# Optimal Decentralized Coded Caching for Heterogeneous Files

Han Cheng*, Chenglin Li †, Hongkai Xiong *, and Pascal Frossard †
* Department of Electronic Engineering, Shanghai Jiao Tong University, China
† Signal Processing Laboratory (LTS4), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Email: *{chenghan, xionghongkai}@sjtu.edu.cn, †{chenglin.li, pascal.frossard}@epfl.ch

***Abstract*—Caching is a technique to reduce the peak network load by pre-fetching some popular contents at the local caches of end users. Coded caching can facilitate and exploit the coded-multicasting opportunities for users with different demands, resulting in an additional and significant reduction of the peak traffic. However, most existing researches on coded caching are limited by the assumption that all files to be delivered have the same size. We show in this paper that current schemes can only achieve suboptimal performance when the files have different sizes. To address this, we propose a novel optimization strategy for coded caching that minimizes the worst-case transmission rate of multicasting the coded content upon users requests, subject to the storage constraint at the local caches, by the optimal allocation of the caching proportion among heterogeneous files. In order to efficiently solve this problem, we develop a practical algorithm by using the Lagrange multiplier method and the sequential quadratic programming (SQP) approach. Experiment results show that the worst-case transmission rate can be reduced by the proposed scheme compared to state-of-the-art coded caching schemes. It certainly offers an important advantage in the deployment of data delivery systems.**

***Index Terms*—Caching, coded caching, heterogeneous content distribution.**

## I. Introduction

The amount of the global mobile traffic continues to grow rapidly with nearly 75% of the traffic will be resource demanding video data by 2020 [1]. This causes more stress on the communication networks and data delivery systems. Furthermore, the data traffic generally presents high temporal variability, which leads to congestion during peak traffic hours and under-utilization during off-peak hours. To reduce the peak traffic, caching is proposed to utilize the storage space of local caches across the network and to perform content placement during off-peak hours, thereby smoothing out the temporal traffic variability and reducing congestion and access latencies in peak hours [2]. Caching techniques can be divided into two families, namely the uncoded and coded caching schemes.

Traditional uncoded caching schemes achieve the local caching gain by pre-fetching some popular contents subject to the cache storage constraint [3]. A fundamental work on coded caching is proposed by Niesen *et al.* [2] that achieves an important caching gain via the joint optimization of both the data placement and delivery phases to increase the coding opportunities. This scheme can significantly reduce the transmission rate in the shared link by carefully designing a centrally coordinated placement phase. The same authors propose a decentralized coded cache scheme in [4], where the content is independently and randomly placed at each cache in a decentralized manner without a central coordination between caches, at the price of only modest performance loss. For different applications, coded caching schemes are studied by the follow-up works, such as for the non-uniform demands [5], for the delay-sensitive content [6]. All these works usually assume that the coded (sub)files have the same size and that local caches therefore pre-fetch the same proportion of data for each file.

Considering the case when files have distinct sizes, the work in [5] tailors the main body of files into the same size to implement coded caching scheme and distributes the rest in the traditional way. In [7], an achievable coded caching scheme using a caching proportion that increases proportionally with the file size is proposed, where caching proportion denotes the proportion of each file pre-fetched in the local cache. However, all of these schemes can only achieve suboptimal performance, since the potential gain of the transmission rate reduction by an appropriate proportion allocation is under exploited.

In practice, however, the files to be delivered are likely to have different sizes (e.g., videos of different durations or different resolutions). We therefore propose here an optimal decentralized coded caching scheme for the situation where files have distinct sizes. We formulate an optimization problem that selects the best coded caching strategy in order to minimize the transmission rate required by the shared link. By solving the proposed problem, we derive the optimal allocation of proportion of data that needs to be cached for each filesize, under constraints on the storage space of the local caches. In order to decrease the computational complexity, we relax the original optimization problem and develop a practical algorithm to solve the relaxed problem by the Lagrange multiplier method and the SQP approach. We depict a modified optimization problem to cope with the performance degradation introduced by excessive relaxation. Finally, we carry out numerical simulations to compare the proposed coded caching scheme with competitor schemes. Experiments show that the proposed scheme can further reduce the worst-

The work has been partially supported by NSFC under Grants 61501293, 61529101, 61425011, 61622112 and 61472234, the Program of Shanghai Academic Research Leader under Grant 17XD1401900, the China Postdoctoral Science Foundation under Grants 2016T90372 and 2015M570365, and the China Scholarship Council.

