A Novel Centralized Strategy for Coded Caching with Non-uniform Demands

Pierre Quinton, Saeid Sahraei and Michael Gastpar
EPFL
IPG (IC)
CH-1015 Lausanne, Switzerland
Email: {pierre.quinton, saeid.sahraei, michael.gastpar}@epfl.ch

***Abstract*—Despite significant progress in the caching literature concerning the worst case and uniform average case regimes, the algorithms for caching with nonuniform demands are still at a basic stage and mostly rely on simple grouping and memory-sharing techniques. In this work we introduce a novel centralized caching strategy for caching with nonuniform file popularities. Our scheme allows for assigning more cache to the files which are more likely to be requested, while maintaining the same sub-packetization for all the files. As a result, in the delivery phase it is possible to perform linear codes across files with different popularities without resorting to zero-padding or concatenation techniques. We will describe our placement strategy for arbitrary range of parameters. The delivery phase will be outlined for a small example for which we are able to show a noticeable improvement over the state of the art.**

I. Introduction

Caching is a communication technique for redistributing the traffic in a broadcast network and thereby reducing its variability over time. The idea is to transfer part of the data to the users during low traffic periods. This data is stored at the caches of the users and helps as side information when later the server transfers the remaining data in a second phase. The central question in the caching literature is that for a given cache size, by how much one can reduce the traffic in this second (delivery) phase, assuming that in the first (placement) phase one only had partial or no knowledge at all of the requests of the users. There has been significant progress in answering this question under two paradigms. Firstly, when we look at the worst case delivery rate, meaning that we aim at minimizing the delivery rate for *any* request vector. Secondly, when we consider an average delivery rate under *uniform* distribution of the popularity of the files. For both of these scenarios the exact tradeoff between the size of the cache and the delivery rate has been characterized under uncoded placement [1], [2] , i.e., when in the placement phase users are not permitted to perform coding across several files.

By comparison, the question about minimizing the average delivery rate when the file popularities are non-uniform is still largely open. The main line of work [3]–[6] consists of partitioning the files into two or more groups, where each group contains files with similar popularity. Then one performs memory-sharing between these groups: each user divides his cache into several chunks, and assigns a chunk to each group of files. Naturally, if a group includes the more popular files a larger chunk of the cache (per file) will be allocated to them. Finally in the delivery phase each group is served individually, ignoring coding opportunities between files from different groups.

This simple scheme even when restricted to two groups has been proved to be order-optimal, meaning that it achieves a rate within a constant factor of an information theoretic converse bound. Nevertheless, the fact that coding opportunities between files from different groups are ignored should be viewed as an unfortunate technical obstacle rather than a natural extension of the strategies that exist for uniform caching. The dilemma is clear: assigning unequal amounts of cache to different groups and applying the centralized caching strategy in [1] for each group results in different sub-packetizations for files that belong to different groups. As a result, their sub-files will be of unequal size. It is therefore impossible to apply linear codes between different groups unless we resort to zero padding strategies or we concatenate the subfiles. Problems of the same nature - but perhaps less severe - appear if we resort to decentralized caching strategies [3], [5], [7].

Our main contribution in this paper is to propose a centralized caching strategy that bypasses this seemingly inevitable barrier. Specifically our placement strategy allows us to assign different amount of cache per file to different groups while maintaining equal sub-packetization for all the files. It is then very natural to allow for coding between files even if they do not belong to the same group. To the best of our knowledge this is the first centralized caching strategy that is specifically tailored for nonuniform file popularity. We will demonstrate the potential of this caching strategy by providing explicit delivery schemes for a small choice of the parameters and comparing its performance with the grouping strategies discussed earlier.

The rest of the paper is organized as follows. In Section II we will briefly describe the model. We will then move on to explaining our placement strategy in Section III. Next, in section IV we will describe our delivery strategy for a small choice of the parameters and compare its performance to the literature. Finally, we will conclude our work in Section V.

