. The additional factor $\frac{1}{1+KM/N}$, which is referred to as the *global caching gain* in [7], suggests a significant improvement over the uncoded case when the *global* storage capability KM of all users is comparable to, or larger than, N. The key idea of [7] is to transmit *coded* packets so that multiple users can benefit from the same broadcast packet. Thus, the broadcast capability in the system can be exploited even if different users request different files. [7] further shows in an information theoretic sense that the worst-case transmission rate of the coded caching scheme in [7] is at most a constant factor (specifically, 12 times) away from the minimum possible. In this sense, the performance of the coded caching scheme of [7] is close to the fundamental limit for the system studied. The works in [8, 14–16] extend this idea to decentralized caching, hierarchical networks, multiple group-cast, and online caching,

respectively.

The studies cited above all focus on the *deterministic* worst-case, i.e., not only does each user request distinct files, the performance of the system is studied against the worst-case request pattern. Arguably, if the popularity of the files are identical, the probability of each request pattern will vary less significantly. Then, the worst-case performance may not differ significantly from the average-case performance [9]. In reality, however, the file popularity *can* differ significantly, and thus some request patterns will occur much more frequently than other request patterns. As a result, the average-case performance can differ significantly from the worst-case bound (see also the discussions at the end of Section II).

While the average-case performance of coded caching under heterogeneous file popularity was studied in [9–11], the optimality bounds obtained are substantially weaker than the results in [7] because the gap between the achievable bound and the lower bound depends on various system parameters. Specifically, in [9], contents are divided into groups with similar popularity. Each group is assigned a separate portion of the cache and uses the coded caching scheme of [8]. The gap between the corresponding transmission rate and the lower bound is found to increase with the total number of groups. Similarly, in [10] the authors study the case when the file popularity has L different levels. The theoretical gap between the achievable transmission rate and the lower bound increases as L^3. The work in [11] is most related to ours, where the authors study the special case when file popularity follows a Zipf distribution. Although the authors also show constant-order gaps between the achievable bound and the lower bound, the gaps estimated by the theoretical results of [11] depend on the parameters of the Zipf distribution, and may also become large for certain ranges of the parameter values [12, 13]. Further, the constant factors are only shown in the asymptotic limit when the number of files and/or the number of users are large. Therefore, it remains an important open question what is the fundamental limit of the performance of coded caching for the more practical scenario of heterogeneous file popularity, and whether one can find a coded-caching scheme whose performance gap from the lower bound is *independent* of the popularity distribution even in the non-asymptotic settings.

In this paper, we make the following contributions to answer the above open question. First, we show that a simple coded-caching scheme (similar to the one in [11]) can attain an average transmission rate that is at most 87 times from the optimal (except a small additive term). Although this factor appears to be large, it is the first result in the literature with a constant-factor gap that is independent of the popularity distributions and the system size. In contrast, in earlier studies the performance gap could be arbitrarily large depending on either the number of groups [9], the number of levels [10], or the parameter of the Zipf distribution [11–13]. Second, a key step towards this result is to use a new construction to establish a much sharper lower bound on the achievable transmission rate of *any* schemes (see Section IV for details). Specifically, we establish this lower bound by a series of *reduction* and *merging* steps that convert the original system with heterogeneous popularity to other systems with uniform popularity. Using these techniques, we are able to quantify the impact of both "popular" files and "non-popular" files, which we believe is the main reason that we can obtain sharp constant-factor characterizations even in the non-asymptotic settings. (See further discussions at the end of Section IV.D.) These techniques may be of independent interest for future studies of coded caching performance. Third, our analysis reveals the important role of the file with a threshold popularity. Specifically, suppose that the number of users is K and the size of each user's storage is M. The achievable scheme caches evenly all files whose popularity is greater than or equal to $\frac{1}{KM}$, and does not cache the rest of the files. The decentralized coded caching scheme in [8] is then used to serve the files whose popularity is greater than or equal to $\frac{1}{KM}$. It is quite remarkable that, regardless of the popularity distribution, this simple coded caching scheme will achieve a transmission rate that is a constant factor away from the fundamental lower bound (except a small additive term). Finally, as an immediate corollary, our result implies that the version of Random LFU scheme in [11] that numerically optimizes the threshold $\tilde{m}$ (which is comparable to N_1 in our paper) also attains an average transmission rate that is away from the optimal by at most a constant factor, independently of the popularity distribution (even though the theoretical results in [11] focus only on Zipf distributions and asymptotic settings).

The remainder of this paper is as follows. We first present the network model in Section II. Main results are summarized in Section III. Followed is the analysis on the information theoretical lower bound in Section IV. The achievable scheme is analyzed in Section V. Simulation results are presented in Section VI. Then, we conclude.

II. Network Model

In the following, we present the network model for a video delivery system with both local caches and broadcast capabilities (see Figure 1).

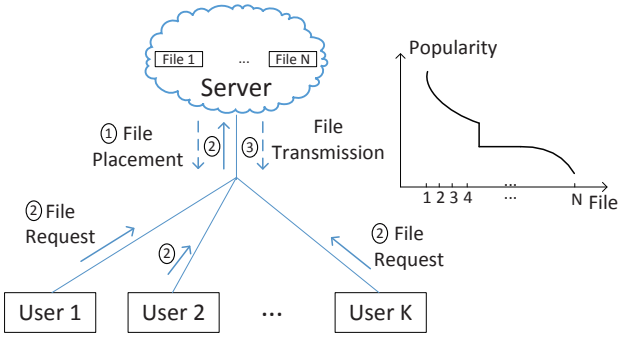

Fig. 1: An illustration of the network model.

We assume that there are N distinct files from the set $\mathcal{F} = \{F_1, F_2, ..., F_N\}$. The popularity of the file F_i is p_i, where $\sum_{i=1}^{N} p_i = 1$. Without loss of generality, we assume that the file size is of unit length and the file popularity is decreasing in the index, i.e., $p_i \geq p_j$ if $i \leq j$.

There is one server who has all N files and who serves these files to K users interested in these files. Each user has a local cache with size M (again measured with respect to the unit-length of the files). The K users are connected to the server through a network with broadcast capability, i.e., each transmission from the server can be received by all users.

Before users request any files, some of the contents are placed in the users' caches. This is called cache placement and in practice is usually carried out during off-peak hours of the network. Then, at each time, a user k will request file F_i with probability p_i, independently of all other users and files. If the user's local cache already has (some of) the content, the request can be served locally. Otherwise, the server must transmit (via broadcast) contents not available

from the local cache. The goal is that every user should be able to reconstruct the file that it requests with the information received from the server and the cached content in its local cache.

A. Definition of the Expected Transmission Rate

In this subsection, we will define the expected rate needed from the server in serving the requests. Note that we do not consider the transmission rate for cache placement.

Let $W_i = \{f_{i1}, f_{i2}, ..., f_{iK}\}$ denote a request pattern, where $f_{ij} \in \mathcal{F}$ is the requested file for the j-th user, $1 \leq j \leq K$. Note that there are N^K such patterns. Let $\mathbb{W}$ be the set of all possible request patterns from K users, i.e., $\mathbb{W} = \{W_1, W_2, ..., W_{N^K}\}$. Since each user can request one file from N files independently, the probability for event W_i is given by

$$P(W_i) = \prod_{j=1}^{K} P(f_{ij})$$

where $P(f_{ij})$ is the probability for a user to request file f_{ij}. Note that in our original system, the probability for a user to request file F_j is p_j. However, later in the analysis we will compare to another system with a different popularity distribution $\mathcal{P}$. Hence, we use the notation $\mathbb{W}(K, \mathcal{F}, \mathcal{P})$ to denote the set $\mathbb{W}$ of possible request patterns associated with the corresponding popularity distribution $\mathcal{P}$.

Obviously, given a set of files $\mathcal{F}$ and the files' corresponding popularity distribution $\mathcal{P}$, there exists numerous caching and transmission schemes to meet users' request. For a caching and transmission scheme $\mathfrak{F}$, let $r_{\mathfrak{F}}(K, W_i)$ denote the amount of broadcast transmission from the server that is needed to satisfy a request W_i. The expected rate under scheme $\mathfrak{F}$ is therefore defined as

$$R_{\mathfrak{F}}(K, \mathcal{F}, \mathcal{P}) = \sum_{i=1}^{N^K} r_{\mathfrak{F}}(K, W_i) P(W_i). \tag{1}$$

We wish to find the schedule $\mathfrak{F}$ that minimizes $R_{\mathfrak{F}}(K, \mathcal{F}, \mathcal{P})$. Define the optimal rate as

$$R(K, \mathcal{F}, \mathcal{P}) = \min_{\mathfrak{F}} R_{\mathfrak{F}}(K, \mathcal{F}, \mathcal{P}). \tag{2}$$

Unfortunately, finding the exact optimal schedule that achieves this optimal rate is very difficult

[7–16]. Like [7–16], our goal is to find a simple scheme $\mathfrak{F}$ whose achievable rate is as close to the optimal rate $R(K, \mathcal{F}, \mathcal{P})$ as possible.