ISBN 978-0-9928626-7-1 

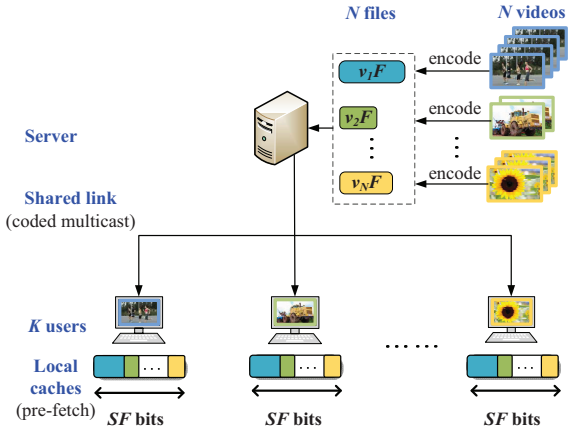



Fig. 1. An illustrative caching network for video applications.

case transmission rate by computing the ideal proportion of cached data for files of different sizes.

The remainder of the paper is organized as follows. Section II describes the system model and design challenges. In Section III, we formulate the optimization problem for the caching proportion allocation of the decentralized coded caching. In Section IV, we develop a practical algorithm to solve this optimization problem. Section V presents the simulation results. The concluding remarks are given in Section VI.

## II. Problem Setting and Design Challenges

The network architecture considered in this paper is shown in Fig. 1. The server stores $N$ files that are encoded bit streams of $N$ raw videos. Due to the diversity of durations, resolutions, and/or rate-distortion behaviors of videos, these $N$ files are with different file sizes. Denote $\mathcal{N}$ as the set of these $N$ files, and assume that each file $n \in \mathcal{N}$ has a file size of $v_n F$ bits, where $F$ is the unit size of files. The file size distribution is then denoted as $V = [v_1, v_2, \ldots, v_n, \ldots, v_N]$. In order to download and watch these video files, K users are connected to the server through a shared link that is considered error-free. Denote the set of these $K$ users as $\mathcal{K}$. Each user $k \in \mathcal{K}$ has a local cache capable of storing $SF$ bits.

We then consider the data delivery process driven by a decentralized coded caching scheme as illustrated in Algorithm 1. In the placement phase, each user pre-fetches (without coding operations) independently and randomly a portion of each file according to a caching proportion allocation $P = [p_1, p_2, \ldots, p_n, \ldots, p_N]$, where $p_n$ is the proportion of the file $n$ that is cached locally. In other words, for each file $n$, the local cache of each user $k$ will randomly cache $p_n \cdot v_n \cdot F$ bits. In contrary to the decentralized coded caching scheme [4] where each file is cached with the same proportion, Algorithm 1 permits to have different proportions for each file.

In the delivery phase, each user sends a request $d_k$. Denote $d = [d_1, d_2, \cdots, d_K]$ as the vector of users' requests and $\mathcal{D}$ as the set of all possible request vectors. The server then collects the users' requests, and multicast the coded content to users through the shared link. In Line 11 of Algorithm 1, $\Omega_{k,\mathcal{I}\setminus\{k\}}$ denotes the bits of the file requested by user $k$ that are cached exclusively at users within the set $\mathcal{I} \setminus \{k\}$ and un-cached by every user outside $\mathcal{I}\setminus\{k\}$. The $\oplus$ operation represents the bit-wise XOR operation. To implement this operation, all elements $\Omega_{k,\mathcal{I}\setminus\{k\}}, \forall k \in \mathcal{I}$ are assumed to be zero padded to the length of the longest element. In other words, the transmission rate is determined by the longest element. When the files are in different sizes, the elements $\Omega_{k,\mathcal{I}\setminus\{k\}}$ are usually different, thus a part of the transmission rate will be wasted. This is the fundamental reason why an appropriate proportion allocation $P$ can improve the performance of the coded caching scheme. Finally, each user can recover the requested file by utilizing the uncoded content pre-fetched at the local cache and the coded content that is multicasted and received from the server.

