K Users Caching Two Files: An Improved Achievable Rate

Saeid Sahraei*, Michael Gastpar†
School of Computer and Communication Sciences, EPFL
Lausanne, Switzerland
Email: *saeid.sahraei@epfl.ch, †michael.gastpar@epfl.ch

***Abstract*—Caching is an approach to smoothen the variability of traffic over time. Recently it has been proved that the local memories at the users can be exploited for reducing the peak traffic in a much more efficient way than previously believed. In this work we improve upon the existing results and introduce a novel caching strategy that takes advantage of simultaneous coded placement and coded delivery in order to decrease the worst case achievable rate with 2 files and K users. We will show that for any cache size $\frac{1}{K} < M < 1$ our scheme outperforms the state of the art.**

***Index Terms*—Coded Caching, Content Delivery, Improved Achievable Rate**

The performance of content delivery services is highly dependent on the habits of the users and how well the servers model these habits and adapt their content distribution strategies to them. A basic observation of these habits is the temporal variability of the demands which in its simplest form can be formulated as high congestion during a particular time interval and low traffic for the rest of the day. One popular mechanism that the network can adapt to cope with this issue is caching: during the low traffic time interval, typically mornings, the servers store parts of the content in local memories of the users which may be helpful in the evenings, and hence reduce the peak traffic load. A notable challenge with this strategy is that typically the servers are not aware of which contents will be requested by the users in the peak time. Therefore, the caching of contents in local memories must be performed in such a way that regardless of what requests the users make, the contents are still helpful in reducing the traffic, as much as possible.

Perhaps the simplest solution to this problem is to partially store every file at the local caches of the users and transfer the rest of the data uncoded according to the demands made in the delivery time. In their seminal works [1], [2] Maddah-Ali and Niesen have proved that by using network coding techniques this simple strategy can be significantly outperformed if one allows coding across different files on the server and jointly optimizes the caching and the delivery strategies.

Despite its impressive potentials, the caching strategy introduced in [1] is known to perform poorly when the cache size is small, and in particular when the number of users is much larger than the number of files, $K \gg N$. The applicability of this paradigm in real world scenarios is manifold. A good example is when the files on the server vary widely in their popularity. It has been proved [3] that a nearly optimal caching strategy is to group files with similar popularities and ignore caching opportunities among files from different groups. The cache of each user is then divided into several segments, each segment dedicated to one such group. If the number of groups is large, then the cache size dedicated to each group, as well as the number of files within each group will be small. Another case appears when there are few popular television hits, say on Netflix, which are streamed by millions of users across the world.

In this work we take a step in improving the performance of caching strategies for small cache size M when $K \gg N$. This scenario has been studied before [4] where a new point $(M, R) = (\frac{1}{K}, \frac{K-1}{K}N)$ is shown to be achievable for arbitrary K and N that satisfy $K \geq N$. In this work we will find $K-1$ new points for the case of $N = 2$ and K arbitrary. One of these $K-1$ points, namely $(M, R) = (\frac{1}{K}, \frac{2(K-1)}{K})$ coincides with the point found in [4].

For the sake of brevity we skip the formal statement of the problem. It is precisely the setting described in [1]. The rest of the paper is organized as follows: In Section I we will demonstrate the ideas in our caching algorithm via a toy example. In Section II we will provide formal description of our placement and delivery strategies for the general case, that is when K is arbitrary and $M = \frac{m}{K}$ for some $m \in \{1, \dots, K-1\}$. In Section III we prove the correctness of our algorithm and find its achievable rate. Finally, in Section IV we compare the performance of our algorithm with the state of the art techniques introduced in [1], [4].