Remark: In [7], instead of studying the expected rate (1), the authors focus on the worst-case rate, i.e.,

$$\max_{W_i} r_{\mathfrak{F}}(K, W_i). \tag{3}$$

Let $\mathfrak{F}^*$ be the optimal scheme that attains the minimum value of (3), and let $\mathfrak{F}$ be the scheme proposed in [7]. Then [7] shows that

$$\frac{\max_{W_i} r_{\mathfrak{F}}(K, W_i)}{\max_{W_i'} r_{\mathfrak{F}^*}(K, W_i')} \leq 12. \tag{4}$$

However, in this paper since we are interested in the expected rate given in (2), we would be interested in the gap

$$\frac{\sum_{i=1}^{N^K} r_{\mathfrak{F}}(K, W_i) P(W_i)}{\sum_{i=1}^{N^K} r_{\mathfrak{F}^*}(K, W_i') P(W_i')}. \tag{5}$$

Note that the bound in (4) does not imply that the expression in (5) is bounded by the same constant, especially when the probability $P(W_i)$ varies significantly. In general, even if the bound in (4) holds, the expression in (5) can still be arbitrarily large. Thus, quantifying the performance gap in terms of the expected rate represents a new research problem.

III. Main Results

In this section, we provide an overview of our main results. Given an arbitrary popularity distribution, our first result establishes a fundamental lower bound on the expected transmission rate for any coded caching scheme. Let $[x]_+$ denote $\max\{0, x\}$.

Theorem 1: With K users requesting files independently in $\mathcal{F}$ according to the corresponding popularity distribution $\mathcal{P}$, the lower bound on the expected transmission rate is given by

$$R(K, \mathcal{F}, \mathcal{P}) \geq \max\left\{ \frac{1}{29}\left[\frac{N_1}{M} - 1\right]_+, \frac{1}{58}\left[\sum_{i>N_1} Kp_i - 2\right]_+ \right\}, \tag{6}$$

where $M \geq 2$ and N_1 is an integer that satisfies $KMp_{N_1} \geq 1$ and $KMp_{N_1+1} < 1$.

To the best of our knowledge, the lower bound in Theorem 1 has not been reported in the literature, and this sharper bound is the main reason behind the improved performance characterization reported in this paper. Thus, this lower bound is one of the main contributions of the paper. Further, we comment on the index N_1, which plays an important role in most of the results in this paper. Recall that the popularity p_i is non-increasing in the file index i. Roughly speaking, N_1 is the index for the file whose popularity is around $\frac{1}{KM}$. We may view all files $i \leq N_1$ as the "more popular" files, and all files $i > N_1$ as the "unpopular" files. As readers will see in the proofs of Theorem 1 in Section IV, the first term $\frac{1}{29}[\frac{N_1}{M} - 1]_+$ is a lower bound on the expected transmission rate for serving the more popular files, while the second term is a lower bound on the expected transmission rate for serving the unpopular files. Thus, they combine to produce the lower bound in Theorem 1. This result is shown by carefully constructing a series of reduced systems whose performance is easier to characterize. Details of the proof will be presented in Section IV.

We next present an achievable scheme that can attain a corresponding upper bound. Recall that each file is of unit length. In order to allow a portion of each file to be cached, we refer to a minimally divisible portion of a file as a "bit", and assume that each file has F "bits". We are most interested in the case of large files, i.e., when the bits are very small compared to the file size, and hence $F \to +\infty$. Our proposed achievable scheme uses the decentralized coded caching scheme of [8] to serve the "popular" files, and uses uncoded transmissions to serve the "unpopular" files. Specifically, each user randomly caches an equal number of $\min\{F, \frac{MF}{N_1}\}$ bits from every file $F_1, ..., F_{N_1}$. The remaining unpopular files are *not* cached unless there are space left after all the more popular files are cached (i.e., when $M > N_1$). After the users request the files according to the popularity distribution $p_1, ..., p_N$, the decentralized *coded* transmission scheme of [8] is used to serve those users requesting popular files, and an *uncoded* transmission scheme is used to serve those users requesting unpopular files. The details will be presented in Section V. We note that this scheme is similar to the Random LFU scheme studied in [11], although the rules for choosing the threshold file N_1 (which corresponds to $\tilde{m}$ in [11]) are

different. The following result summarizes an upper bound on the expected transmission rate of this simple scheme.

Theorem 2: With K users independently requesting files in $\mathcal{F}$ according to the popularity distribution $\mathcal{P}$, as $F \to +\infty$, the optimal achievable rate can be upper bounded by

$$R(K, \mathcal{F}, \mathcal{P}) \leq \left[\frac{N_1}{M} - 1\right]_+ + \min\left(\sum_{i>N_1} Kp_i, \frac{N - N_1}{[M - N_1]_+} - 1\right). \tag{7}$$

In (7), the first term $[\frac{N_1}{M} - 1]_+$ is an upper bound on the expected transmission rate to serve the more popular files (i.e., with index $i \leq N_1$), and the second minimization term is an upper bound on the expected transmission rate to serve unpopular files. Assuming that $M < N_1$, note that increasing N_1 by 1 will increase the first term by $1/M$, and will reduce the second term by roughly Kp_{N_1}. Thus, by setting $p_{N_1} \approx \frac{1}{KM}$, this index N_1 is chosen such that the net effect to the upper bound (7) is approximately zero, and thus the sum in (7) is approximately minimized.

Remark: We note that a similar upper bound is also reported in [11], although without the last term in (7), which captures the case with abundant caches (i.e., $M > N_1$).

From Theorem 1 and Theorem 2, it is easy to show that the gap between the lower bound R_{lb} and the upper bound R_{up} is bounded by

$$R_{up} \leq 87R_{lb} + 2. \tag{8}$$

Thus, except a small additive term of 2, the bounds differ by at most a factor of 87. As we discuss in the introduction, although this factor may appear to be large, it is the first result in the literature with a constant-factor gap that is independent of the popularity distributions. In contrast, the gap (between upper- and lower-bounds) estimated by the existing results can be arbitrarily large depending on either the number of groups [9], the number of levels [10], or the parameter of the Zipf distribution [11]. It is remarkable that such a simple coded caching scheme, with a very simple choice of N_1, can achieve such a strong performance guarantee, independently of the popularity distribution.

We briefly discuss the relationship between the above scheme and the Random LFU (RLFU)

scheme in [11] because they are similar. RLFU also evenly caches files whose popularity is above a threshold. In RLFU, the file with the threshold popularity is denoted as $\tilde{m}$, which plays a similar role as N_1 in this paper. In this sense, the above simple scheme can also be viewed as a member within the class of RLFU. However, the details in choosing the threshold popularity differ. In the theoretical analysis in [11], $\tilde{m}$ is chosen as a function of the exponent α of the Zipf distribution (assuming that the popularity of the i-th most-popular file is proportional to $1/i^{\alpha}$). Based on this choice of $\tilde{m}$, [11] bounds the gap between the achievable rate and the lower-bound as a function of α. This theoretical bound on the performance gap roughly scales as $1/(\alpha-1)$, which becomes unbounded as α approaches 1. On the other hand, [11] also proposes a practical version of RLFU that numerically optimizes an upper bound over all possible values of $\tilde{m}$. Since the performance of the numerically-optimized RLFU scheme is always no worse than that with any fixed $\tilde{m}$, the theoretical performance guarantees in [11] for Zipf distributions also apply to this numerically-optimized RLFU scheme. In contrast, in this paper the threshold file N_1 is chosen to be the one with popularity close to $1/KM$. Not only does this rule apply to all popularity distributions, it also leads to an achievable transmission rate that is away from the lower bound by at most a constant factor *independent of the popularity distribution*. Thus, our results reveal new insights on the choice of this threshold. Further, we note that the performance of the numerically-optimized RLFU scheme in [11] must also be no worse than that with our choice of N_1. Thus, as an immediate corollary of our result, it implies that the numerically-optimized RLFU scheme in [11] also attains a constant-factor performance gap for arbitrary popularity distributions. We also note that, for certain ranges of the exponent α of the Zipf distributions, the performance characterization in [11] may be tighter than the 87 factor reported in (8). Thus, the results in [11] and in this paper combined provide a more complete characterization of the performance guarantees for the numerically-optimized RLFU scheme across both Zipf and non-Zipf distributions.

A. Main Intuition

Before we present the proofs for these main results, we would like to illustrate the main

intuition behind. First, consider only the "popular files" 1 to N_1, i.e., assuming that the unpopular files N_1+1 to N are removed. Let us refer to this system as "System 1". In our proof, we will consider an alternate system where the popularity of all popular files is reduced to p_{N_1}. We will refer to this alternate system as "System 2" (see Section IV-A). Intuitively, the average transmission rate in System 2 is no larger than that in System 1. Further, since all files are with the same popularity in System 2, the average-case and the worst-case performance will not differ too much [9]. Thus, one can then use System 2 to derive a lower bound on the average transmission rate, and compare it with an upper bound attained by an achievable scheme.