**Algorithm 1** Decentralized coded caching scheme for distinct file sizes

1: **Placement Phase**
2: determine $P = [p_1, p_2, \cdots p_n]$;
3: **for** $k = 1, 2, \ldots, K$ **do**
4: **for** $n = 1, 2, \ldots, N$ **do**
5: user $k$ randomly pre-fetches $p_n \cdot v_n \cdot F$ bits of file $n$.
6: **end for**
7: **end for**
8: **Delivery Phase**
9: **for** $i = K, K-1, \ldots, 1$ **do**
10: **for** $\mathcal{I} \subset \mathcal{K} : |\mathcal{I}| = i$ **do**
11: the server multicasts $\oplus_{k\in\mathcal{I}}\Omega_{k,\mathcal{I}\setminus\{k\}}$ to all users.
12: **end for**
13: **end for**

TABLE I
TRANSMISSION RATE OF DIFFERENT SCHEMES

| Caching proportion allocation | Worst-case transmission rate |
|---|---|
| $P_1 = [0.75, 0.75]$ | $4.375F$ |
| $P_2 = [(0.75, 0.75, 0.75), (0.75, 0.75)]$ | $4.375F$ |
| $P_3 = [0.86, 0.57]$ | $4.5026F$ |
| $P_4 = [0.81, 0.65]$ | $3.9845F$ |

Next, we show the importance of the appropriate allocation of caching proportion $P$ though a motivating example below. Consider a caching problem of $N = 2$ files $A$ and $B$, with corresponding size $V = [v_1, v_2] = [15, 10]$, normalized by the unit size $F$ bits. There are $K = 2$ users each with a local cache of size $S = 18.75$, normalized by the unit size $F$ bits. As shown in Table I, $P_1$ corresponds to the decentralized coded caching scheme [4] where each file has the same caching proportion. To address the above transmission rate waste problem caused by zero padding operations, a straightforward intuition is to break the larger files into several subfiles of the same length. Then we can apply the decentralized coded caching scheme in [4] to these subfiles, and pre-fetch the same proportion for each subfile. This proportion allocation is denoted by $P_2$. Another possible solution is to allocate the

Due to the space limit, we only outline the comparison of the worst-case transmission rate here. For the detailed derivation process and the theoretical proof, please refer to our technical report [8].

ISBN 978-0-9928626-7-1 © EURASIP 2017

caching proportion for each file to be linearly proportional to the file size [7], which is referred to as $P_3$. It can be seen that these three schemes fail to achieve the best coded caching gain in terms of the worst-case transmission rate, since there exists an optimal caching proportion allocation $P_4 = [0.81, 0.65]$. Hence, an appropriate allocation of $P$ is the key point to minimizing the worst-case transmission rate over the shared link. This is exactly the objective of the optimization problem described in the next section.

## III. Optimization Problem

### A. Primal Optimization Problem

**Lemma 1** *For a given file size distribution $V$ and a fixed caching proportion allocation $P$, the transmission rate incurred by serving a request vector $d = [d_1, d_2, \ldots, d_K]$ is*

$$R(d,P,V) = \sum_{i=1}^{K} \sum_{\substack{\mathcal{I}\subset\mathcal{K} \\ |\mathcal{I}|=i}} \max_{j\in\mathcal{I}} \{{p_{d_j}}^{i-1}(1-p_{d_j})^{K-i+1} v_{d_j} F\}. \quad (1)$$

*Proof:* For a particular bit in any file $n$, the probability for this bit to be pre-fetched by any user is $p_n$. For any $i$-sized subset of $\mathcal{K}$, the probability that this bit is pre-fetched by those $i$ users is ${p_n}^i(1-p_n)^{K-i}$. Hence, the average number of bits of file $n$ that are cached at those $i$ users is ${p_n}^i(1-p_n)^{K-i}\cdot v_n F$. Since $\mid \mathcal{I}\setminus\{k\} \mid = i-1$, the expected size of $\Omega_{k,\mathcal{I}\setminus\{k\}}$ is

$${p_{d_k}}^{i-1}(1-p_{d_k})^{K-i+1} v_{d_k} F. \quad (2)$$

Considering the iterations in Algorithm 1, we can further generalize Eq. (2) to the total transmission rate as shown in Eq. (1), where the two summation symbols correspond to the two for-loops and the maximization operation relates to the zero padding operation during coding. ∎