I. Example 1

We demonstrate our caching strategy via the simplest example which fully represents all the ideas involved in our algorithm. Assume we have $N = 2$ files A and B, of equal size and $K = 6$ users each with a cache of size $M = 0.5$ (normalized by file size). We break each file into $\binom{6}{3} = 20$ parts of equal size and index each part by a set of size three $\mathcal{T} = \{d_1, d_2, d_3\}$ where $1 \leq d_1 < d_2 < d_3 \leq 6$. User $\ell = 1, \dots, 6$ stores $A_{\mathcal{T}} \oplus B_{\mathcal{T}}$ in its cache if and only if $\ell \in \mathcal{T}$. In the following table we represent the content of the cache of each user:

User 1	User 2	User 3
$A_{123} \oplus B_{123}$ $A_{124} \oplus B_{124}$ $A_{125} \oplus B_{125}$ $\dots$ $A_{156} \oplus B_{156}$	$A_{123} \oplus B_{123}$ $A_{124} \oplus B_{124}$ $A_{125} \oplus B_{125}$ $\dots$ $A_{256} \oplus B_{256}$	$A_{123} \oplus B_{123}$ $A_{134} \oplus B_{134}$ $A_{135} \oplus B_{135}$ $\dots$ $A_{356} \oplus B_{356}$

User 4	User 5	User 6
$A_{124} \oplus B_{124}$ $A_{134} \oplus B_{134}$ $A_{145} \oplus B_{145}$ $\dots$ $A_{456} \oplus B_{456}$	$A_{125} \oplus B_{125}$ $A_{135} \oplus B_{135}$ $A_{145} \oplus B_{145}$ $\dots$ $A_{456} \oplus B_{456}$	$A_{126} \oplus B_{126}$ $A_{136} \oplus B_{136}$ $A_{146} \oplus B_{146}$ $\dots$ $A_{456} \oplus B_{456}$

Since the size of each subfile is $\frac{1}{20}$ and there are 10 subfiles stored at each cache, the total size of each cache is $M = 0.5$. Now assume users 1 and 2 ask for file A and users $3, 4, 5$ and 6 ask for B. In the delivery phase we start by transmitting subfiles of the form $A_{\mathcal{T}}$ or $B_{\mathcal{T}}$. For each such index $\mathcal{T}$ we decide whether to transmit $A_{\mathcal{T}}$ or $B_{\mathcal{T}}$ depending on how many digits of $\mathcal{T}$ are from $\{1, 2\}$ and how many are from $\{3, 4, 5, 6\}$. More precisely, we fix an integer $0 \leq j \leq KM + 1$ and for every such set $\mathcal{T}$ follow this rule:

If $|\mathcal{T} \bigcap \{3, 4, 5, 6\}| \geq j$ then transmit $A_{\mathcal{T}}$. Otherwise, transmit $B_{\mathcal{T}}$.

As we will show in Section III, there is an optimal (not necessarily unique) choice for this parameter j which in our case is 2. Therefore, we transmit:

$$\begin{aligned} |\mathcal{T} \bigcap \{3,4,5,6\}| &= 3: \quad A_{345}, A_{346}, A_{356}, A_{456}. \\ |\mathcal{T} \bigcap \{3,4,5,6\}| &= 2: \quad A_{134}, A_{135}, A_{136}, A_{145}, \\ & A_{146}, A_{156}, A_{234}, A_{235}, A_{236}, A_{245}, A_{246}, A_{256}. \\ |\mathcal{T} \bigcap \{3,4,5,6\}| &= 1: \quad B_{123}, B_{124}, B_{125}, B_{126}. \end{aligned}$$

At this stage, each user has access to every subfile he needs except:

User 3	B_{145}	B_{146}	B_{156}	B_{245}	B_{246}	B_{256}	B_{456}
User 4	B_{135}	B_{136}	B_{156}	B_{235}	B_{236}	B_{256}	B_{356}
User 5	B_{134}	B_{136}	B_{146}	B_{234}	B_{236}	B_{246}	B_{346}
User 6	B_{134}	B_{135}	B_{145}	B_{234}	B_{235}	B_{245}	B_{345}

The last stage of the algorithm is to help each user recover the remaining subfiles. We can transmit (a more formal way of accomplishing this is given below in Section II-B).