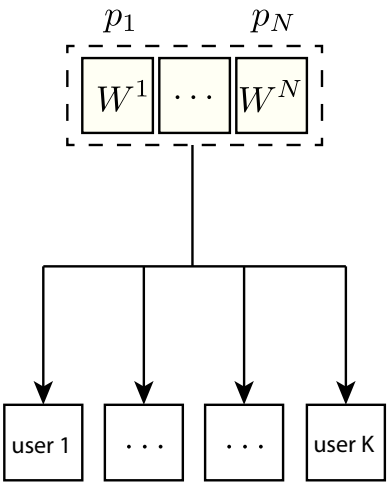

Fig. 1: An illustration of the caching network

II. Model Description

Our model and notation will be almost identical to the one described in [1]. We have a server which is in possession of N independent files $\{W^1, \ldots, W^N\}$ of equal size F and K users each equipped with a cache of size MF. The communication is done in two phases. In the placement phase, the server fills in the cache of each user without prior knowledge of their requests but with the knowledge of the popularity of the files. Next, each user requests precisely one file from the server. The request of each user is drawn independently from a distribution $p_{[1:N]}$ where p_i represents the probability of requesting file i. Note that this distribution does not vary across different users. We represent the set of requests by a vector d where $d_i \in [1:N]$ for all $i \in [1:K]$. In the delivery phase the server broadcasts a message of rate $R(d)$ to satisfy all the users simultaneously. See Figure 1 for an illustration. We deviate from the model in [1] in that we look at the expected delivery rate instead of the peak delivery rate. We say that a memory-rate pair (M, R) is achievable if and only if there exists a joint caching and delivery strategy with a cache of size MF such that for any request vector d a delivery message of rate $R(d)F$ satisfies all the users simultaneously, and

$$R = \sum_d \mathbb{P}(d)R(d) = \sum_{d\in[1:N]^K} \prod_{i=1}^{K} p_{d_i} R(d).$$

III. The Placement Phase of Strategy β

The placement phase of our strategy, which we refer to as strategy β, starts by partitioning the N files into L groups G_1,...,G_L of respective size N_1,...,N_L, such that $\sum_{i=1}^{L} N_i = N$. How to perform this partitioning is left as a design parameter but in general files within one partition should have similar probabilities of being requested. We represent by $g_i \in [1:L]$ the group to which the file W^i belongs. Accordingly, each user partitions his cache into L chunks of size M_1,...,M_L such that for any $\ell \in [1:L]$, we have $M_\ell \in \{0, N_\ell/K, 2N_\ell/K, \ldots, N_\ell\}$. It should be clear that this is only possible for discrete values of $M = \sum_{i=1}^{L} M_i$. The overall achievable memory-rate region will be the convex hull of all the discrete pairs (M, R) which can be served by our strategy. We define

$$r_\ell = KM_\ell/N_\ell \tag{1}$$

and assume without loss of generality that $r_1 \geq r_2 \geq \cdots \geq r_L$. Note that $r_{[1:L]}$ are integers.
Naturally, the following two identities hold.

$$\sum_{\ell=1}^{L} N_\ell r_\ell = MK \tag{2}$$

$$0 \leq r_\ell \leq K \qquad \forall \ell \in [1:L]. \tag{3}$$

Every file in the network regardless of which group they belong to is divided into S subfiles of equal size where

$$S = \binom{K}{K - r_1, r_1 - r_2, \ldots, r_{L-1} - r_L, r_L}.$$

The subfiles are indexed as follows

$$W^i_{\tau_1,\ldots,\tau_L} \quad \text{where } \tau_1 \subseteq [1:K], \quad \tau_j \subseteq \tau_{j-1} \text{ for } j \in 2,\ldots,L, \quad |\tau_i| = r_i \text{ for } i \in [1:L].$$

Note that there are precisely S such distinct indices.
For any (i,k) user k stores subfile $W^i_{\tau_1,\ldots,\tau_L}$ in his cache if and only if $k \in \tau_{g_i}$.