The optimization problem for the decentralized coded caching can be formulated as

$$\textbf{OPT1:} \quad \min_{P} \quad \max_{d\in\mathcal{D}} R(d,P,V) \quad (3a)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} p_n v_n F \le SF, \quad 0 \le p_n \le 1, \quad (3b)$$

where the optimization objective in Eq. (3a) is to minimize the maximum (i.e., the worst-case) transmission rate under different situations of users' requests. The local caching storage constraint in Eq. (3b) ensures that the total size of cached content does not exceed the caching capacity. Here, we are concerned about the worst-case transmission in the optimization formulation for the following reasons. In general, the quality of experience (QoE) of users degrades greatly if the users cannot receive enough data from the server to recover the requested video file. This may occur when the total traffic in the delivery phase exceeds the capacity of the shared link. To avoid such QoE degradation and satisfy arbitrary file requests from the users, we focus on the minimization of the worst-case transmission rate, instead of the average transmission rate.

It can be found that the computational complexity to solve problem **OPT1** increases exponentially with N and K, which causes the practical solution infeasible for larger system settings. Therefore, in the following, we reformulate a relaxed optimization problem for **OPT1**, which has an approximate solution but with low computational complexity.

### B. Relaxed Optimization Problem

Observing the transmission rate model in Eq. (1), the number of subsets for $\mathcal{I}\subset\mathcal{K}, |\mathcal{I}| = i$ is $\binom{i}{K}$. On the other hand, $\max_{j\in\mathcal{I}}\{{p_{d_j}}^{i-1}(1-p_{d_j})^{K-i+1}\cdot v_{d_j}F\} \le \max_{n\in\mathcal{N}}\{{p_n}^{i-1}(1-p_n)^{K-i+1}\cdot v_n F\}$ holds for any subset $\mathcal{I}\subset\mathcal{K}$ since $d_j \in \mathcal{N}, \forall j \in \mathcal{I}$. Therefore, for any situation of users' demand $d\in\mathcal{D}$, we have the following relaxation

$$\begin{aligned} R(d,P,V) &\le \sum_{i=1}^{K} \binom{i}{K} \max_{n\in\mathcal{N}} \{{p_n}^{i-1}(1-p_n)^{K-i+1} v_n F\} \\ &\triangleq R_{max}. \end{aligned} \quad (4)$$

Since the relaxation in Eq. (4) is always true for any demand situation, we can replaced the objective function of **OPT1** by $R_{max}$ in Eq. (4) to obtain a relaxed optimization problem.

## IV. Algorithm Design

### A. Sequential Quadratic Programming Approach

Specifically, the relaxed optimization problem of **OPT1** can be transformed to an equivalent problem **OPT2**, by introducing an auxiliary vector variable $Z = [z_1, z_2, \ldots, z_K]$, as follows,

$$\textbf{OPT2:} \quad \min_{P} \quad \sum_{i=1}^{K} \binom{i}{K} z_i F \quad (5a)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} v_n p_n F \le SF, \quad 0 \le p_n \le 1, \quad (5b)$$

$${p_n}^{i-1}(1-p_n)^{K-i+1} v_n F \le z_i F, \forall i \in \mathcal{K}, n \in \mathcal{N}. \quad (5c)$$

With the Lagrange multiplier method, **OPT2** can be converted to an unconstrained problem, where $\mu \ge 0$ and $\eta_{in} \ge 0$ are the lagrange multipliers corresponding to the constraints in Eqs. (6b) and (6c), respectively.

$$\begin{aligned} \min \quad L(P,Z,\mu,\eta) &= \sum_{i=1}^{K} \tbinom{i}{K} z_i F + \mu\cdot[\sum_{n=1}^{N} p_n\cdot v_n - S]F \\ &+ \sum_{i=1}^{K}\sum_{n=1}^{N} \eta_{in}\cdot[{p_n}^{i-1}(1-p_n)^{K-i+1}\cdot v_n - z_i]F. \end{aligned} \quad (6)$$

Obviously, problem **OPT2** is non-convex. In the following, a practical algorithm is proposed for the optimization problem based on Karush-Kuhn-Tucker (KKT) condition and SQP method. To solve the first-order necessary conditions of optimality for problem **OPT2**, the SQP method can be used to construct a quadratic programming (QP) subproblem at a given approximate solution, and then the solution is introduced to this subproblem to obtain a better approximation. This process is iterated to construct a sequence of approximations that is expected to converge to the optimal solution $(P^*, Z^*, \mu^*, \eta^*)$. Specifically, given an iterate $(P^t, Z^t, \mu^t, \eta^t)$, a new iterate