However, the potential problem of this argument is that, when we reduce the popularity of all popular files to p_{N_1}, some popularity values could be reduced by several orders of magnitude. It is then unclear why the lower bound derived from System 2 is still a reasonable lower bound for System 1. The intuition behind this insensitivity can be explained as follows. Suppose that there are K' users in System 1 that request any of the popular files. Then, according to the result in [7], the worst-case transmission rate to serve these K' users is no larger than

$$K'(1-\frac{M}{N_1})\frac{1}{1+\frac{K'M}{N_1}}. \tag{9}$$

Now, suppose that the individual cache size M is much smaller than N_1, and the global cache size $K'M$ is much larger than N_1 (note that this is precisely the regime where coded caching will be most helpful [7]). Then, we have $1-\frac{M}{N_1}\approx 1$ and $1+\frac{K'M}{N_1}\approx\frac{K'M}{N_1}$. Thus, the expression in (9) is approximately equal to N_1/M. The significance of this observation is that this approximated expression is independent of K'. In other words, in a suitable regime of interest, the exact popularity of the "popular files" does not seem to matter! It is then plausible to argue that, even when we reduce the popularity values to p_{N_1} in System 2, there is no substantial change in the lower-bound performance. Of course, this argument needs to be carefully made. Further, we have to account for not only popular files, but also unpopular files. The proofs in the next section will make this intuition precise.

IV. Lower Bound on the Expected Rate

In this section, we present the proof of Theorem 1, i.e., the lower bound.

The proof consists of two parts. Subsections A-C focus on popular files 1 to N_1, and prove the part that $R(K,\mathcal{F},\mathcal{P}) \geq \frac{1}{29}(\frac{N_1}{M}-1)$. This proof is composed of 5 steps. From the first step to the fourth one, we map the original system into a series of reduced systems, whose information-theoretical rate is strictly smaller than previous ones. Then, we calculate the rate needed for the system constructed in the fourth step. Finally, Subsection D focuses on the unpopular files and proves the part that $R(K,\mathcal{F},\mathcal{P}) \geq \frac{1}{58}(\sum_{i>N_1} Kp_i - 2)$. As we elaborate further below, while some of the techniques for quantifying the impact of popular files are similar to [9][13], our treatment of unpopular files is new and is the key reason for the sharper constant-factor characterization in our paper.

A. Reduction Steps 1 & 2

Recall that the set of files is given by $\mathcal{F} = \{F_1, F_2, ..., F_N\}$ and their popularity distribution is given by $\mathcal{P} = \{p_1, p_2, ..., p_N\}$. Next, we will compare to a series of reduced systems with different sets of files and popularity distributions. Again, let N_1 be the integer defined in Theorem 1.

In the first constructed system, the set of files is given by $\mathcal{F}_1 = \{F_0, F_1, F_2, ..., F_{N_1}\}$, where F_0 denotes the empty file, which is introduced for ease of presentation. Its corresponding popularity distribution is $\mathcal{P}_1 = \{p_0, p_1, p_2, ..., p_{N_1}\}$ and $p_0 = 1 - \sum_{i=1}^{N_1} p_i$. In other words, we replace all unpopular files $F_{N_1+1}, ..., F_N$ in the original system by the empty file F_0, and reassign their popularity all to F_0. Intuitively, the new system should require a lower transmission rate than the original system, which is stated in the following lemma.

Lemma 1: Let $R(K,\mathcal{F}_1,\mathcal{P}_1)$ be the minimum expected rate required to meet the requests by the K users, each of which randomly requests a file in $\mathcal{F}_1$ according to the popularity distribution $\mathcal{P}_1$. We have

$$R(K,\mathcal{F},\mathcal{P}) \geq R(K,\mathcal{F}_1,\mathcal{P}_1). \tag{10}$$

We next create another new system by a further adjustment on the tuple $(K, \mathcal{F}_1, \mathcal{P}_1)$. Note that $N_1 \leq KM$. Otherwise, we will have $\sum_{i=1}^{N} p_i > KM \cdot p_{N_1} \geq 1$, which is a contradiction with $\sum_{i=1}^{N} p_i = 1$. Define a new popularity distribution $\mathcal{P}_2 = \{1 - N_1 p_{N_1}, p_{N_1}, p_{N_1}, ..., p_{N_1}\}$. In other words, compared to $(K, \mathcal{F}_1, \mathcal{P}_1)$, in this new system, each non-empty file is requested with a smaller probability p_{N_1}. Intuitively, its expected transmission rate should be even lower, which is stated below.

Lemma 2: Let $R(K, \mathcal{F}_1, \mathcal{P}_2)$ be the minimum expected rate required to meet the requests by K users, each of which randomly requests a file in $\mathcal{F}_1$ according to the popularity distribution $\mathcal{P}_2$. We have

$$R(K, \mathcal{F}_1, \mathcal{P}_1) \geq R(K, \mathcal{F}_1, \mathcal{P}_2). \tag{11}$$

The proofs of Lemma 1 $\&$ 2 use similar coupling idea as in Lemma 5, and are omitted here.

With Lemma 1 and Lemma 2, we have proved that $R(K, \mathcal{F}, \mathcal{P}) \geq R(K, \mathcal{F}_1, \mathcal{P}_2)$. In the following analysis for the first part of Theorem 1, we will focus on $R(K, \mathcal{F}_1, \mathcal{P}_2)$. Note that the system $(K, \mathcal{F}_1, \mathcal{P}_2)$ is precisely the "System 2" that we discussed in Section III-A. Next, we will derive a lower bound on the average transmission rate of System 2, which also provides a lower bound on $R(K, \mathcal{F}, \mathcal{P})$. We will derive this lower bound on the *average* transmission rate of System 2 by relating it to a lower bound on the *worst-case* transmission rate. Note that since all files have equal popularity in System 2, the fact that its *average* transmission rate is at most a constant factor away from its *worst-case* transmission rate is in fact known from the results in [9] and [13]. For example, we can obtain a lower bound on the average transmission rate of System 2 from Theorem 2 in [9] by choosing $c = 1$, $N_l = N_1$ there and by choosing K_l in [9] as the number of users requesting popular files. However, the lower bound derived in this way involves an expectation over K_l. Since later we will use System 2 again to deal with unpopular files, we wish to obtain a lower bound that is a function of the total number of users K. The following derivation accounts for such technical details, and at the same time yields a tighter characterization (which eventually translates to the factor $1/29$ and $1/58$ in (6)). We note that some of the reduction techniques below and in Sections IV-B to IV-C are also similar to [9][13],

although here we exploit the fact that $p_{N_1} \approx \frac{1}{KM}$ to obtain the tighter characterization.

To proceed, note that in the system $(K, \mathcal{F}_1, \mathcal{P}_2)$, it is possible that some file is requested by multiple users. In Section IV-B, we will reduce it to the third system where every non-empty requested file is requested exactly once. Towards that end, we first characterize the number of distinct files requested in system $(K, \mathcal{F}_1, \mathcal{P}_2)$.

For a given system setting $(K, \mathcal{F}_1, \mathcal{P}_2)$, let $I_i = 1$ if user i requests a non-empty file, and let $I_i = 0$ if user i requests the empty file. Denote $K_r = \sum_{i=1}^{K} I_i$. Then, K_r is the number of users who request non-empty files. All I_i are *i.i.d.* distributed with mean $N_1 p_{N_1}$. The probability distribution for K_r is given by

$$P(K_r = K_1) = C_K^{K_1} \left(N_1 p_{N_1}\right)^{K_1} \left(1 - N_1 p_{N_1}\right)^{K - K_1}.$$

Lemma 3: Define $K_1 \triangleq \lfloor \frac{N_1}{M} \rfloor$. Then we have $K_1 \leq \lfloor K N_1 p_{N_1} \rfloor$, and

$$P(K_r \geq K_1) \geq \frac{1}{2}. \tag{12}$$

This follows from the result in [17], which shows that any median must lie in the interval $[\lfloor np \rfloor, \lceil np \rceil]$, for a binomial distribution $B(n, p)$.

In other words, with probability no less than 0.5, no less than K_1 users request non-empty files. Still, some of these K_1 users may request a common file. Next, we are interested in the number of distinct files that are requested. Denote this number as K_d.

Lemma 4: Given that there are K_r users requesting non-empty files, the probability that the number of distinct files requested is no smaller than $\min\{\lfloor \frac{1}{2} K_r \rfloor, \lfloor \frac{1}{2} K_1 \rfloor\}$ is greater than or equal to 0.56.