At this point it may help to illustrate this placement strategy via a simple example. Let us say that we have 3 users and 2 files and 2 groups such that each group contains exactly one file. Let us call the files $A = W^1$ and $B = W^2$ and assume that $r_1 = 2$ and $r_2 = 1$, so $M = \frac{r_1 N_1 + r_2 N_2}{K} = 1$. We must divide file A into 6 subfiles $A = \{A_{12,1}, A_{12,2}, A_{13,1}, A_{13,3}, A_{23,2}, A_{23,3}\}$. Same division applies to file B. The contents of the caches of the two users are illustrated in Table I.

user 1	user 2	user 3
$A_{12,1}$	$A_{12,1}$	$A_{13,1}$
$A_{12,2}$	$A_{12,2}$	$A_{13,3}$
$A_{13,1}$	$A_{23,2}$	$A_{23,2}$
$A_{13,3}$	$A_{23,3}$	$A_{23,3}$
$B_{12,1}$	$B_{12,2}$	$B_{13,3}$
$B_{13,1}$	$B_{23,2}$	$B_{23,3}$

TABLE I: Placement phase of strategy β for parameters $N = 2$, $K = 3$, $M = 1$, $r_1 = 2$, $r_2 = 1$

Let us now go back to the general placement strategy and calculate the amount of cache that user k dedicates to the ℓ'th group. By definition the index of the k'th user must be present in all the sets $\tau_1, \ldots, \tau_\ell$ whereas its index may or may not be present in the sets $\tau_{\ell+1}, \ldots, \tau_L$. We should divide the number

of such indices $\tau_1 \dots \tau_L$ by the total number of subfiles S to find the amount of cache dedicated to each file in group ℓ.

$$\begin{aligned} M_\ell &= \frac{N_\ell}{S}\binom{K-1}{r_1-1} \times \prod_{i=1}^{\ell-1}\binom{r_i-1}{r_{i+1}-1} \times \prod_{i=\ell}^{L-1}\binom{r_i}{r_{i+1}} \\ &= N_\ell \frac{\binom{K-1}{r_1-1}}{\binom{K}{r_1}} \times \prod_{i=1}^{\ell-1}\frac{\binom{r_i-1}{r_{i+1}-1}}{\binom{r_i}{r_{i+1}}} \\ &= N_\ell \frac{r_1}{K} \times \prod_{i=1}^{\ell-1}\frac{r_{i+1}}{r_i} \\ &= \frac{r_\ell N_\ell}{K}. \end{aligned}$$

Note that this expression matches with the way we defined the parameter r_ℓ in Equation (1).

IV. Delivery Strategy for $K = 3, N = 2$ and Comparison To the Literature

Let us start by describing our delivery strategy for the same toy example as in the previous section. The explicit delivery messages for all possible request vectors are provided in Table II.

request vector	delivery message	delivery rate
(A, A, A)	$A_{12,1} \oplus A_{13,1} \oplus A_{23,2}$ $A_{12,2} \oplus A_{13,3} \oplus A_{23,3}$	$1/3$
(A, A, B)	$B_{12,1} \oplus A_{23,2}$, $B_{13,1} \oplus A_{23,3}$ $B_{12,2} \oplus A_{13,1}$, $B_{23,2} \oplus A_{13,3}$	$2/3$
(A, B, B)	$B_{12,1} \oplus A_{23,2}$, $B_{13,1} \oplus A_{23,3}$ $B_{12,2} \oplus B_{13,3}$, $B_{23,2} \oplus B_{23,3}$	$2/3$
(B, B, B)	$B_{12,1} \oplus B_{12,2}$, $B_{12,1} \oplus B_{13,3}$ $B_{13,1} \oplus B_{23,2}$, $B_{13,1} \oplus B_{23,3}$	$2/3$

TABLE II: the set of delivery messages for $N = 2, K = 3$ and $r_1 = 2, r_2 = 1$ for all possible request vectors (different permutations are omitted.)

Let us say that file A is requested with probability p and file B with probability $1-p$. We assume without loss of generality that $p \geq 0.5$. The expected delivery rate is

$$R = \frac{1}{3}p^3 + \frac{2}{3}(1-p^3) = \frac{2}{3} - \frac{1}{3}p^3.$$

Alternatively we can set $(r_1, r_2) = (3, 0)$ which results in an expected delivery rate of $1 - p^3$. Therefore,