ISBN 978-0-9928626-7-1 © EURASIP 2017

$(P^{t+1}, Z^{t+1}, \mu^{t+1}, \eta^{t+1})$ can be derived by solving the QP minimization subproblem:

$$\min_{\delta^t} \quad \nabla_Z \left[\sum_{i=1}^{K} \binom{i}{K} z_i^t F\right]^T \delta_Z^t + \frac{1}{2}\delta^{t^T} \nabla^2 L(P^t, Z^t, \mu^t, \eta^t)\delta^t \tag{7a}$$

$$\text{s.t.} \quad \nabla_P \left[\sum_{n=1}^{N} v_n p_n^t F\right]^T \delta_P^t + \sum_{n=1}^{N} v_n p_n^t F = SF, \tag{7b}$$

$$\nabla_P \left[{p_n^t}^{i-1}(1-p_n^t)^{K-i+1} v_n F\right]^T \delta_P^t - \delta_{z_i}^t F + (p_n^t)^{i-1}(1-p_n^t)^{K-i+1} v_n F = z_i^t F, \forall i \in \mathcal{K}, n \in \mathcal{N}, \tag{7c}$$

where the derivative operators $\nabla$ and $\nabla^2$ refer to the first-order gradient vector and the second-order Hessian matrix with regard to the primal variables $(P, Z)$, respectively. $\delta^t = (P^{t+1} - P^t, Z^{t+1} - Z^t)^T$ is defined as the vector standing for the update direction of primal variables, and we further denote $\delta_P^t = (P^{t+1} - P^t)^T$ and $\delta_Z^t = (Z^{t+1} - Z^t)^T$, respectively. Thus, an SQP algorithm is proposed for the relaxed optimization problem **OPT2**, as illustrated in Algorithm 2, where $t_{max}$ denotes the maximum number of iterations for the SQP method.

### B. Modified Solution

As will be shown in the next section, Algorithm 2 (the solution to the relaxed optimization problem **OPT2**) performs well when the number of files $N$ is larger than the number of users $K$. In this case, the relaxation of the objective function $R_{max}$ in problem **OPT2** is still acceptable, since usually the worst case occurs when users request the largest $K$ files, which is a subset of all files. When $N \leq K$, however, this relaxation becomes excessive, since the users are more than the files and the transmission rate for $i = 1$ in Eq. (4) is duplicately computed for the uncached subset of a file that is requested by two or more users. The worst case transmission when $N \leq K$ usually occurs when all files are requested by users. Therefore, we need to revise the optimization problem for the case of $N \leq K$, by adopting a tighter relaxation $R'_{max}$:

$$R'_{max} = \sum_{n=1}^{N}(1-p_n)^K v_n F + \sum_{i=2}^{K} \binom{i}{K} \max_{n\in\mathcal{N}} \{{p_n}^{i-1}(1-p_n)^{K-i+1} v_n F\}. \tag{8}$$

Eq. (8) is based on the fact that, when $i = 1$, at most $\sum_{n=1}^{N}(1-p_n)^K \cdot v_n F$ bits need to be transmitted by the server in the delivery phase. Therefore, replacing the objective function in **OPT1** by $R'_{max}$ in Eq. (8), we can get the modified optimization problem. Then, the similar approach to Algorithm 2 in Section IV-A can be applied to obtain the optimal solution to this modified optimization problem, which is omitted here due to the space limit.

**Algorithm 2** Determining the Optimal Caching Proportion $P$

1: **Initialization Step**
2: Set $(P^0, Z^0, \mu^0, \eta^0)$ to some non negative numbers;
3: **Iteration Step**
4: **for** $t = 0, 1, 2, \ldots, t_{max}$ **do**
5: Obtain $\delta^t$ and $(\mu^{t+1}, \eta^{t+1})$ by solving Eq. (7);
6: **if** $\mu^{t+1} < 0$ or ${\eta_{in}}^{t+1} < 0$ **then**
7: project it onto the set of nonnegative real numbers;
8: **end if**
9: **if** $\delta^t = 0$ **then**
10: Break;
11: **end if**
12: Update $(P^{t+1}, Z^{t+1})^T = (P^t, Z^t)^T + \delta^t$.
13: **end for**

## V. Experimental Results

In this section, we compare the performance of the proposed scheme with two baseline schemes: 1) *the uniform* $P$, each local cache pre-fetches files with an uniform proportion [4], and 2) *the proportional* $P$, each local cache pre-fetches files in a proportion which is linearly proportional to file size [7].

