On Coded Caching Systems with Offline Users

Yinbin Ma and Daniela Tuninetti
University of Illinois Chicago, Chicago, IL 60607, USA
Email:{yma52, danielat}@uic.edu

***Abstract*—Coded caching is a technique that leverages locally cached contents at the users to reduce the network's peak-time communication load. Coded caching achieves significant performance gains compared to uncoded caching schemes and is thus a promising technique to boost performance in future networks. In the original model introduced by Maddah-Ali and Niesen (MAN), a server stores multiple files and is connected to multiple cache-aided users through an error-free shared link; once the local caches have been filled and *all* users have sent their demand to the server, the server can start sending coded multicast messages to satisfy all users' demands. A practical limitation of the original MAN model is that it halts if the server does not receive all users' demands, which is the limiting case of asynchronous coded caching when the requests of some users arrive with infinite delay. This paper formally defines a coded caching system where some users are offline. Achievable and converse bounds are proposed for this novel setting and shown to meet under certain conditions; otherwise, they are within a constant multiplicative gap of two. Interestingly, when optimality can be be shown, the optimal load-memory tradeoff only depends on the number active users, and not on the total (active plus offline) number of users.**

***Index Terms*—Coded caching with offline users; Achievablity; Optimality for small memory size; Multiplicative constant gap.**

I. Introduction

Coded caching, first introduced by Maddah-Ali and Niesen (MAN) in [1], leverages locally cached contents at the users to reduce the communication load during peak-traffic times. A coded caching system has two phases. During the cache placement phase, the server populates the users' local caches, without knowing the users' future demands. During the delivery phase, the server broadcasts coded multicast messages to satisfy the users' demands. The achievable scheme proposed in [1] (referred to as MAN in the following) has a combinatorial *uncoded cache placement* phase[1] and a network coded delivery phase. In [2], an improved delivery was proposed (referred to as YMA in the following), which improves on the MAN delivery by removing those linearly dependent multicast messages that may occur when a file is requested by multiple users. The MAN placement with the YMA delivery match the converse bound derived in [3] under the constraint of uncoded placement; otherwise it is optimal within a factor of two [4].

Coded placement strictly improves performance compared to uncoded placement, and can be exactly optimal. A non-exhaustive list of related works is as follows: [5] showed how to achieve the cut-set bound in the small memory regime when there are more users than files; [6] shows an improved performance compared to [5] in the same regime; [7] derived the optimal performance for the case of two users (and any number of files), and a partial characterization for the case of two files (and any number of users).

A limitation of the classical coded caching setting [1] is that all users present during the placement phase must be active and synchronously send their demand before the delivery phase starts. The "asynchronous demands" setting, already discussed in [1], allows the server to start transmission as soon as the first demand arrives; known schemes (as in [8], and references therein) however assume that all demands eventually arrive in finite time, otherwise the system fails to complete the delivery or the delivery time is infinite.

The case of coded caching with offline users is the focus of this paper. Performance with offline users was investigated in [9] for the shared-link model (where a scheme is proposed under the assumption that the number of active users is fixed), and in [10] for the Device-to-Device model (which allows for an "outage event" when too many users are active). Here we assume that the demands of the offline users never arrive (or arrive with infinite delay) and the demands of the remaining users arrive synchronously. We refer to this setting as "hotplug" coded caching[2]. The "decentralized" coded caching setting, already discussed in [1], allows each user to cache from the server at random and independently of the other users. This type of placement gives an achievable load for our hotplug setting because the decentralized scheme works for any number of user demands. In addition, the performance of the decentralized setting is useful to derive constant multiplicative gap results [2], [11].

Contributions: We first formalize the hotplug coded caching problem. We propose two schemes that allow the demands of the active users to be satisfied regardless of the set of offline users, where the number (but not the identity) of the offline users is assumed known at the time of placement, similarly to [9][3]. Our schemes use coded cache placement.

1) Our first new achievable scheme exploits Maximum Distance Separable (MDS) codes in the placement phase, where coding is done within each file but not across files. We show that such a strategy reduces the load significantly in the small cache size regime compared to a centralized baseline schemes. Furthermore, it achieves the optimal performance when the memory is small and

[1]Uncoded cache placement means that bits of the files are directly copied into the caches without coding.

[2]Hotplug is a computer system term that refers to a device that can be added or removed from the running system without having to restart the system.