$$R_\beta = \min\{\frac{2}{3} - \frac{1}{3}p^3, 1 - p^3\}. \tag{4}$$

Therefore, the point $(M, R) = (1, \min\{\frac{2}{3} - \frac{1}{3}p^3, 1 - p^3\})$ is achievable with strategy β. We want to compare this with the achievable rate of grouping strategy in [3]. The strategy in [3] is particularly designed for decentralized caching, which by nature has an inferior performance (in terms of delivery rate) compared to its centralized counterpart. Thus, before we perform the comparison we slightly modify the strategy in [3] without compromising its basic concepts: the files are grouped in L disjoint sets and each user partitions his cache into L segments. Coding opportunities between several groups are ignored in the placement and delivery phase. However, instead of performing decentralized caching within each group we deploy the centralized caching strategy from [1], [2]. We refer to this as strategy α. It is easy to see that strategy α always outperforms the strategy in [3] in terms of expected delivery rate. It is also easy to see that strategy α always performs at least as good as the strategy in [1], [2] since by definition we can have only one partition which includes all the files. Let us now proceed to compare the two strategies α and β.

For the same choice of parameters $K = 3, N = 2, M = 1$, strategy α can be deployed with $L = 1$ or $L = 2$ groups. The former gives an expected rate of

$$\begin{aligned} R_{\alpha,L=1} &= \frac{1}{2}(p^3 + (1-p)^3) + \frac{2}{3}(1 - p^3 - (1-p)^3) \\ &= \frac{2}{3} - \frac{1}{6}(p^3 + (1-p)^3). \end{aligned}$$

If instead we set $L = 2$, we must divide the cache into two segments of sizes M_1 and $M_2 = 1 - M_1$. We will then ignore any coding opportunities between the files A and B, so the delivery rate is given by

$$\begin{aligned} R_{\alpha,L=2} &= [1 - M_1]p^3 + [1 - M_2](1-p)^3 \\ &+ [(1 - M_1) + (1 - M_2)](1 - p^3 - (1-p)^3) \\ &= 1 - M_1 p^3 - (1 - M_1)(1-p)^3. \end{aligned}$$

Assuming $p \geq \frac{1}{2}$ it is then profitable to set $M_1 = 1$ and we get a rate of

$$R_{\alpha,L=2} = 1 - p^3.$$

To summarize, we can write

$$R_\alpha = \min\{\frac{2}{3} - \frac{1}{6}(p^3 + (1-p)^3), 1 - p^3\}. \tag{5}$$

Comparing Equations (4) and (5) we see that strategy β strictly outperforms strategy α as long as $\frac{1}{2} < p < (1/2)^{\frac{1}{3}} \approx 0.794$. Let us summarize this in a table.

probability of file A	Expected Delivery Rate	
	strategy α	Strategy β
$0.5 \leq p \leq 0.739$	$\frac{2}{3} - \frac{1}{6}(p^3 + (1-p)^3)$	$\frac{2}{3} - \frac{1}{3}p^3$
$0.739 < p \leq 0.794$	$1 - p^3$	$\frac{2}{3} - \frac{1}{3}p^3$
$0.794 < p \leq 1$	$1 - p^3$	$1 - p^3$

TABLE III: Comparison of the expected delivery rate of strategies α and β when $K = 3, N = 2$ and $M = 1$. We assume that file A is requested with probability $p \geq 1/2$.

In Figure 2 we compare the delivery rates of the two strategies for $N = 2, K = 3, M = 1$. On the horizontal axis the probability of ordering file A increases from 0.5 to 1 and on the vertical axis we have the expected delivery rate. The maximum gain is offered over strategy α when $p = 0.738$ in which case $R_\beta \approx 0.89 R_\alpha$. A converse bound from [8] is plotted for comparison.

Similar analysis can be done for other cache sizes. In Table IV we summarize the achievable rate of strategy β for

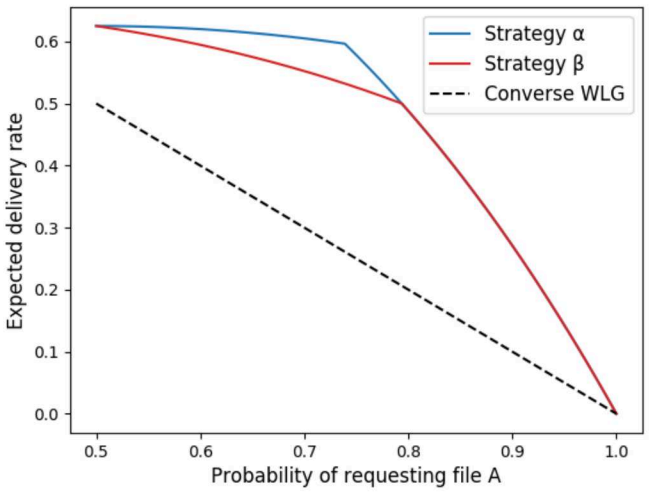

Fig. 2: Comparison of strategies α and β resumed in Table III together with the converse bound from [8].