Proof: Clearly, we only need to consider $K_r \leq K_1$ (because a larger value of K_r only increases the number of distinct files). When $K_r = 1, 2,$ or 3, we have that $\lfloor \frac{1}{2} K_r \rfloor$ equals 0 or 1. In this case, it is easy to see that this lemma holds, since there must be at least one distinct file requested.

For $K_r \geq 4$, consider only those K_r users requesting non-empty files. Each user requests one

file from the N_1 non-empty files uniformly randomly and independently. There are $N_1^{K_r}$ possible request patterns for the K_r users, each of which is equally likely. For some of these request patterns, the number of distinct files are smaller than $K_2 \triangleq \lfloor \frac{1}{2} K_r \rfloor$. The number of such request patterns must be smaller than $C_{N_1}^{K_2} \cdot K_2^{K_r}$. To see this, note that the first term is the number of ways to choose K_2 files from the N_1 non-empty files. The second term is the number of ways that each user can choose one of the K_2 files. We thus have

$$\begin{aligned} P(K_d \leq K_2 | K_r) < & \frac{C_{N_1}^{K_2} K_2^{K_r}}{N_1^{K_r}} \\ \leq & \frac{e\sqrt{N_1}(\frac{N_1}{e})^{N_1}}{\sqrt{2\pi(N_1-K_2)}(\frac{N_1-K_2}{e})^{N_1-K_2}} \\ & \cdot \frac{1}{\sqrt{2\pi K_2}(\frac{K_2}{e})^{K_2}} \left(\frac{K_2}{N_1}\right)^{K_r} \\ \leq & \frac{e}{2\pi} \left(\frac{N_1}{N_1-K_2}\right)^{N_1-K_2} \cdot (\frac{N_1}{K_2})^{K_2-K_r}. \end{aligned}$$

Here, we have used Stirling's formula in the third step, i.e.,

$$\sqrt{2\pi n} \left(\frac{n}{e}\right)^n \leq n! \leq e\sqrt{n} \left(\frac{n}{e}\right)^n.$$

In the fourth step, we have used $\sqrt{\frac{N_1}{K_2(N_1-K_2)}} \leq 1$, due to $K_2 \geq 2$ and $K_2 \leq \frac{1}{2}N_1$. It is easy to prove that $(1+x)^{\frac{1}{x}} \leq e$ for any $x > 0$. Therefore,

$$\left(\frac{N_1}{N_1-K_2}\right)^{N_1-K_2} = \left(1+\frac{K_2}{N_1-K_2}\right)^{\frac{N_1-K_2}{K_2}\cdot K_2} \leq e^{K_2}.$$

Due to $K_2 \leq \frac{1}{2}K_r \leq \frac{1}{2}K_1 \leq \frac{1}{4}N_1$ (since $M \geq 2$), we have

$$P(K_d \leq K_2 | K_r) < \frac{e}{2\pi} e^{K_2} \cdot e^{K_2-K_r} \leq \frac{e}{2\pi}.$$

Finally, $P(K_d > K_2 | K_r) = 1 - P(K_d \leq \frac{K_r}{2} | K_r) \geq 0.56$. ■

B. Reduction Step 3

Combing Lemma 3 and Lemma 4, we can show that, with probability no less than 0.28, the number of distinct files requested is no smaller than $\lfloor \frac{1}{2} K_1 \rfloor$. We now perform the third reduction.

For $\mathbb{W}(K, \mathcal{F}_1, \mathcal{P}_2)$, let $\pi(K_1)$ be the probability that either the number of users requesting non-empty files, K_r, is less than K_1, or the number of distinct non-empty files requested, K_d, is less than $K_3 \triangleq \lfloor \frac{K_1}{2} \rfloor$.

Then, in the third system, with probability $\pi(K_1)$ the users will all request the empty file. With probability $1 - \pi(K_1)$, exactly K_3 users will request exactly K_3 distinct non-empty files from $F_1, ..., F_{N_1}$, and all other users will request the empty file. Note that there are exactly $C_K^{K_3} A_{N_1}^{K_3}$ request patterns where exactly K_3 users request K_3 distinct non-empty files. We let each such request pattern occur with equal probability $\frac{1-\pi(K_1)}{C_K^{K_3} A_{N_1}^{K_3}}$.

Let this third system be denoted by $\mathbb{W}_3(K_3, K_1)$, and let $R(K, \mathbb{W}_3)$ be the corresponding minimum expected transmission rate. Then. we have the following lemma.

Lemma 5:

$$R(K, \mathcal{F}_1, \mathcal{P}_2) \geq R(K, \mathbb{W}_3). \tag{13}$$

Proof: The proof uses coupling [19]. For every $W_i \in \mathbb{W}(K, \mathcal{F}_1, \mathcal{P}_2)$, map it to a random $W_i^{'} \in \mathbb{W}_3(K_3, K_1)$ as follows. If the number of users requesting non-empty files in W_i is less than K_1, or the number of distinct non-empty files requested is less than $K_3 \triangleq \lfloor \frac{K_1}{2} \rfloor$, then in $W_i^{'}$ all users request the empty file. Otherwise, we perform the mapping described below.

For every remaining W_i with $K_d \geq K_3$, we conduct the following splitting procedure.

- For each non-empty file that is requested by some users, randomly choose one user requesting it. Note that there are K_d such chosen users.
- Among the chosen users, randomly choose K_3 of them. These K_3 users now request distinct non-empty files, and we let all other users request the empty file.

It is easy to see that, given any cache placement and transmission scheme $\mathfrak{F}$, we have

$$r_{\mathfrak{F}}(K, W_i) \geq r_{\mathfrak{F}}(K, W_i^{'}), \tag{14}$$

because $W_i^{'}$ requests a subset of the files in W_i. It remains to show that, if W_i is chosen according to the distribution of $\mathbb{W}(K, \mathcal{F}_1, \mathcal{P}_2)$, then the resulting $W_i^{'}$ has the same distribution

as $\mathbb{W}_3(K_3, K_1)$. To see this, note that the probability with which W_i requests no empty files is exactly $1-\pi(K_1)$. Further, due to symmetry on the files and the users in $\mathbb{W}(K, \mathcal{F}_1, \mathcal{P}_2)$, along with the symmetry of our mapping, each pattern W_i' that requests non-empty files must occur with equal probability. We can then conclude that each W_i' occurs with the same probability as in $\mathbb{W}_3(K_3, K_1)$.

Thus, with the coupling method [19], we have

$$R_{\mathfrak{F}}(K, \mathcal{F}_1, \mathcal{P}_2) \geq R_{\mathfrak{F}}(K, \mathbb{W}_3). \tag{15}$$

and the result then follows. ■

C. Reduction Step 4 & the Lower Bound

We now consider the 4th system $\mathbb{W}_4$. In this system, there are always $K_3 \triangleq \lfloor \frac{K_1}{2} \rfloor$ users requesting K_3 distinct non-empty files and all the other users request the empty file. Further, each such request pattern occurs with equal probability $\frac{1}{C_K^{K_3} A_{N_1}^{K_3}}$. Let $R(K, \mathbb{W}_4)$ be the minimum expected transmission rate for the above system $\mathbb{W}_4$. The following lemma is easy to show.

Lemma 6: $R(K, \mathbb{W}_3) = (1-\pi(K_1))R(K, \mathbb{W}_4)$.

Next we focus on the system $\mathbb{W}_4$.

Let H_i, $i = 1, 2, ..., C_K^{K_3}$ be the $C_K^{K_3}$ choices of picking K_3 users out of the K users. In system $\mathbb{W}_4$, if in a request W_j, the users requesting distinct non-empty files are exactly in H_i, we denote it by $W_j \in \overline{H_i}$. Note that there are $A_{N_1}^{K_3} = \frac{N_1!}{(K-K_3)!}$ such patterns in each $\overline{H_i}$. We have the following result.

Lemma 7: Consider systems $\mathbb{W}_4$ where there are always exactly K_3 users requesting distinct files in $\mathcal{F}_1$ and the other $K-K_3$ users request the empty file. For any H_i, $i = 1, 2, ..., C_K^{K_3}$, the following holds,

$$\sum_{W_j \in \overline{H_i}} r_{\mathfrak{F}}(K, W_j) \geq A_{N_1}^{K_3} \cdot \frac{\left\lfloor \frac{N_1}{K_3} \right\rfloor K_3 - K_3 M}{\left\lfloor \frac{N_1}{K_3} \right\rfloor}. \tag{16}$$

Note that Lemma 7 immediately implies that

$$R(K, \mathbb{W}_4) \geq \frac{\left\lfloor \frac{N_1}{K_3} \right\rfloor K_3 - K_3 M}{\left\lfloor \frac{N_1}{K_3} \right\rfloor}. \tag{17}$$

Proof: Without loss of generality, suppose that $H_i = \{1, 1, ..., 1, 0, 0, ..., 0\}$. In other words, user 1, 2,..., K_3 are requesting distinct non-empty files. Each user has a cache, labeled $M_1, M_2, ..., M_{K_3}$, each of which has a common storage size M.