[3]The server can serve users in multiple rounds if the actual number of active users is greater than what the server assumed in the placement.

the number of files is large. The matching converse is obtained from [4].

2) Our second new achievable scheme applies MDS-like coding to the coded placement of [5]. This scheme achieves the optimal performance when the memory is small and there are less files than users. The matching converse is obtained from the cut-set bound in [1].

Paper Outline: The rest of the paper is organized as follows. Section II states the problem formulation and summarizes related known results. Section III summarizes our main results. Section IV shows the optimal scheme when there are two files and two active users. Section V provides some numerical examples. Section VI concludes the paper. Some proofs can be found in the longer version of this paper [12].

II. Problem Formulation and Known Results

A. Notation Convention

We adopt the following notation convention.

- Calligraphic symbols denote sets, bold lowercase symbols vectors, bold uppercase symbols matrices, and sans-serif symbols system parameters.
- $\mathbf{M}[\mathcal{Q}]$ denotes the submatrix of $\mathbf{M}$ obtained by selecting the rows indexed by $\mathcal{Q}$. Similarly, $\mathbf{d}[\mathcal{I}]$ is the subvector of $\mathbf{d}$ obtained by selecting the elements indexed by $\mathcal{I}$.
- For an integer b, we let $[b] := \{1, \dots, b\}$.
- For sets $\mathcal{S}$ and $\mathcal{Q}$, we let $\mathcal{S} \setminus \mathcal{Q} := \{k : k \in \mathcal{S}, k \notin \mathcal{Q}\}$.
- For a vector $\mathbf{d}$, $\mathsf{rank}(\mathbf{d})$ returns the number of distinct elements in $\mathbf{d}$. For example, $\mathsf{rank}([1,5,5,1]) = 2$.
- For a ground set $\mathcal{G}$ and an integer t, we let $\Omega^t_{\mathcal{G}} := \{\mathcal{T} \subseteq \mathcal{G} : |\mathcal{T}| = t\}$.
- For $\mathcal{T} \in \Omega^t_{\mathcal{G}}$, we let $\mathcal{T}_i$ be the i-th subset in $\Omega^t_{\mathcal{G}}$ in lexicographical order. For example, the sets in $\Omega^2_{\{1,2,3\}}$ are indexed as $\mathcal{T}_1 = \{1,2\}$, $\mathcal{T}_2 = \{1,3\}$, $\mathcal{T}_3 = \{2,3\}$.
- For integers a and b, $\binom{a}{b}$ is the binomial coefficient, or zero if $a \geq b \geq 0$ does not hold.

B. Problem Formulation

In a $(\mathsf{K}, \mathsf{K}', \mathsf{N})$ hotplug coded caching system:

- A central server stores N files, denoted as $F_1, \cdots, F_{\mathsf{N}}$.
- Each file has B i.i.d. uniformly distributed bits.
- The server communicates with K users through a error-free shared link.
- Each user has a local memory that can contain up to MB bits, where $\mathsf{M} \in [0, \mathsf{N}]$. We refer to M as the *memory size*. Caches are denoted as $Z_1, \dots, Z_{\mathsf{K}}$.
- The server sends the signal X to the users through the shared link, where X has no more than RB bits, with $\mathsf{R} \geq 0$. We refer to R as the *load*.
- The system has a placement phase and a delivery phase. The placement phase occurs at a time when the server is still unaware of which users will be offline, and which files the active users will request. We assume that the server knows that K' users will be active, with $\mathsf{K}' \leq \mathsf{K}$. The delivery phase occurs after the active users have sent their demands to the server.

In particular:

Placement Phase: The server populates the local caches as a function of the files it stores, i.e.,

$$H(Z_k | F_1, \dots F_{\mathsf{N}}) = 0, \quad \forall k \in [\mathsf{K}]. \tag{1}$$