cache size, M	(r_1, r_2)	Expected Delivery Rate
0	$(0,0)$	$2-p^3-(1-p)^3$
$\frac{1}{3}$	$(1,0)$	$\frac{5}{3}-p^3-\frac{2}{3}(1-p)^3$
$\frac{2}{3}$	$(1,1)$	$1-\frac{1}{3}p^3-\frac{1}{3}(1-p)^3$
1	$(2,1)$	$\frac{2}{3}-\frac{1}{3}p^3$
1	$(3,0)$	$1-p^3$
$\frac{4}{3}$	$(2,2)$	$\frac{1}{3}$
$\frac{4}{3}$	$(3,1)$	$\frac{2}{3}-\frac{2}{3}p^3$
$\frac{5}{3}$	$(3,2)$	$\frac{1}{3}-\frac{1}{3}p^3$
2	$(3,3)$	0

TABLE IV: The expected delivery rate of strategy β when $K=3, N=2$ for different values of (r_1, r_2) which results in different cache sizes M. We assume that file A is requested with probability $p \geq 1/2$.

difference choices of the parameters r_1 and r_2 which results in $M=(r_1+r_2)/K$.

The achievable memory-rate region for $K=3$, $N=2$ is the convex hull of all these points. Note that depending on the value of p some of these points may become irrelevant. For instance if $p=1$, the points achieved by setting $(r_1, r_2) = (2,2)$ does not lie on the boundary of the convex hull. In Figure 3 we have plotted the achievable memory-rate region for strategies α and β for $N=2, K=3$ and for $p=0.765$, where the improvements offered by strategy β are most visible. Again, the converse bound from [8] has been included for comparison. Note that the plot has been trimmed, since the performance is identical for very small or very large cache

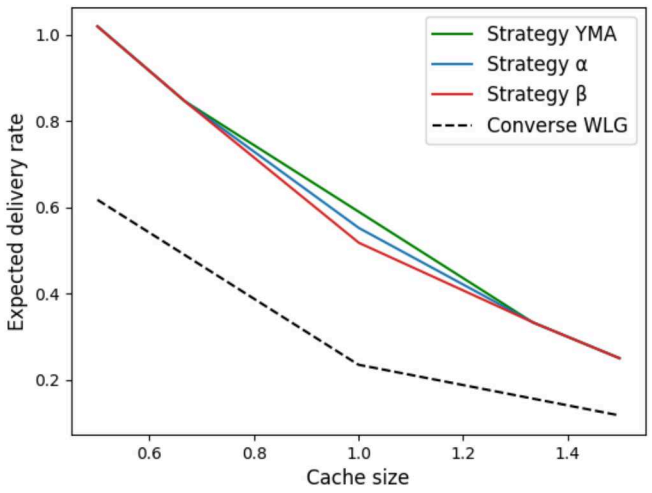

Fig. 3: Comparison of the expected delivery rate of strategies α and β and the strategy described in [2] (YMA) when $K=3, N=2$, together with the converse bound from [8]. We assume that file A is requested with probability $p=0.765$.

sizes. The gains are most visible in the vicinity of $M=1$.

V. Conclusion and Future Work

In this paper we presented a novel centralized caching strategy for non-uniform demands and demonstrated that for a small choice of parameters it outperforms the state of the art. For our future work, we intend to generalize our delivery strategy to arbitrary range of parameters. It is noteworthy that our strategy has the potential to be adapted to a user-specific popularity scenario, that is when the probability of requesting different files varies across the users. This can serve as another interesting direction for future research.

References

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

[2] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," in *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1613–1617.

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

[4] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," in *Information Theory and Applications Workshop (ITA), 2015*. IEEE, 2015, pp. 98–107.

[5] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "On the average performance of caching and coded multicasting with random demands," in *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*. IEEE, 2014, pp. 922–926.

[6] ——, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Transactions on Information Theory*, 2017.

[7] 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, 2015.

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