$$\begin{aligned} & B_{134} \oplus B_{135} \oplus B_{145} \quad , \quad B_{134} \oplus B_{136} \oplus B_{146} \\ & B_{135} \oplus B_{136} \oplus B_{156} \quad , \quad B_{145} \oplus B_{146} \oplus B_{156} \\ & B_{234} \oplus B_{235} \oplus B_{245} \quad , \quad B_{234} \oplus B_{236} \oplus B_{246} \\ & B_{235} \oplus B_{236} \oplus B_{256} \quad , \quad B_{245} \oplus B_{246} \oplus B_{256} \\ & B_{345} \oplus B_{346} \oplus B_{356} \oplus B_{456} \end{aligned}$$

which helps each user in $\{3, 4, 5, 6\}$ recover their desired subfiles. Nevertheless, an important observation here is that the fourth and the eighth messages in the chain above, that is $B_{145} \oplus B_{146} \oplus B_{156}$ and $B_{245} \oplus B_{246} \oplus B_{256}$ can already be constructed using the earlier transmissions and there is no need to separately transmit them:

$$B_{145} \oplus B_{146} \oplus B_{156} = (B_{134} \oplus B_{135} \oplus B_{145}) \oplus \\ (B_{134} \oplus B_{136} \oplus B_{146}) \oplus (B_{135} \oplus B_{136} \oplus B_{156})$$

and

$$B_{245} \oplus B_{246} \oplus B_{256} = (B_{234} \oplus B_{235} \oplus B_{245}) \oplus \\ (B_{234} \oplus B_{236} \oplus B_{246}) \oplus (B_{235} \oplus B_{236} \oplus B_{256}) .$$

Therefore, in total, we are transmitting 27 messages in the delivery phase which shows we are transmitting at rate $R = \frac{27}{20}$. The worst case achievable rate is obtained by considering all possible choices of different users over A and B. In our case, this happens precisely when two users ask for A and the other four ask for B (or vice versa) which is the scenario we studied. This proves that the point $(M, R) = (0.5, \frac{27}{20})$ is achievable when $K = 6$ and $N = 2$.

II. Formal Description of the Caching Algorithm

In this section we describe our caching algorithm for the general case. The setting is as follows: we have $N = 2$ files, which we name A and B. We have K users each with a cache of size $M = \frac{m}{K}$ for some integer $1 \leq m \leq K - 1$. Similar to [1] our caching strategy is comprised of two phases: the placement and the delivery phases. We now formally describe each phase.

A. Placement Strategy

Suppose $M = \frac{m}{K}$ for some integer $1 \leq m \leq K-1$. Partition each file into $\binom{K}{m}$ subfiles of equal size and index each subfile with a set of size m, i.e. $\mathcal{T} = \{d_1, \dots, d_m\}$ where $1 \leq d_1 < d_2 < \cdots < d_m \leq K$.
Store $A_{\mathcal{T}} \oplus B_{\mathcal{T}}$ at the cache of user ℓ if and only if $\ell \in \mathcal{T}$. This requires $\frac{\binom{K-1}{m-1}}{\binom{K}{m}} = \frac{m}{K}$ bits which is the size of the cache.

B. Delivery Strategy

Without loss of generality, assume the first L users ask for file A and the last $K - L$ users ask for B for some $L \in \{0, \dots, K\}$ (otherwise sort and re-label the users and the subfiles). If $L = K$ or $L = 0$ we transmit all $A_{\mathcal{T}}$ or $B_{\mathcal{T}}$ subfiles, respectively (therefore, the delivery rate is $R = 1$). From here on we assume $L \in \{1, \dots, K-1\}$. The delivery strategy is as follows: Fix an integer $0 \leq j \leq m + 1$. Then:

1) Transmit $A_{\mathcal{T} \bigcup \mathcal{S}}$ for all sets $\mathcal{T}$ and $\mathcal{S}$ such that $|\mathcal{T}| + |\mathcal{S}| = m$ and $|\mathcal{S}| \geq j$ and $\mathcal{T} \subseteq \{1, \dots, L\}$ and $\mathcal{S} \subseteq \{L+1, \dots, K\}$.
2) Transmit $B_{\mathcal{T} \bigcup \mathcal{S}}$ for all sets $\mathcal{T}$ and $\mathcal{S}$ such that $|\mathcal{T}| + |\mathcal{S}| = m$ and $|\mathcal{S}| < j$ and $\mathcal{T} \subseteq \{1, \dots, L\}$ and $\mathcal{S} \subseteq \{L+1, \dots, K\}$.
3) Transmit $\mathcal{M}_{\mathcal{T},\mathcal{S}} = A_{\mathcal{T} \bigcup \mathcal{S}} \oplus \sum_{t \in \mathcal{T}} A_{((\mathcal{T} \bigcup \{1\}) \setminus \{t\}) \bigcup \mathcal{S}}$ for all sets $\mathcal{S}$ and $\mathcal{T}$ such that $|\mathcal{S}| + |\mathcal{T}| = m$ and $|\mathcal{S}| < j$ and $\mathcal{T} \subseteq \{2, \dots, L\}$ and $\mathcal{S} \subseteq \{L+1, \dots, K\}$.
4) Transmit $\mathcal{N}_{\mathcal{T},\mathcal{S}} = B_{\mathcal{T} \bigcup \mathcal{S}} \oplus \sum_{s \in \mathcal{S}} B_{\mathcal{T} \bigcup ((\mathcal{S} \bigcup \{L+1\}) \setminus \{s\})}$ for all sets $\mathcal{S}$ and $\mathcal{T}$ such that $|\mathcal{S}| + |\mathcal{T}| = m$ and $|\mathcal{S}| \geq j$ and $\mathcal{T} \subseteq \{1, \dots, L\}$ and $\mathcal{S} \subseteq \{L+2, \dots, K\}$.

III. Analysis

A. Correctness

We will show that each user is capable of decoding his desired file based on his cache content and based on the messages sent in the delivery phase. Let us concentrate on user ℓ for some $\ell \in \{1, \dots, L\}$. The arguments are analogous for $\ell \in \{L+1, \dots, K\}$.

Based on the messages sent in step 1 of the delivery phase, user ℓ can decode all $A_{\mathcal{T}\bigcup\mathcal{S}}$ when $|\mathcal{S}| \geq j$. From the messages in step 2, user ℓ can decode $A_{\mathcal{T}\bigcup\mathcal{S}}$ when $|\mathcal{S}| < j$ and $\ell \in \mathcal{T}$. What are left to decode after these two phases are $A_{\mathcal{T}\bigcup\mathcal{S}}$ when $|\mathcal{S}| < j$ and $\ell \notin \mathcal{T}$. If $\ell = 1$, he can decode these messages from $\mathcal{M}_{\mathcal{T},\mathcal{S}} = A_{\mathcal{T}\bigcup\mathcal{S}} \oplus \sum_{t\in\mathcal{T}} A_{((\mathcal{T}\bigcup\{1\})\backslash\{t\})\bigcup\mathcal{S}}$ which are sent in step 3 of delivery. If $\ell \neq 1$ but $1 \in \mathcal{T}$, user ℓ can again decode $A_{\mathcal{T}\bigcup\mathcal{S}}$ from $\mathcal{M}_{\mathcal{T}',\mathcal{S}}$ sent in step 3 of delivery where $\mathcal{T}' = (\mathcal{T}\bigcup\{\ell\})\backslash\{1\}$. Assume now that $\ell \neq 1$ and $1 \notin \mathcal{T}$. User ℓ forms:

$$\begin{aligned} & \mathcal{M}_{\mathcal{T},\mathcal{S}} \oplus \sum_{t\in\mathcal{T}} M_{(\mathcal{T}\bigcup\{\ell\})\backslash\{t\},\mathcal{S}} \\ \overset{(a)}{=} \; & A_{\mathcal{T}\bigcup\mathcal{S}} \oplus \sum_{t\in\mathcal{T}} A_{((\mathcal{T}\bigcup\{\ell\})\backslash\{t\})\bigcup\mathcal{S}}. \end{aligned} \tag{1}$$