Delivery Phase: Once the set of active users becomes known to the server, denoted by $\mathcal{I} \in \Omega^{\mathsf{K}'}_{[\mathsf{K}]}$, as well as the demand $d_k \in [\mathsf{N}]$ of user $k \in \mathcal{I}$, the server starts sending. We denote the demands of all users by the vector $\mathbf{d} = [d_1, \dots d_{\mathsf{K}}]$, thereby also including the demands of the offline users; the server is thus aware of the pair $(\mathcal{I}, \mathbf{d}[\mathcal{I}])$. The message X sent by the sever must guarantee that each active user, with the help of its locally cached content, can revere its desired file, i.e., for every $\mathcal{I} \in \Omega^{\mathsf{K}'}_{[\mathsf{K}]}$ and $\mathbf{d}[\mathcal{I}] \in [\mathsf{N}]^{\mathsf{K}'}$, we must have

$$H(X | \mathcal{I}, \mathbf{d}[\mathcal{I}], F_1, \dots F_{\mathsf{N}}) = 0, \tag{2}$$

$$H(F_{d_k} | Z_k, X) = 0, \quad \forall k \in \mathcal{I}. \tag{3}$$

Performance: For $\mathsf{M} \in [0, \mathsf{N}]$, we denote by $\mathsf{R}^\star(\mathsf{M})$ the minimum worst-case load, defined as

$$\mathsf{R}^\star(\mathsf{M}) = \limsup_{\mathsf{B} \to \infty} \min_{X, Z_1, \dots Z_{\mathsf{K}}} \max_{\mathcal{I}, \mathbf{d}[\mathcal{I}]} \{\mathsf{R} : \text{conditions in (1)-(3) are satisfied with memory size } \mathsf{M}\}. \tag{4}$$

C. Known Results for $\mathsf{K}' = \mathsf{K}$

When $\mathsf{K}' = \mathsf{K}$, the hotplug model is equivalent to the classical setting in [1] for which the following is known.

MAN Placement Phase: Fix $t \in [0 : \mathsf{K}]$ and partition each file into $\binom{\mathsf{K}}{t}$ equal-size subfiles as

$$F_i = (F_{i,\mathcal{W}} : \mathcal{W} \in \Omega^t_{[\mathsf{K}]}), \quad \forall i \in [\mathsf{N}]. \tag{5}$$

For each user $k \in [\mathsf{K}]$, the cache content Z_k is

$$Z_k = (F_{i,\mathcal{W}} : i \in [\mathsf{N}], \mathcal{W} \in \Omega^t_{[\mathsf{K}]}, k \in \mathcal{W}), \quad \forall k \in [\mathsf{K}]. \tag{6}$$

The memory size is $\mathsf{M} = \mathsf{N}\binom{\mathsf{K}-1}{t-1} / \binom{\mathsf{K}}{t} = \mathsf{N}t/\mathsf{K}$. The MAN placement is referred to a *centralized* as it requires coordination among users during the placement phase.

MAN Multicast Messages: For the demand vector $\mathbf{d} \in [\mathsf{N}]^{\mathsf{K}}$, the server constructs the multicast messages

$$X_{\mathcal{S}} = \sum_{k \in \mathcal{S}} F_{d_k, \mathcal{S} \setminus \{k\}}, \quad \forall \mathcal{S} \in \Omega^{t+1}_{[\mathsf{K}]}. \tag{7}$$

Notice that user $k \in \mathcal{S}$ can recover the missing subfile $F_{d_k, \mathcal{S}\setminus\{k\}}$ from $X_{\mathcal{S}}$ in (7) by "caching out" $\sum_{u \in \mathcal{S}\setminus\{k\}} F_{d_u, \mathcal{S}\setminus u}$ which can be computed from Z_k in (6).

YMA Delivery Phase: Some multicast messages in (7) may be linearly dependent on the others when a file is requested by multiple users [2]. By not sending the redundant multicast messages, the lower convex envelope of the following points for all $t \in [0 : \mathsf{K}]$ is achievable

$$(\mathsf{M}_t, \mathsf{R}^{\text{cen}}_t) = \left(\mathsf{N} \frac{\binom{\mathsf{K}-1}{t-1}}{\binom{\mathsf{K}}{t}}, \frac{\binom{\mathsf{K}}{t+1} - \binom{\mathsf{K}-r}{t+1}}{\binom{\mathsf{K}}{t}} \right) \Bigg|_{r = \min\{\mathsf{N}, \mathsf{K}\}}. \tag{8}$$

Remark 1 (Centralized vs. Decentralized)**.** Decentralized placement [2], [11] refers to the case where users cache each bit of the library i.i.d. at random with probability $\mu :=$

$\mathsf{M}/\mathsf{N} \in [0,1]$. An achievable memory-load tradeoff with such a decentralized placement is $\mathsf{R}^{\text{de-cen}}$ given by

$$\mathsf{R}^{\text{de-cen}} = \frac{1-\mu}{\mu}\Big(1-(1-\mu)^r\Big)\Big|_{\substack{\mu:=\mathsf{M}/\mathsf{N}\\ r:=\min\{\mathsf{N},\mathsf{K}\}}} \geq \mathsf{R}^{\text{cen}}, \quad (9)$$

where R^{cen} is the lower convex envelop of (8) and the inequality in (9) is from [4, eq(20)]. For fixed μ, R^{cen} depends on both K and $r = \min\{\mathsf{N},\mathsf{K}\}$, while $\mathsf{R}^{\text{de-cen}}$ only on r. □

III. Main Results

In this section we summarize our main results, which will be proved in the following sections.