There are $N_1!$ permutations for the N_1 files. For each permutation, we split it into $\lfloor \frac{N_1}{K_3} \rfloor$ subgroups, each with K_3 files. Denote $r(i, j)$ as the rate needed to meet the users' requests if their request pattern is the same as the j-th subgroup in the i-th permutation, i.e., when the k-th user requests the k-th file in the subgroup, $k = 1, 2, ..., K_3$.

For each permutation i, consider all the sub-groups (i.e., request patterns) as a whole. Recall that the cache content is fixed when these request patterns vary. Consider a feasible cache placement and transmission scheme $\mathfrak{F}$. Based on the cached content $M_1, ..., M_{K_3}$, and the transmissions from the server for each request pattern (with rates $r(i, 1), ..., r(i, \lfloor \frac{N_1}{K_3} \rfloor)$, respectively), the K_3 users together must be able to reconstruct all $K_3 \cdot \lfloor \frac{N_1}{K_3} \rfloor$ files. Hence,

$$\sum_{j=1}^{\left\lfloor \frac{N_1}{K_3} \right\rfloor} r_{\mathfrak{F}}(i, j) + \sum_{k=1}^{K_3} M_k \geq K_3 \cdot \left\lfloor \frac{N_1}{K_3} \right\rfloor. \tag{18}$$

Summarizing over all $N_1!$ permutations, we have

$$\sum_{i=1}^{N_1!} \sum_{j=1}^{\left\lfloor \frac{N_1}{K_3} \right\rfloor} r_{\mathfrak{F}}(i, j) \geq \left(\left\lfloor \frac{N_1}{K_3} \right\rfloor \cdot K_3 - K_3 M \right) \cdot N_1!. \tag{19}$$

Note that there are $A_{N_1}^{K_3}$ request patterns $W_j \in \overline{H_i}$, while there are $\left\lfloor \frac{N_1}{K_3} \right\rfloor \cdot N_1!$ subgroups among all the $N_1!$ permutations. By symmetry, each $W_j \in \overline{H_i}$ appears an equal number of times in these subgroups. Hence, the number of times each $W_j \in \overline{H_i}$ appears in the summation in Equation

(19) is $\frac{\left\lfloor \frac{N_1}{K_3} \right\rfloor \cdot N_1!}{A_{N_1}^{K_3}}$. Hence,

$$\begin{aligned}\frac{\sum_{W_j \in \overline{H_i}} r_{\mathfrak{F}}(K, W_j)}{A_{N_1}^{K_3}} &= \frac{1}{\left\lfloor \frac{N_1}{K_3} \right\rfloor \cdot N_1!} \cdot \sum_{i=1}^{N_1!} \sum_{j=1}^{\left\lfloor \frac{N_1}{K_3} \right\rfloor} r_{\mathfrak{F}}(i,j) \\ &\geq \frac{1}{\left\lfloor \frac{N_1}{K_3} \right\rfloor \cdot N_1!} \cdot N_1! (\left\lfloor \frac{N_1}{K_3} \right\rfloor K_3 - K_3 M) \\ &= \frac{\left\lfloor \frac{N_1}{K_3} \right\rfloor K_3 - K_3 M}{\left\lfloor \frac{N_1}{K_3} \right\rfloor}.\end{aligned} \tag{20}$$

We therefore conclude this lemma. ■

Denote the right hand side of Equation (17) by $f(K_3)$. From Lemmas 1, 2, 5 and 6, the minimum expected rate can be bounded by

$$\begin{aligned}R(K, \mathcal{F}, \mathcal{P}) &\geq R(K, \mathcal{F}_1, \mathcal{P}_2) \\ &\geq R(K, \mathbb{W}_3) \\ &= (1 - \pi(K_1)) \cdot R(K, \mathbb{W}_4) \\ &\geq (1 - \pi(K_1)) f(K_3).\end{aligned} \tag{21}$$

Recall that $K_1 \triangleq \lfloor \frac{N_1}{M} \rfloor$ and $K_3 \triangleq \lfloor \frac{1}{2} K_1 \rfloor$. We now consider two cases.

If $\frac{N_1}{M} \leq 6$, it is easy to verify that

$$\begin{aligned}f(K_3) &\geq f(1) \\ &= \frac{M}{N_1} (\frac{N_1}{M} - 1) \\ &\geq \frac{1}{8} (\frac{N_1}{M} - 1).\end{aligned} \tag{22}$$

On the other hand, if $\frac{N_1}{M} > 6$, we have $K_3 \geq \frac{N_1}{2M} - 1 \geq \frac{N_1}{3M}$, and $\left\lfloor \frac{N_1}{K_3} \right\rfloor \geq \lfloor 2M \rfloor$. Since $M \geq 2$, we have

$$\begin{aligned}f(K_3) &= K_3 - \frac{K_3 M}{\left\lfloor \frac{N_1}{K_3} \right\rfloor} \\ &\geq K_3 \left(1 - \frac{M}{\lfloor 2M \rfloor} \right) \\ &\geq \frac{N_1}{3M} \cdot \frac{3}{8} \quad (\text{using} M \geq 2) \\ &\geq \frac{1}{8} (\frac{N_1}{M} - 1).\end{aligned} \tag{23}$$

Using both (22) and (23) into (21), we conclude that the minimum expected rate needed for $\mathbb{W}(K, \mathcal{F}, \mathcal{P})$ is bounded by

$$\begin{aligned} R(K, \mathcal{F}, \mathcal{P}) &\geq 0.28 \cdot \frac{1}{8} \left[\frac{N_1}{M} - 1 \right]_+ \\ &\geq \frac{1}{29} \left[\frac{N_1}{M} - 1 \right]_+ . \end{aligned} \tag{24}$$

D. Second Part of Theorem 1

Now we will move our attention to the unpopular files and prove the other part of the lower bound, i.e., $R(K, \mathcal{F}, \mathcal{P}) \geq \frac{1}{58}[\sum_{i>N_1} Kp_i - 2]_+$. To the best of our knowledge, this treatment of unpopular files has not been reported in the literature. Intuitively, depending on the system setting, the lower bound may be dominated by either the popular files or the unpopular files. Thus, we believe that our capability to quantify the impact of unpopular files is the key reason that we can obtain the improved constant-factor characterization in this paper, even in non-asymptotic settings.

Interestingly, readers will see soon that we will re-apply the results for System 2 constructed in Section IV.A. Recall that in System 2 all users either request one of the non-empty files with a common popularity that is no less than $1/KM$, or request the empty file. For the unpopular files, their popularity is lower than $1/KM$, and hence we cannot directly use the results for System 2. However, next we introduce a new "merging" idea that will eventually allow us to re-apply the results for System 2. Specifically, we will merge several unpopular files into one file, so that the popularity of the new file is no less than $1/KM$. We will show that this merging step will only lower the achievable rate for serving unpopular files. Thus, in the end we obtain a new system similar to System 2, from which a lower bound for the achievable rate of the original system can be derived. The detail analysis is as follows.

Consider another system, where the set of files is $\mathcal{F}_3 = \{F_0, F_{N_1+1}, F_{N_1+2}, ..., F_N\}$ (recall that F_0 is again an empty file). The corresponding popularity distribution is $\mathcal{P}_3 = \{p_0^{'}, p_{N_1+1}, p_{N_1+2}, ..., p_N\}$ where $p_0^{'} = 1 - \sum_{i=N_1+1}^{N} p_i$. In other words, we replace files $F_1, F_2, ..., F_{N_1}$ in the original system

by the empty file and use the corresponding popularity. Similar to Lemma 1, we can prove that

$$R(K, \mathcal{F}, \mathcal{P}) \geq R(K, \mathcal{F}_3, \mathcal{P}_3). \tag{25}$$

Again, we will perform a series of further reductions and finally construct a system with a smaller rate, which can utilize the results in previous analysis of "System 2".

To proceed, we need the lemma below. Denote a file set by $\mathcal{T}_1 = \{T_1, T_2, ..., T_t\}$. Each element T_i can either be a regular file or the empty file. Let its corresponding popularity distribution be $\mathcal{Q}_1 = \{q_1, q_2, ..., q_t\}$, where $\sum_{i=1}^{t} q_i = 1$. Denote another file set by $\mathcal{T}_2 = \{T_1, T_2, ..., T_{t-2}, T_{t+1}\}$ with popularity distribution $\mathcal{Q}_2 = \{q_1, q_2, ..., q_{t-2}, q_{t+1}\}$. Here $q_{t+1} = q_{t-1} + q_t$. In other words, the two files T_{t-1} and T_t in the first system are replaced by *one* file T_{t+1} in the second system. Intuitively, it should be easier (i.e., requiring less cache and lower transmission rate) to serve the second system because there is less "diversity". This statement is made precise below.