To establish (a), first note that each term of the form $A_{((\mathcal{T}\bigcup\{1\})\backslash\{t_1\})\bigcup\mathcal{S}}$, $t_1 \in \mathcal{T}$ appears exactly twice on the left hand side of the equation, once in $\mathcal{M}_{\mathcal{T},\mathcal{S}}$ and once in $M_{(\mathcal{T}\bigcup\{\ell\})\backslash\{t_1\},\mathcal{S}}$. Each term of the form $A_{((\mathcal{T}\bigcup\{1,\ell\})\backslash\{t_1,t_2\})\bigcup\mathcal{S}}$, $t_1, t_2 \in \mathcal{T}$, $t_1 \neq t_2$ also appears exactly twice, once in $M_{(\mathcal{T}\bigcup\{\ell\})\backslash\{t_1\},\mathcal{S}}$ and once in $M_{(\mathcal{T}\bigcup\{\ell\})\backslash\{t_2\},\mathcal{S}}$. On the other hand, each term of the form $A_{((\mathcal{T}\bigcup\{\ell\})\backslash\{t_1\})\bigcup\mathcal{S}}$, $t_1 \in \mathcal{T}$ appears exactly once in $M_{(\mathcal{T}\bigcup\{\ell\})\backslash\{t_1\},\mathcal{S}}$. Finally, the term $A_{\mathcal{T}\bigcup\mathcal{S}}$ also appears exactly once in $\mathcal{M}_{\mathcal{T},\mathcal{S}}$. From Equation (1) user ℓ can recover $A_{\mathcal{T}\bigcup\mathcal{S}}$ since he knows every other term in the summation.

B. Achievable Rate

We count the total number of messages sent in the delivery phase and multiply this by the size of each message that is $\frac{1}{\binom{K}{m}}$.

First note that each index appears exactly once in the first two steps of delivery. Therefore, the number of messages sent in these two steps is $\binom{K}{m}$.

The number of messages sent in step 3 of delivery is

$$\begin{aligned} \#\text{msgs}^3 &= \sum_{i=0}^{j-1} \#\text{msgs}(|S| = i) \\ &= \sum_{i=\max(0,m-L+1)}^{\min(j-1,K-L)} \binom{K-L}{i}\binom{L-1}{m-i}. \end{aligned}$$

Similarly, the number of messages sent in step 4 of delivery is

$$\#\text{msgs}^4 = \sum_{i=\max(j,m-L)}^{\min(m,K-L-1)} \binom{K-L-1}{i}\binom{L}{m-i}.$$

Therefore, the total number of messages sent in the delivery phase multiplied by message size is:

$$\begin{aligned} R_K(M,L,j) &= 1 + \frac{\sum_{i=\max(0,m-L+1)}^{\min(j-1,K-L)} \binom{K-L}{i}\binom{L-1}{m-i}}{\binom{K}{m}} \\ &+ \frac{\sum_{i=\max(j,m-L)}^{\min(m,K-L-1)} \binom{K-L-1}{i}\binom{L}{m-i}}{\binom{K}{m}}. \end{aligned}$$

We make the following observation:

Proposition 1. *There exists a solution to* $j^* = \arg\min_j R_K(M,L,j)$ *that satisfies* $j^* = \left\lceil m(1-\frac{L}{K}) \right\rceil$.

Proof: First note that we can restrict j^* to $\max(m-L+1,0) \leq j^* \leq \min(K-L, m+1)$ since if $j^* < \max(m-L+1,0)$ then $R_K(M,L,j^*) \geq R_K(M,L,\max(m-L+1,0))$ and if $j^* > \min(K-L,m+1)$ then we have $R_K(M,L,j^*) \geq R_K(M,L,\min(K-L,m+1))$.