A. Achievability

Theorem 1 (Achievability)**.** *Let* $\mathsf{r}' := \min\{\mathsf{N},\mathsf{K}'\}$. *For a* $(\mathsf{K},\mathsf{K}',\mathsf{N})$ *hotplug system, the lower convex envelope of the following point is achievable*

$$(\mathsf{M}_t,\mathsf{R}_t^{\text{base}}) = \left(\mathsf{N}\frac{\binom{\mathsf{K}-1}{t-1}}{\binom{\mathsf{K}}{t}}, \frac{\binom{\mathsf{K}}{t+1}-\binom{\mathsf{K}-\mathsf{r}'}{t+1}}{\binom{\mathsf{K}}{t}}\right), \forall t\in[0:\mathsf{K}], \quad (10)$$

$$(\mathsf{M}_t^{\text{new1}},\mathsf{R}_t^{\text{new1}}) = \left(\mathsf{N}\frac{\binom{\mathsf{K}-1}{t-1}}{\binom{\mathsf{K}'}{t}}, \frac{\binom{\mathsf{K}'}{t+1}-\binom{\mathsf{K}'-\mathsf{r}'}{t+1}}{\binom{\mathsf{K}'}{t}}\right), \forall t\in[0:\mathsf{K}']. \quad (11)$$

When $\mathsf{K}\geq\mathsf{K}'\geq\mathsf{N}$*, the following is achievable*

$$(\mathsf{M}^{\text{new2}},\mathsf{R}^{\text{new2}}) = \left(\frac{1}{\mathsf{K}'},\mathsf{N}\left(1-\frac{1}{\mathsf{K}'}\right)\right). \quad (12)$$

Few comments are in order.

Baseline Scheme: The performance of the baseline scheme in (10) is that of a classical coded caching system with K users and N files but with a restricted set of demand vectors $\mathbf{d}$: $\mathsf{rank}(\mathbf{d}) \in [\min\{\mathsf{N},\mathsf{K}'\}]$, that is, the largest number of distinct files that can be requested is $\mathsf{r}' = \min\{\mathsf{N},\mathsf{K}'\}$ (i.e., the minimum between the number of files and the number of *active* users) rather than $\min\{\mathsf{N},\mathsf{K}\}$ (as in (8)). Here the server "fills in" the demand of the offline users by repeating in a predefined order the demands of the active users[4] and uses the YMA delivery for the "filled in" demand vector; with this, the number of distinct files that must be delivered by the server is not increased compared to that in the hotplug system.

New Schemes: Our first novel scheme attains the load in (11)–the proof can be found in [12, Appendix B]. At a high level, we split each file into $\binom{\mathsf{K}'}{t}$ equal-length subfiles and then code the subfiles with an MDS code of rate $\binom{\mathsf{K}'}{t}/\binom{\mathsf{K}}{t}$. The placement of the MDS-coded symbols follows the MAN spirit and the delivery the YMA spirit.

Our second novel scheme attains the load in (12)–the proof can be found in [12, Appendix C]. In this scheme, we first code the files together, and then we apply another level of MDS coding before the placement. The general delivery has two steps: first we 'decode' the cache contents of the active users as in [5], and then we perform a sequence of YMA-same-file-deliveries to subsets of active users.

Comparisons: In general $\mathsf{R}^{\text{base}} \leq \mathsf{R}^{\text{new1}} \leq \mathsf{R}^{\text{de-cen}}$ (all evaluated for $r = \mathsf{r}'$), with $\mathsf{R}^{\text{base}} = \mathsf{R}^{\text{new1}} = \mathsf{R}^{\text{cen}}$ if $\mathsf{K} = \mathsf{K}'$.