Lemma 8: Let $R(K, \mathcal{T}_1, \mathcal{Q}_1)$ be the minimum expected rate required to meet the requests by K users, each of which randomly requests a file in $\mathcal{T}_1$ according to the popularity distribution $\mathcal{Q}_1$. Let $R(K, \mathcal{T}_2, \mathcal{Q}_2)$ be defined similarly for $\mathcal{Q}_2$. We have

$$R(K, \mathcal{T}_1, \mathcal{Q}_1) \geq R(K, \mathcal{T}_2, \mathcal{Q}_2). \tag{26}$$

Proof: The request set for $(K, \mathcal{T}_1, \mathcal{Q}_1)$ is $\mathbb{W}(K, \mathcal{T}_1, \mathcal{Q}_1) = \{W_i\}$, where $W_i = \{f_{i1}, f_{i2}, ..., f_{iK}\}$ and $f_{ij} \in \mathcal{T}_1$. We first construct a mapping from $\mathbb{W}(K, \mathcal{T}_1, \mathcal{Q}_1)$ to $\mathbb{W}(K, \mathcal{T}_2, \mathcal{Q}_2)$.

For every request $W_i \in \mathbb{W}(K, \mathcal{T}_1, \mathcal{Q}_1)$, we map it to a request $W_i^{'} \in \mathbb{W}(K, \mathcal{T}_2, \mathcal{Q}_2)$ as follows. If file T_{t-1} or T_t in $\mathcal{T}_1$ is requested in W_i, we replace it by T_{t+1}.

For a cache placement and transmission scheme $\mathfrak{F}$, suppose that each user can retrieve the file requested in W_i with rate $r_{\mathfrak{F}}(K, W_i)$. Using the same $\mathfrak{F}$, with the replacement of T_{t+1} for T_{t-1} or T_t in both cache placement and transmissions, we can show that the rate $r_{\mathfrak{F}}(K, W_i)$ can also satisfy the request of $W_i^{'}$. Therefore, we have

$$r_{\mathfrak{F}}(K, W_i) \geq r_{\mathfrak{F}}(K, W_i^{'}). \tag{27}$$

Further, if W_i follows the distribution of $\mathbb{W}(K, \mathcal{T}_1, \mathcal{Q}_1)$, $W_i^{'}$ must follow the distribution of $\mathbb{W}(K, \mathcal{T}_2, \mathcal{Q}_2)$. Thus, by the coupling method [19], we must have

$$R_{\mathfrak{F}}(K, \mathcal{T}_1, \mathcal{Q}_1) \geq R_{\mathfrak{F}}(K, \mathcal{T}_2, \mathcal{Q}_2). \tag{28}$$

The results of this lemma then follows. ■

Next, we create a new system $(K, \mathcal{F}_4, \mathcal{P}_4)$ originated from $(K, \mathcal{F}_3, \mathcal{P}_3)$, by merging multiple files in $\mathcal{F}_3$ to a new file in $\mathcal{F}_4$ (described below) and combine their popularity (similar to the mapping from $\mathcal{Q}_1$ to $\mathcal{Q}_2$). We denote this new file set as $\mathcal{F}_4 = \{V_0, V_1, ..., V_{N_2}\}$ and the popularity distribution as $\mathcal{P}_4 = \{v_0, v_1, v_2, ...v_{N_2}\}$. Here, V_0 is the empty file and $v_0 = 1 - \sum_{i=1}^{N_2} v_i$. The other files $V_1, ..., V_{N_2}$ are non-empty files and we pick them in such a way that they all have similar popularity $v_i \approx 1/KM$. Specifically, recall that $\frac{1}{KM} > p_{N_1+1} \geq p_{N_1+2} \geq ... \geq p_N$. Let $h_0 = 0$. Pick h_1 as the smallest integer such that $\sum_{j=N_1+1}^{N_1+h_1} p_j \geq 1/KM$. We then replace files $F_{N_1+1}, ..., F_{N_1+h_1}$ by one file V_1, whose popularity is $v_1 = \sum_{j=N_1+1}^{N_1+h_1} p_j$. Similarly, for each $i = 2, 3, ...$, we pick h_i as the smallest integer such that

$$\sum_{j=N_1+h_{i-1}+1}^{N_1+h_i} p_j \geq 1/KM \tag{29}$$

and then replace files $F_{N_1+h_{i-1}+1}, ..., F_{N_1+h_i}$ by one file V_i, whose popularity is $v_i = \sum_{j=N_1+h_{i-1}+1}^{N_1+h_i} p_j$. In this way, each non-empty file's popularity satisfies $1/KM \leq v_i \leq 2/KM$ for all $1 \leq i \leq N_2$, and we further have $\sum_{i=1}^{N_2} v_i = \sum_{i>N_1} p_i \leq \frac{2N_2}{KM}$.

By applying Lemma 8 iteratively, we can show that

$$R(K, \mathcal{F}_3, \mathcal{P}_3) \geq R(K, \mathcal{F}_4, \mathcal{P}_4). \tag{30}$$

Define $\mathcal{P}_5 = \{1 - \frac{N_2}{KM}, \frac{1}{KM}, ..., \frac{1}{KM}\}$. Similar to Lemma 2, we can prove that

$$R(K, \mathcal{F}_4, \mathcal{P}_4) \geq R(K, \mathcal{F}_4, \mathcal{P}_5). \tag{31}$$

Now, note that the system $(K, \mathcal{F}_4, \mathcal{P}_5)$ is of the same form as the system $(K, \mathcal{F}_1, \mathcal{P}_2)$: all non-

empty files are requested with a common probability that is greater than or equal to $\frac{1}{KM}$ (this is also of the form of the "System 2" that we referred to in Sections III-A and IV-A). Readers can check that the analysis in Sections IV-B and IV-C also applies to the system $(K, \mathcal{F}_4, \mathcal{P}_5)$. Thus, like Equation (24), we have

$$R(K, \mathcal{F}_4, \mathcal{P}_5) \geq \frac{1}{29}\left[\frac{N_2}{M} - 1\right]_+ \geq \frac{1}{58}\left[\sum_{i>N_1} Kp_i - 2\right]_+. \tag{32}$$

Combining Equations (24) and (32), we have proved Theorem 1.

Remark: We believe that the above characterization for the unpopular files is crucial for obtaining the improved constant-factor results that hold for arbitrary distributions and system settings. Take Zipf distribution with $\alpha = 1$ as an example. Suppose that the number of files N is large. Then, it is easy to see that $p_i \approx \frac{1}{i \log N}$, and thus the threshold is $N_1 \approx \frac{KM}{\log N}$. From our earlier results, the lower bound due to popular files is $\frac{N_1}{M} \approx \frac{K}{\log N}$, while the lower bound due to unpopular files is about $\frac{K}{\log N} \cdot \log \frac{N \log N}{KM}$. Note that depending on the relationship between N and K, the term due to unpopular files may be larger or smaller than the term due to popular files. For instance, if we keep N fixed and let $K \to \infty$, then the term due to unpopular files will be dominated by the term due to popular files. Such a setting has been studied in Corollary 1 of [13]. However, in general $N/(KM)$ could be large, and thus the unpopular files may dominate. In that case, if we did not use the characterization in this subsection, we would be unable to obtain the sharper results in this paper.

V. Upper Bound on Expected Transmission Rate

In this section, we will show that the achievable transmission rate of a simple cache-placement and transmission scheme (similar to the RLFU scheme in [11]) provides an upper bound on $R(K, \mathcal{F}, \mathcal{P})$. Recall the following statement of Theorem 2,

$$R(K, \mathcal{F}, \mathcal{P}) \leq \left[\frac{N_1}{M} - 1\right]_+ + \min\left(\sum_{i>N_1} Kp_i, \frac{N - N_1}{[M - N_1]_+} - 1\right). \tag{33}$$

Proof of Theorem 2: We again divide the whole file set into two subsets $\mathcal{F}_1 = \{F_1, F_2, ..., F_{N_1}\}$

and $\mathcal{F}_2 = \{F_{N_1+1}, F_{N_1+2}, ..., F_N\}$. The files in $\mathcal{F}_1$ are the "more popular" files whose popularity is larger than $\frac{1}{KM}$. Recall that in our model each file is of unit length. The minimum indivisible portion of a file is called a "bit". We have assumed that each file has F such bits. The cache placement strategy is given as follows.