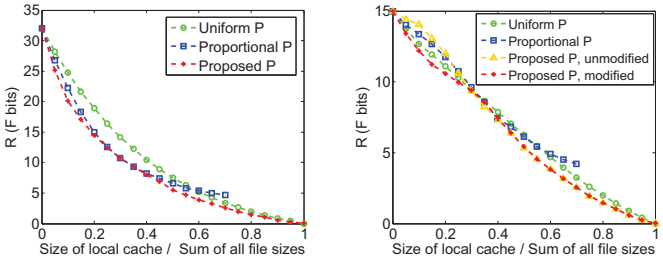
### A. Performance Comparison

Assume that there are four users, $K = 4$, and two file settings: $N = 16$, $V = [8, 8, 8, 8, 4, 4, 4, 4, 2, 2, 2, 2, 1, 1, 1, 1]$, and $N = 4$, $V = [8, 4, 2, 1]$ respectively. In Fig. 2, we illustrate the worst-case transmission rate achieved by different algorithms for these two file settings under different local cache capacity constrains. Here, the size of each local cache is represented by the total caching proportion, which is defined as the ratio between the size of the local cache and the sum of the sizes of all files. Note that for the proportional $P$ allocation, when the total caching proportion is greater than 0.7, the proportion calculated for file size 8 is greater than 1, then the scheme adopting a caching proportion linearly proportional to the file size is meaningless. Thus the performance of this scheme is only evaluated when the total caching proportion is smaller than 0.7. It can be seen that when the number of files is larger than the number of users, the optimal caching proportion allocation of $P$ by Algorithm 2 can effectively reduce the worst-case transmission rate, as shown in Fig. 2(a). In Fig. 2(b), however, when the number of files is comparable with the number of users, the performance of proposed scheme is better than the other two schemes only when the total caching proportion is greater than 0.3. As explained in Section IV-B, the main reason that causes the deficiency of the proposed scheme is the excessive relaxation in Eq. (4) when multiple users are requesting the same file. Therefore, we also illustrate the performance of the modified solution in Fig. 2(b). It can be seen that after the modification, the performance of the proposed scheme is significantly improved when the total caching proportion is smaller than 0.3.

### B. Experimental results on larger scale settings

Most of the previous studies [4], [7], focus on the application scenario where the number of files is much larger than the number of users, namely $N \gg K$. To evaluate

ISBN 978-0-9928626-7-1 © EURASIP 2017



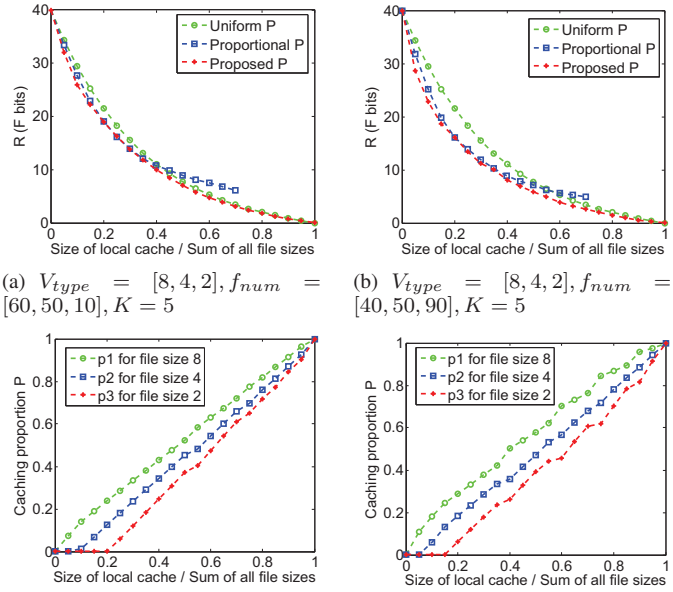

(a) $N = 16, K = 4, V = [8, 8, 8, 8, 4, 4, 4, 4, 2, 2, 2, 2, 1, 1, 1, 1]$ (b) $N = 4, K = 4, V = [8, 4, 2, 1]$

Fig. 2. Worst-case transmission rate vs. total caching proportion.

the performance of the proposed decentralized coded caching scheme for this situation, we perform numerical simulations on larger scale settings. Denote $V_{type} = [v_{t_1}, v_{t_2} \dots]$ as the type of file size and $f_{num} = [f_{n_1}, f_{n_2} \dots]$ as the number of files for each type of file size. Assume that there are $K = 5$ users and three types of file size, $V_{type} = [8, 4, 2]$. For each type, there are a number of files stored in the server. Assume that the total size of all files is fixed at $700F$ bits. Then two file settings are considered: $f_{num} = [60, 50, 10]$ and $f_{num} = [40, 50, 90]$. For the first setting, the number of files with larger file size is greater than the number of files with smaller file size, and an opposite trend is adopted by the second file setting.