By comparing the YMA load for a classical coded caching system with K' users and $r = \mathsf{r}' = \min\{\mathsf{N},\mathsf{K}'\}$ in (8), with the load of our first new proposed scheme in (11), we notice they have the exact same expression; the difference is in the memory requirement, which is $\mathsf{M}/\mathsf{N} = t/\mathsf{K}'$ for the YMA scheme with K' users and $\mathsf{M}/\mathsf{N} = t/\mathsf{K}\cdot\binom{\mathsf{K}}{t}/\binom{\mathsf{K}'}{t}$ for our first scheme with K' active users out of K total users. In other words, we need more cache space (quantified by the inverse of the MDS code rate) in order to serve K' online users and tolerate $\mathsf{K}-\mathsf{K}'$ offline users, compared to the classical YMA coded caching scheme for K' users. Note that the two schemes have the same memory requirement for $t = 1$.

Consider the following corner points

$$(\mathsf{M}_0^{\text{new1}},\mathsf{R}_0^{\text{new1}}) = (\mathsf{M}_0,\mathsf{R}_0^{\text{base}}) = (0,\mathsf{r}')|_{\mathsf{r}':=\min\{\mathsf{N},\mathsf{K}'\}}, \quad (13)$$

$$(\mathsf{M}_1^{\text{new1}},\mathsf{R}_1^{\text{new1}}) = \begin{cases}\left(\frac{\mathsf{N}}{\mathsf{K}'},\mathsf{r}'-\frac{\mathsf{r}'(\mathsf{r}'+1)}{2\mathsf{K}'}\right) & \mathsf{K}'-\mathsf{r}'\geq 2\\ \left(\frac{\mathsf{N}}{\mathsf{K}'},\frac{\mathsf{K}'-1}{2}\right) & \mathsf{K}'-\mathsf{r}'\in\{0,1\}\end{cases}. \quad (14)$$

The segment connecting the points (13) and (14) (achievable by memory sharing) outperforms the baseline scheme in the small memory regime and is optimal when the number of files is large enough; that connecting the points (13) and (12) is optimal in the small memory regime when the number of files is less than the number of users as stated in the next theorem.

B. Optimality Guarantees

As a converse bound, we can use *any* converse result for the classical coded caching system with K' users and N files; this is so because the performance of our hotplug system cannot be better than that of a system in which the server knows a priori which set of K' users will be active, and does the optimal placement and delivery for those users. With this type of converse bounds, we can show the following optimality result, whose proof can be found in [12, Appendix A].

Theorem 2 (Optimality Guarantees)**.** *For a* $(\mathsf{K},\mathsf{K}',\mathsf{N})$ *hotplug system. We have the following optimality guarantees.*

1) *When* $\mathsf{r}' = \min\{\mathsf{N},\mathsf{K}'\} = 1$, R^{base} *is optimal.*
2) *When* $\mathsf{K}\geq\mathsf{K}' = 2$ *and* $\mathsf{N} = 2$*, the optimal scheme has two non-trivial corner points:* $(\mathsf{M}_1^{\text{new1}},\mathsf{R}_1^{\text{new1}}) = (1,1/2)$ *and* $(\mathsf{M}^{\text{new2}},\mathsf{R}^{\text{new2}}) = (1/2,1)$.
3) *When* $\mathsf{K}\geq\mathsf{K}' = 2$ *and* $\mathsf{N}\geq 3$*, the only non-trivial optimal corner point is* $(\mathsf{M}_1^{\text{new1}},\mathsf{R}_1^{\text{new1}}) = (\mathsf{N}/2,1/2)$.
4) *When* $\mathsf{N}\leq\mathsf{K}'$ *and* $\mathsf{M}\leq\mathsf{N}/\mathsf{K}'$*, the corner point* $(\mathsf{M}_1^{\text{new2}},\mathsf{R}_1^{\text{new2}}) = (1/\mathsf{K}',\mathsf{N}(1-1/\mathsf{K}'))$ *is optimal.*
5) *When* $\mathsf{N}\geq\mathsf{K}'(\mathsf{K}'+1)/2$ *and* $\mathsf{M}\leq\mathsf{N}/\mathsf{K}'$*, the corner point* $(\mathsf{M}_1^{\text{new1}},\mathsf{R}_1^{\text{new1}}) = (\mathsf{N}/2,1/2)$ *is optimal.*
6) *When* $\mathsf{M}\geq\mathsf{N}(1-1/\mathsf{K})$, R^{base} *is optimal.*
7) R^{base} *is at most a factor 2 from optimal.*

Few remarks are in order.