Next we prove that $j^* \geq m(1-\frac{L}{K})$. If $j^* = \min(K-L, m+1)$ then the inequality is trivial. Assume $j^* < \min(K-L, m+1)$. Since j^* is optimal, we have $R_K(M,L,j^*+1) \geq R_K(M,L,j^*)$. It follows that:

$$\begin{aligned} & R_K(M,L,j^*+1) - R_K(M,L,j^*) \geq 0 \\ \Rightarrow \; & \binom{K-L}{j^*}\binom{L-1}{m-j^*} \geq \binom{K-L-1}{j^*}\binom{L}{m-j^*} \\ \Rightarrow \; & \frac{K-L}{K-L-j^*} - \frac{L}{L-m+j^*} \geq 0 \\ \Rightarrow \; & j^* \geq m(1-\frac{L}{K}). \end{aligned}$$

Finally, we show that $j^* \leq m(1-\frac{L}{K})+1$. If $j^* = \max(m-L+1,0)$ then the inequality is trivial. Assume that $j^* > \max(m-L+1,0)$. Then from optimality of j^* it follows that (similar to the previous case) $R_K(M,L,j^*-1) \geq R_K(M,L,j^*) \Rightarrow j^*-1 \leq m(1-\frac{L}{K})$. Putting these two inequalities together, we obtain $j^* = \left\lceil m(1-\frac{L}{K}) \right\rceil$.

■

We can now define

$$\begin{aligned} R_K(M,L) &= 1 + \frac{\sum_{i=\max(0,m-L+1)}^{j^*-1} \binom{K-L}{i}\binom{L-1}{m-i}}{\binom{K}{m}} \\ &+ \frac{\sum_{i=j^*}^{\min(m,K-L-1)} \binom{K-L-1}{i}\binom{L}{m-i}}{\binom{K}{m}} \end{aligned} \tag{2}$$

with $m = MK$ and $j^* = \left\lceil m(1-\frac{L}{K}) \right\rceil$. The achievable rate is the maximum of $R_K(M,L)$ over all possible $1 < L < K$:

$$R_K(M) = \max_{0<L<K} R_K(M,L). \tag{3}$$

IV. Comparison with the State of the Art

In this section we perform a comparison between the achievable rate of our scheme and that of [1], [4]. The achievable rate of our scheme for $K = 10$ and $N = 2$ is plotted in red in Figure 1 and is found via equation (3) for every $M \in \{\frac{1}{K}, \dots, \frac{K-1}{K}\}$. We again emphasize that the leftmost point of our curve, here $(\frac{1}{10}, \frac{9}{5})$ has been previously found in [4]. The achievable rate via Maddah-Ali–Niesen caching

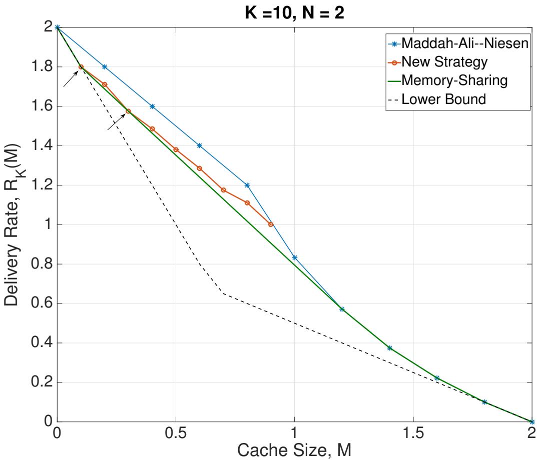

Fig. 1. Comparison of the achievable rate of our caching strategy with that of Maddah-Ali–Niesen for $K = 10$, $N = 2$.

strategy has been plotted in blue. Evidently from the plot, and as has been proved in the following proposition, our scheme outperforms that of [1] for every M at which both rates are defined, that is: $M \in \{\frac{2}{K}, \frac{4}{K} \ldots, \frac{2(\lceil\frac{K}{2}\rceil-1)}{K}\}$ and for every K. The proof is in Appendix A.