Algorithm 1 Cache Placement Procedure

for $1 \leq k \leq K$, $1 \leq n \leq N_1$
  User k randomly caches $\min\left(\frac{MF}{N_1}, F\right)$ bits of the file F_n
end for

Note that we only cache fractions of the N_1 popular files in the users' storage. On the other hand, the files requested by the K users may also come from files in $\mathcal{F}_2$. Assume that there are K_4 users requesting files in $\mathcal{F}_1$ and denote these users as U_1. Denote the other $K - K_4$ users requesting files in $\mathcal{F}_2$ as U_2. For every S that is a subset of U_1 and for every $k \in S$, let $V_{k,S\backslash\{k\}}$ represent all the bits that are requested by user k, that are stored in the cache of every other user of S except user k, and that are not stored in the caches of any other user in $U_1\backslash S$. Denote $\oplus_{k\in S} V_{k,S\backslash\{k\}}$ as the XOR across the sets of bits $V_{k,S\backslash\{k\}}$. More precisely, order the bits in each $V_{k,S\backslash\{k\}}$ in some way. Then, each bit of $\oplus_{k\in S} V_{k,S\backslash\{k\}}$ is the XOR of the corresponding bits across $V_{k,S\backslash\{k\}}, k \in S$. Note that the size of $\oplus_{k\in S} V_{k,S\backslash\{k\}}$ equals to $\max\{|V_{k,S\backslash\{k\}}|, k \in S\}$.

Now we are ready to present the transmission scheme, which econsists of two steps. In the first step, the server will send coded data (as in the decentralized coded caching scheme of [8]) to meet the requests of users in U_1. In the second step, the server sends uncoded data to meet the requests of users in U_2. Recall that the size of U_1 is K_4.

Algorithm 2 Transmission Procedure

Step 1: for $s = K_4, K_4 - 1, ..., 1$
    for every $S \subset U_1$ such that $|S| = s$, do
      Server sends $\oplus_{k\in S} V_{k,S\backslash\{k\}}$
    end for
  end for
Step 2: for every user $k \in U_2$
    Sever sends its requested file d_k
  end for

After both steps, all requests of K users will be satisfied. The reason is as follows. If a user

is in U_2, its request will be immediately satisfied in step 2. If a user k is in U_1, a bit b of its requested file will be in some $V_{k,S_b\backslash\{k\}}$, for a specific set S_b. After step 1, $V_{k,S_b\backslash\{k\}}$ will be retrieved by user k from the transmission received and its local storage. Hence, user K must be able to decode the bit b.

We now compute the rate required by the transmission scheme. This analysis is similar to [8]. We first calculate the rate R_1 sent by the server in step 1. For a subset $S \subset U_1$ and $|S| = s$, a bit of file d_k is in $V_{k,S\backslash\{k\}}$ with probability

$$(\frac{M}{N_1})^{s-1}(1-\frac{M}{N_1})^{K_4-s+1}. \tag{34}$$

The expected number of bits in $V_{k,S\backslash\{k\}}$ is $F\cdot(\frac{M}{N_1})^{s-1}(1-\frac{M}{N_1})^{K_4-s+1}$. When the file size F is large, the number of bits in $V_{k,S\backslash\{k\}}$ is $F\cdot(\frac{M}{N_1})^{s-1}(1-\frac{M}{N_1})^{K_4-s+1}+o(F)$ with high probability. Therefore, the rate needed to be sent for a specific subset S is

$$\begin{aligned} |\oplus_{k\in S} V_{k,S\backslash\{k\}}| &= \max_{k\in S}|V_{k,S\backslash\{k\}}| \\ &= F\cdot(\frac{M}{N_1})^{s-1}(1-\frac{M}{N_1})^{K_4-s+1}+o(F). \end{aligned} \tag{35}$$

In the sequel, we focus on the "large file-size" regime and ignore the factor $o(F)$. For each s, there are $C_{K_4}^s$ subsets S that satisfies $S\subset U_1$ and $|S|=s$. Summing over all possible s and all subsets S, the rate needed in the first step (in the unit of "bit") can be bounded by

$$\begin{aligned} R_1 &\le \sum_{s=1}^{K_4} C_{K_4}^s \cdot F\left(\frac{M}{N_1}\right)^{s-1}\left(1-\frac{M}{N_1}\right)^{K_4-s+1} \\ &= F(1-\frac{M}{N_1})\frac{1-(1-\frac{M}{N_1})^{K_4}}{\frac{M}{N_1}} \\ &< F(\frac{N_1}{M}-1). \end{aligned} \tag{36}$$

Note that this bound does not depend on K_4.

Next, we calculate the rate needed for step 2. Since each user requests a file in $\mathcal{F}_2$ with probability $\sum_{i=N_1+1}^{N} p_i$, the expected rate that it needs in step 2 is $F\sum_{i=N_1+1}^{N} p_i$. Summing over

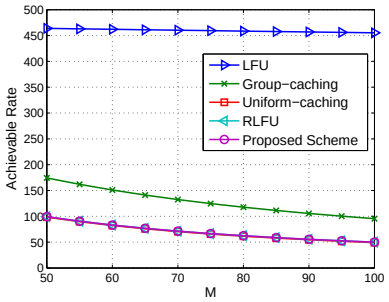

Fig. 2: N=5000,K=500,α=0.2.

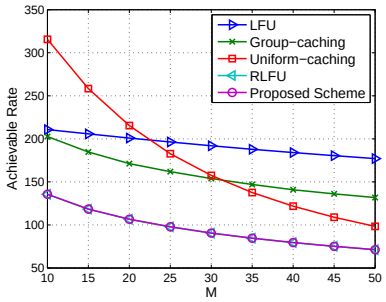

Fig. 3: N=5000,K=500,α=1.1.

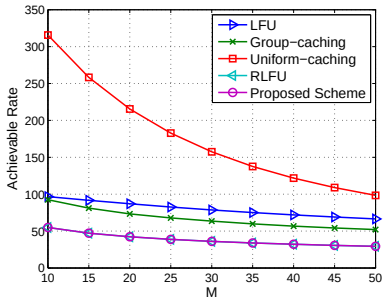

Fig. 4: N=5000,K=500,α=1.4.

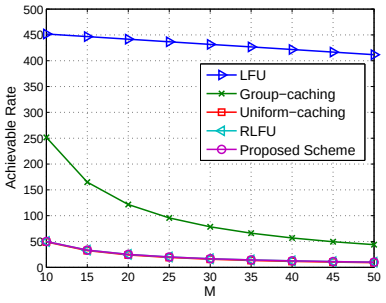

Fig. 5: N=500,K=5000,α=1.

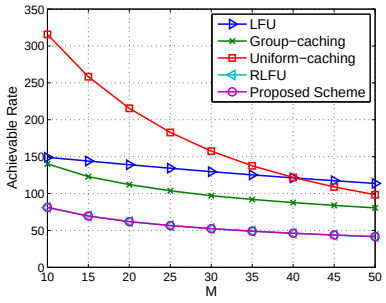

Fig. 6: N=5000,K=500,α=1.4, r=2.

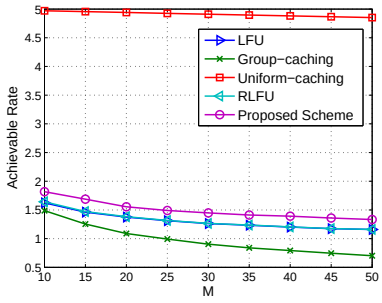

Fig. 7: N=5000,K=5,α=1.4.

all K users, the expected rate of R_2 (again in the unit of "bit") can be represented by

$$R_2 \leq KF \sum_{i=N_1+1}^{N} p_i. \tag{37}$$

Note that it is possible that N_1 is smaller than M. In that case, after each user caches all N_1 popular files, there are still some space left for caching unpopular files. Therefore, when $N_1 < M$, we let the remaining storage to randomly cache a equal portion of files F_i ($i > N_1$). Similar to the proof of R_1, the expected rate of R_2 can also be bounded by

$$R_2 \leq F(\frac{N - N_1}{M - N_1} - 1). \tag{38}$$

Finally, by a conversion from the unit of "bit" to the unit of "file" (recall that each file is unit length), the result of Theorem 2 then follows from Equation (36), (37) and (38).

VI. Simulation Results

We next present numerical results that demonstrate the superior performance of the proposed scheme and discuss the insights from the results.

We will compare with four other schemes. The first one is an uncoded scheme, i.e., least-frequently used (LFU) caching strategy [18], which caches the M most popular files in all users' storage. The second scheme is the decentralized uniform coded caching scheme in [8], where a $\frac{M}{N}$ portion of every file is cached in each user's storage, regardless of its popularity. The remaining two are group-caching [9] and RLFU [11], both of which consider heterogenous popularity. In group-caching [9], files with close popularity (differing by at most a factor of 2) are grouped together. The scheme in [9] assigns an equal fraction of the cache space to each group, and performs coded transmission only among users requesting files from the same group. This equal allocation of cache space is found to perform poorly in our experiments (not reported). Hence, in our simulation, we further allow group-caching to optimize the allocation of cache space to each file. However, as we will see shortly, the requirement that coded transmission is only performed among users requesting files from the same group still becomes a limiting factor, and as a result the performance of group-caching can be even worse than uniform coded-caching.