[4]For example, the server does the YMA delivery as if the demand of each offline users is the same as the demand of the online user with the smallest index.

Item 5 with $\mathsf{K}' = 2$ only covers the first half of the memory range of Item 3; the second half is not covered by Item 6 as the memory regime in Item 6 depends on K (which can be any value no smaller than $\mathsf{K}' = 2$ in Item 3).

Theorem 2 does not provide a tight characterization for $\mathsf{r}' = \min\{\mathsf{N}, \mathsf{K}'\} = 2$ as the classical coded caching setting for two files is only partially solved [7] (only up to three users).

The proof for Item 1 and Item 6 is as for the classical coded caching system, with converse given by the cut-set bound [1, Theorem 2]; for Item 2 is given in Section IV and the converse is from [1]; for Item 3 the converse is [7, Theorem 3]; for Item 4 the converse is the cut-set bound [1, Theorem 2] for Item 5 the converse is [4, Theorem 2]; for Item 7 uses [4, Lemma 1] (where one upper bounds the performance of the proposed centralized scheme by that of the decentralized one–see also Remark 1; this is possible because the load of the classical coded caching model is bounded/finite when the number of users grows to infinity).

It is interesting to note that the exact optimality results in Theorem 2 (except Item 6) do not depend on K (the total number of users) but only on K' (the total number of *active* users). It is not obvious that this should be the case in general.

IV. Optimality for $\mathsf{K} \geq \mathsf{K}' = \mathsf{N} = 2$

We consider the hotplug system with $\mathsf{K} \geq 3$ users, $\mathsf{N} = 2$ files, and $\mathsf{K}' = 2$ active users, i.e., $\mathsf{K} - \mathsf{K}' = 1$ offline user.

In this section we go into the proof details for $\mathsf{K} = 3$ users only, which is the simplest case that highlights the novelty of our new schemes. The general case $\mathsf{K} \geq 3$ follows from the proofs in [12, Appendix B and Appendix C].

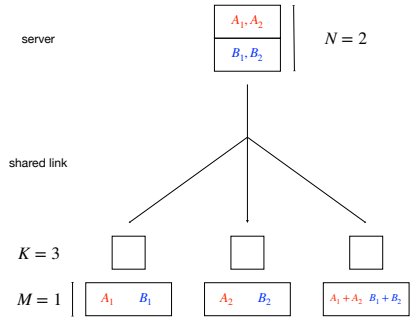
Next, we aim to show the achievability of the two non-trivial corner points of the optimal region for the classical coded caching setting with two users and two files [1], which is a converse bound for any hotplug system with $\mathsf{K} \geq \mathsf{K}' = \mathsf{N} = 2$. To prove the achievability of the non-trivial corner points $(1, 1/2)$ and $(1/2, 1)$ (in addition to the trivial points $(2, 0)$ and $(0, 2)$) we proceed as follows. We first derive the performance of our first new scheme, which achieves the point $(1, 1/2)$ by using MDS coded placement (where coding is only within each file). We then combine the coded placement idea of [1] with our MDS coded placement of our first new proposed scheme to show the achievability of the point $(1/2, 1)$.

Case $\mathsf{K} = 3$ *and* $\mathsf{M} = 1$*: First new scheme:* In Fig. 1 we consider memory size $\mathsf{M} = 1$ and $\mathsf{K} = 3$ users. The files are partitioned into two equal-size subfiles as $F_1 = (A_1, A_2)$ and $F_2 = (B_1, B_2)$. The subfiles of each file are coded with an MDS code of rate $2/3$. The cache contents are

$$Z_1 = (A_1, B_1),\ Z_2 = (A_2, B_2),\ Z_3 = (A_1 + A_2, B_1 + B_2),$$

as shown in Fig. 1a. The third user caches the parity bits.

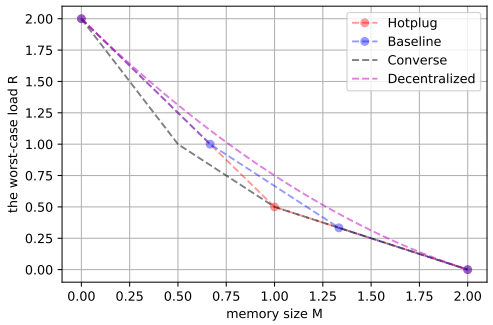
Regardless of which user is active and what the other two demand, each active user must receive the missing half of the demanded file. Fig. 1b gives the signals sent by the server according to Theorem 1, for two different demand vectors as a function of which user is offline; all the other demand vectors can be dealt similarly. The load is $\mathsf{R}_1^{\text{new1}} = 1/2$.