Proposition 2. *Let $R_K(M)$ be our achievable rate as defined in* (3)*. Let $\hat{R}_K(M)$ be the achievable rate from [1]. Then we have:*

$$\begin{aligned} R_K(M) \quad &\leq \quad \hat{R}_K(M)\,, \\ &\qquad \forall K,\ \forall M \in \{\frac{2}{K}, \frac{4}{K}, \ldots, \frac{2(\lceil\frac{K}{2}\rceil-1)}{K}\}. \end{aligned}$$

The inequality is strict, except when both K is odd and $M = \frac{K-1}{K}$.

For the sake of completeness we have plotted a lower bound on the achievable rate (the dotted black line). Since [1], there has been several works to improve this bound [5], [6], [7]. The bound that is plotted here corresponds to the work in [6] (which in this case coincides with the lower bound given in [1]).

A. Memory Sharing and Minimum Size of the Files

In Figure 1 we have also plotted the memory sharing region (green curve). The two points marked by arrows contribute to this region. The first point (found in [4]) is $(\frac{1}{10}, \frac{9}{5})$ and the second point is $(\frac{3}{10}, \frac{63}{40})$ The leftmost point from [1] that contributes to the memory sharing region is $(\frac{12}{10}, \frac{4}{7})$. As K grows large there will be more points $0 < M < 1$ contributing to the memory-sharing region. For instance when $K = 16$, there are three, namely $(\frac{1}{16}, \frac{15}{8})$, $(\frac{3}{16}, \frac{97}{56})$, and $(\frac{5}{16}, \frac{331}{208})$ and at $K = 23$ there are 4 new points. However this dependency is not monotonic in K.

It is noteworthy that when the file size is small, the other points found by our scheme which lie within the memory sharing region are still relevant. Recall first that our scheme requires each file A and B to be of size at least $\binom{10}{m}$ for any particular memory size $M = \frac{m}{K}, m \in \{1, \ldots, K-1\}$. On the other hand, since the memory sharing strategy interpolates between the points $M = \frac{3}{10}$ and $M = \frac{12}{10}$, the minimum file size for $m \in \{4, \ldots, 9\}$ must be (see Appendix B)

$$\begin{aligned} F_{\min} \quad &= \quad \binom{10}{3} \frac{7(12-m)}{gcd\{7(12-m), m-3\}} \\ &\quad + \quad \binom{10}{6} \frac{m-3}{gcd\{7(12-m), m-3\}} \end{aligned} \tag{4}$$

which is strictly larger than $\binom{10}{m}$ for any $m \in \{4, \ldots, 9\}$. The difference becomes particularly visible, for instance when $m = 9$ where memory sharing requires a file size about 100 times larger than directly applying our caching strategy for $m = 9$.

V. Conclusion

The small cache paradigm with much larger number of users than files has not received as much attention in the literature as it deserves. In this work we took a step in improving the achievable rate of this regime by introducing a novel caching strategy for arbitrary number of users and 2 files. Our algorithm takes advantage of simultaneous coded placement and coded delivery to improve upon the achievable rate of [1] when the cache is smaller than the size of one file. Future work will explore the possibility of generalizing our caching algorithm to more than two files.

Appendix A

Proof of Proposition 2:

From [1]:

$$\begin{aligned} \hat{R}_K(M) \quad &= \quad K(1-\frac{M}{N})\min\{\frac{1}{1+\frac{MK}{N}}, \frac{N}{K}\} \\ &= \quad \begin{cases} 1 & \text{if } K \text{ is odd and } M = \frac{K-1}{K}, \\ 2-M & \text{Otherwise.} \end{cases} \end{aligned}$$

If K is odd and $M = \frac{K-1}{K}$, then we have

$$R_K(M, L) = R_K(M, L, j = K-L) = 1 = \hat{R}_K(M).$$

We will show that if K is even or $M < \frac{K-1}{K}$ then $R_K(M) < \hat{R}_K(M)$. We consider three cases. First assume $L \geq K-m$. Then we have:

$$\begin{aligned} R_K(M, L) \quad &\leq \quad R_K(M, L, j = K-L) \\ &= \quad 1 + \frac{\sum_{i=\max(0,m-L+1)}^{K-L-1} \binom{K-L}{i}\binom{L-1}{m-i}}{\binom{K}{m}} \\ &< \quad 1 + \frac{\binom{K-1}{m}}{\binom{K}{m}} = 2 - \frac{m}{K} = \hat{R}_K(M). \end{aligned}$$