Fig. 3(a) and Fig. 3(b) show the worst-case transmission rate achieved by different schemes. We can clearly see that choosing an appropriate caching proportion $P$ through the proposed scheme can effectively improve the caching performance compared to the other two schemes. To have a further insight of the proposed scheme and provide some practical design guidelines, in Fig. 3(c) and Fig. 3(d) we illustrate the change of the optimal caching proportion allocation $P = [p_1, p_2, p_3]$ when the total caching proportion varies. The files of the same type (with the same size) are allocated with the same caching proportion. It can be seen that the general trend of the optimal caching proportion allocation $P$ is to allocate a larger proportion to files with a larger size, however it is not linearly proportional to the file size. Actually, when the total caching proportion begins to increase from 0, the files with the largest size begin to be pre-fetched, then files with the second largest size begin to be pre-fetched. Namely the files begin to be pre-fetched successively according to their sizes. Once a file begins to be pre-fetched, the caching proportion for this file is nearly linear with the total caching proportion.

Comparing the optimal $P$ in the two file settings, when the number of smaller size files is larger, these smaller size files will begin to be pre-fetched at a smaller total caching proportion. For example, in Fig. 3(c), files with the smallest file size of $2F$ begin to be pre-fetched when the total caching proportion reaches 0.2, while this value reduces to 0.15 in Fig. 3(d). On the other hand, when the number of larger files becomes smaller, these files will be cached in a higher proportion for the same total caching proportion. Usually the worst-case transmission happens when users request different largest files. Therefore, the higher proportion of the largest files are cached, the lower worst-case transmission rate can




(a) $V_{type} = [8, 4, 2], f_{num} = [60, 50, 10], K = 5$ (b) $V_{type} = [8, 4, 2], f_{num} = [40, 50, 90], K = 5$

(c) $V_{type} = [8, 4, 2], f_{num} = [60, 50, 10], K = 5$ (d) $V_{type} = [8, 4, 2], f_{num} = [40, 50, 90], K = 5$

Fig. 3. (a), (b) Worst-case transmission rate, and (c), (d) optimal caching proportion achieved by the proposed scheme vs. total caching proportion

be achieved. This also explains the fact that the worst-case transmission rate in Fig. 3(b) is smaller than that in Fig. 3(a).

## VI. Conclusion

We studied the decentralized coded caching scheme when files have different file sizes and shown that the caching proportion of each file is an important design parameter on the transmission rate over the shared link. To derive the optimal caching proportion, we formulated an optimization problem for the placement phase with a practical algorithm to solve it. Experiments have shown that the worst-case transmission rate using coded caching scheme with the derived caching proportion is smaller than the ones obtained by competitor algorithms. This brings some important benefits in the deployment of large scale data delivery systems, such as the video-on-demand application.

## References

[1] "Global mobile data traffic forecast update, 2015-2020," *Cisco Visual Networking Index (VNI) Forecast*, 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

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

[3] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, 2010.

[4] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug 2015.

[5] S. Wang, X. Tian, and H. Liu, "Exploiting the unexploited of coded caching for wireless content distribution," in *Proc. IEEE ICNC*, 2015.

[6] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *Proc. IEEE ICC*, 2015.

[7] J. Zhang, X. Lin, C. C. Wang, and X. Wang, "Coded caching for files with distinct file sizes," in *Proc. IEEE ISIT*, 2015.

[8] H. Cheng, C. Li, H. Xiong, and P. Frossard, "Optimal decentralized coded caching for heterogeneous files: Detailed derivation processes and theoretical proofs," *Thechnical Report*, 2017. [Online]. Available: http://min.sjtu.edu.cn/technicalReport/TR2017-1.pdf

ISBN 978-0-9928626-7-1 © EURASIP 2017