Finally, as we discussed earlier, the RLFU scheme proposed in [11] is similar to ours in that they both evenly cache files whose popularity is above a threshold. Note that [11] presents two ways for choosing the threshold. One set of thresholds (parameterized by the parameters of the Zipf-distribution) was used to prove the theoretical results. It turns out these theoretical values for the thresholds tend to perform poorer in our experiments (not shown). [11] also proposes another way of setting the threshold, by taking an additional optimization step over all possible threshold values. In our simulation results below, we compare with this version of RLFU with optimized threshold values. Since our proposed algorithm (using the simple threshold N_1) can also be viewed as a member in the class of RLFU algorithms, the optimized version of RLFU will clearly achieve better performance. What is interesting, however, is that in most of the simulation settings below, our simple choice N_1 performs almost as well as the optimized version of RLFU,

which suggests that the simple choice N_1 is in fact quite close-to-optimal.

After the cache placement setting, we simulate the request and transmission processes. The requests of all users are generated randomly according to the file popularity distribution. The rate of each scheme is calculated as follows. 1) For files that are not cached at all but are requested, the rate is calculated as the distinct number of such files. 2) For files that are cached, the rate is calculated as follows, depending on the scheme. For LFU, a cached file is always cached in its entirety. Thus, the rate is zero. For other schemes, a file may be partially cached. Specifically, the rate for uniform caching is $(1-\frac{M}{N})(1-(1-\frac{M}{N})^K)/(\frac{M}{N})$, similar to (36). The rate for group caching is the summation of the rates for each group, with uniform caching applied within each group. The rate for RLFU and our scheme is calculated by applying uniform caching to only the popular files. The threshold for popular files in RLFU is optimized according to Eqs. (1) and (4) in [11].

In the following figures, we present the mean transmission rate calculated from a large number (> 1000) of request patterns randomly generated according to the popularity distribution. The confidence intervals are very small and thus not shown.

Comparison under Zipf popularity distribution: The first set of numerical results are for Zipf popularity distribution, i.e., the popularity of the i-th popular file is $p_i = \frac{H(\alpha)}{i^\alpha}$, where α is the Zipf exponent and $H(\alpha)$ is the normalization factor. Note that $\alpha > 1$ means that the distribution is heavily skewed to the most popular files, while $\alpha < 1$ means that the distribution is "flatter". We simulate a system with $K = 500$ users and $N = 5000$ files. In Figures 2-4, we plot the achievable transmission rate as a function of users' storage size M. As readers can see, our proposed scheme achieves the best performance under all scenarios. Specifically, in Figure 2, $\alpha = 0.2$ is small, which implies that files have a "flat" popularity distribution. We can observe that LFU performs poorly because it doesn't exploit coded transmission opportunities. A deeper investigation reveals that both RLFU and our scheme turn into uniform coded caching, which outperform group-caching. In contrast, in Figure 4, $\alpha = 1.4$ is large, which implies that a small fraction of files dominate the popularity distribution. Uniform coded-caching performs poorly

because it neglects the significant popularity difference. On the other hand, by preferably caching the most popular files, both LFU, RLFU, group-caching and our scheme all perform well.

Finally, from Figures 2-4, we can see that group-caching appears to also exhibit robust performance, except in Figure 5. On the other hand, our scheme and RLFU consistently performs better. We do emphasize that the simulation of both RLFU and group-caching involves an extra step of optimizing cache allocation across groups. In contrast, our scheme is much simpler and does not involve such an additional optimization step. Thus, our proposed scheme not only achieves better performance, but also is easy to implement.

Comparison under non-Zipf distribution: Next, we simulate these algorithms under a non-Zipf distribution, i.e., Zipf-Mandelbrot law distribution, where the i-th popular file is requested with probability $p_i = \frac{H(\alpha)}{(i+r)^\alpha}$ for a constant $r = 2$ and $\alpha = 1.4$. The simulation results are presented in Figure 6. Again, our scheme and RLFU achieve the best performance.

As we mentioned earlier, in most of the simulations (Figs. 2-6), our proposed schedule achieves the same performance as the optimized version of RLFU [11], even though our choice of threshold N_1 is very simple and does not involve the optimization step in [11]. We note that there are indeed cases where the optimized version of RLFU performs strictly better. An example is shown in Fig. 7. Note that in this setting, since K is small, the threshold popularity $\frac{1}{KM}$ is large. Hence, the threshold N_1 may be even smaller than M. In contrast, a closer inspection indicates that RLFU turns into LFU, i.e., the most popular M files are cached in their entirety. What is interesting is that group-caching turns out to out-perform RLFU. For this setting, it turns out that group caching scheme divides the files into more than 2 groups. As a result, some of the most popular files are caching entirely, the less popular files are cached uniformly, and the least popular files are not cached at all. This simulation result thus suggests that, in some cases, it may be useful to use 2 thresholds, instead of 1 threshold as in our scheme and RLFU. We leave the design of such 2-threshold scheme as a topic for future work.

VII. Conclusion

In this work, given an arbitrary popularity distribution, we first derive a new information-theoretical lower bound on the expected transmission rate of any coded caching schemes. We then show that a simple coded-caching scheme attains an expected transmission rate that is at most a constant factor away from the lower bound. Unlike other existing studies, the constant factor that we derived is independent of the popularity distribution.

There are a number of interesting questions for future studies. First, the complexity of the transmission scheme in Section V can be high (esp. for enumerating all the subsets S). Thus, an important question is whether we can develop low-complexity transmission schemes that still attain similar performance guarantees. Further, it would be interesting to study how the benefits of coded caching can be extended to wireless environments (in particular heterogeneous wireless networks).

Acknowledgment

This work was partially supported by a grant from the Army Research Office W911NF-14-1-0368, a grant from NSF China (No. 61325012), and the CSC scholarship.

References

[1] J. Zhang, X. Lin and X. Wang, "Coded Caching under Arbitrary Popularity Distributions", in *Information Theory and Applications Workshop*, UCSD, USA, Feb. 2015.

[2] White Paper, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018", Feb. 2014.

[3] D. Wessels, *Web Caching*. O'Reilly, 2001.

[4] N. Golrezaei, K. Shanmugam, A.G. Dimakis, A.F. Molisch and G. Caire, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers", in *Proc. IEEE INFOCOM*, Orlando, USA, Mar. 2012.

[5] V. Shah and G. de Veciana, "Performance Evaluation and Asymptotics for Content Delivery Networks", in *Proc. IEEE INFOCOM*, Toronto, Canada, Apr. 2014.

[6] B. Tan and L. Massoulie, "Optimal Content Placement for Peer-to-Peer Video-on-Demand Systems", *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 566-579, Apr. 2013.

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

[8] M. A. Maddah-Ali, and U. Niesen, "Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff", *IEEE/ACM Trans. Netw.*, to appear.

[9] U. Niesen, and M. A. Maddah-Ali, "Coded Caching with Nonuniform Demands", *arXiv:1308.0178v2 [cs.IT]*, Mar. 2014.

[10] J. Hachem, N. Karamchandani and S. Diggavi, "Multi-level Coded Caching", *arXiv:1404.6563 [cs.IT]*, Apr. 2014.

[11] M. Ji, A. Tulino, J. Llorca and G. Caire, "On the Average Performance of Caching and Coded Multicasting with Random Demands", in *11th International Symposium on Wireless Communication Systems (ISWCS),* Barcelona, Spain, Aug. 2014, pp. 922-926.

[12] M. Ji, A. Tulino, J. Llorca and G. Caire, "Order Optimal Coded Caching-Aided Multicast under Zipf Demand Distributions", *arXiv:1402.4576v1 [cs.IT]*, Feb. 2014.

[13] M. Ji, A. Tulino, J. Llorca and G. Caire, "Order-Optimal Rate of Caching and Coded Multicasting with Random Demands", *arXiv:1502.03124v1 [cs.IT]*, Feb. 2015.

[14] N. Karamchandani, U. Niesen, M. A. Maddah-Ali and S. Diggavi, "Hierarchical Coded Caching", *arXiv:1403.7007v2 [cs.IT]*, Jun. 2014.

[15] M. Ji, A. Tulino, J. Llorca and G. Caire, "Order Optimal Coded Delivery and Caching: Multiple Groupcast Index Coding", *arXiv:1402.4572 [cs.IT]*, Feb. 2014.

[16] R. Pedarsani, M. A. Maddah-Ali and U. Niesen, "Online Coded Caching", *arXiv:1311.3646 [cs.IT]*, Nov. 2013.

[17] R. Kaas and J.M. Buhrman, "Mean, Median and Mode in Binomial Distributions", *Statistica Neerlandica*, vol. 34, no. 1, pp. 13-18, Mar. 1980.

[18] D. Lee, S. H. Noh, S. L. Min, J. Choi, J. H. Kim, Y. K. Cho and C. S. Kim, "LRFU: A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies", *IEEE Trans. Computers*, vol. 50, no. 12, pp. 1352-1361, 2001.

[19] T. Lindvall, *Lectures on the Coupling Method*. Wiley, New York, 1992.