Next, assume $L < K-m$ and $L < m+1$. Then:

$$\begin{aligned} R_K(M, L) \quad &\leq \quad R_K(M, L, j = m-L+1) \\ &= \quad 1 + \frac{\sum_{i=m-L+1}^{m} \binom{K-L-1}{i}\binom{L}{m-i}}{\binom{K}{m}} \\ &< \quad 1 + \frac{\binom{K-1}{m}}{\binom{K}{m}} = 2 - \frac{m}{K} = \hat{R}_K(M). \end{aligned}$$

Finally, assume $L < K - m$ and $L \geq m + 1$. Then:

$$\begin{aligned} & R_K(M, L) \leq R_K(M, L, j = m) \\ = \quad & 1 + \frac{\sum_{i=0}^{m-1} \binom{K-L}{i}\binom{L-1}{m-i} + \binom{K-L-1}{m}\binom{L}{0}}{\binom{K}{m}} \\ = \quad & 1 + \frac{\binom{K-1}{m} - \binom{K-L}{m} + \binom{K-L-1}{m}}{\binom{K}{m}} \\ < \quad & 2 - \frac{m}{K} = \hat{R}_K(M). \end{aligned}$$

■

Appendix B

Proof of Equation (4):

First note that the point $M = \frac{3}{10}$ requires $\binom{10}{3}$ bits and the points $M = \frac{12}{10}$ needs $\binom{10}{6}$ bits (from [1]). In order to perform memory sharing for a point $M = \frac{m}{K}$ for some $m \in \{4, \ldots, 9\}$ we need to first divide file A into two subfiles $A^{(1)}$ and $A^{(2)}$ (same for B). We also break the cache into $M^{(1)}$ and $M^{(2)}$ such that $size(M^{(1)}) = \frac{12-m}{m-3} size(M^{(2)})$. Due to the particular caching strategies used at the two end points, we have $size(M^{(1)}) = \frac{3}{10} size(A^{(1)})$ and $size(M^{(2)}) = \frac{12}{10} size(A^{(2)})$. Therefore,

$$\begin{aligned} & \frac{3}{10} size(A^{(1)}) = \frac{12}{10} size(A^{(2)}) \frac{12-m}{m-3} \\ \Rightarrow \quad & size(A^{(1)}) = \frac{4(12-m)}{m-3} size(A^{(2)}). \end{aligned}$$

But $size(A^{(1)})$ must of the form $\binom{10}{3}\ell$ for some integer ℓ. Similarly, $size(A^{(2)}) = \binom{10}{6}\ell'$ for some integer ℓ'. Thus:

$$\begin{aligned} \binom{10}{3}\ell \quad &= \quad \frac{4(12-m)}{m-3}\binom{10}{6}\ell' \\ \Rightarrow \quad & \frac{\ell}{\ell'} = \frac{7(12-m)}{m-3}. \end{aligned}$$

To choose the smallest ℓ and ℓ' we have $\ell = \frac{7(12-m)}{gcd\{m-3, 7(12-m)\}}$ and $\ell' = \frac{m-3}{gcd\{m-3, 7(12-m)\}}$. The claim follows since $F_{\min} = size(A^{(1)}) + size(A^{(2)})$.

Acknowledgement

This work was supported in part by the European ERC Starting Grant 259530-ComCom.

References

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

[2] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*. IEEE, 2013, pp. 421–427.

[3] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 2014, pp. 221–226.

[4] Z. Chen, P. Fan, and K. Ben Letaief, "Fundamental limits of caching: Improved bounds for small buffer users," *arXiv preprint arXiv:1407.1935*, 2014.

[5] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," *arXiv preprint arXiv:1501.06003*, 2015.

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

[7] C. Tian, "A note on the fundamental limits of coded caching," *arXiv preprint arXiv:1503.00010*, 2015.