(a) Cache contents for our first new scheme for memory size $\mathsf{M} = 1$. The third user caches the two parity bits.

	User 1 offline	User 2 offline	User 3 offline
$\mathbf{d} = (1, 1, 1)$	A_1	A_2	$A_1 + A_2$
$\mathbf{d} = (1, 2, 1)$	$A_1 + B_1 + B_2$	A_2	$A_2 + B_1$

(b) The delivery for our first new scheme for memory size $\mathsf{M} = 1$, for two different demand vectors as a function of which user is offline.



(c) Memory-load tradeoffs.

Fig. 1: Memory-load tradeoffs for the hotplug system with $\mathsf{K} = 3$ users, $\mathsf{N} = 2$ files, and $\mathsf{K}' = 2$ active users. The converse is achievable for any $\mathsf{K} \geq 3$. The performance of our new schemes does not depend on K.

Fig. 1c shows the memory-load tradeoff attained by our first new scheme by the red dashed line, which is the lower convex envelope of the corner point $(1, 1/2)$ achieved by the novel scheme with the trivial corner points $(0, 2)$ and $(2, 0)$. The blue dashed line represents the memory-load tradeoff when all three users are active. The gray dashed line is the optimal memory-load for a classical coded caching system with two users and two files [1], which is achievable for any $\mathsf{K} \geq 3$. For comparison, we also added to the figure the performance of a decentralized coded caching scheme in magenta dashed line, given by (9) with $r = \mathsf{r}' = \min\{\mathsf{N}, \mathsf{K}'\} = 2$; the decentralized performance does not depend on K and is an upper bound for the centralized performance for any K.

This example shows that load savings are possible when the system is aware that only two users out of three can be active.

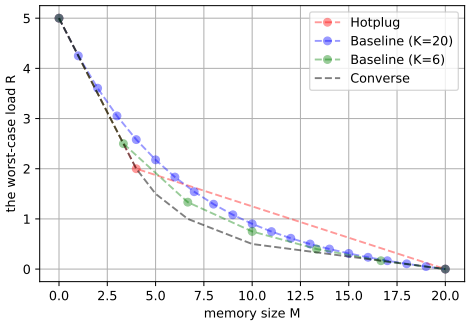


Fig. 2: Memory-load tradeoffs for the hotplug system for with $(\mathsf{K}', \mathsf{N}) = (5, 20)$ and various values of K.

Case $\mathsf{K} = 3$ *and* $\mathsf{M} = 1/2$*: Second new scheme:* Our first new scheme with MDS-coded placement attains only one corner point on the converse bound from [1]. In [1] it was shown that the point $(1/2, 1)$ can be achieved by coded placement in the classical setting with two files and two users. We next combine the idea of [1] with our MDS coded placement idea to show that $(1/2, 1)$ is achievable for $\mathsf{K} = 3 > \mathsf{K}' = \mathsf{N} = 2$.

Consider memory size $\mathsf{M} = 1/2$ and $\mathsf{K} = 3$ users. The files are partitioned as before but the cache contents are

$$\mathbf{A} = [A_1; A_2], \ \mathbf{B} = [B_1; B_2], \ \text{(files seen as column vectors)},$$
$$Z_1 = A_1 + B_1 = \mathbf{g}_1(\mathbf{A} + \mathbf{B}), \ \mathbf{g}_1 := [1, 0],$$
$$Z_2 = A_2 + B_2 = \mathbf{g}_2(\mathbf{A} + \mathbf{B}), \ \mathbf{g}_2 := [0, 1],$$
$$Z_3 = A_1 + A_2 + B_1 + B_2 = \mathbf{g}_3(\mathbf{A} + \mathbf{B}), \ \mathbf{g}_3 := [1, 1].$$

When the pair of active users requests the same file, the server transmits the requested file.

For the pair of active users (i, j) with $d_i = 1, d_j = 2$ the signal sent is $X = (\mathbf{g}_j\mathbf{A}, \ \mathbf{g}_i\mathbf{B})$. User i requesting file $\mathbf{A}$ does

$$\begin{bmatrix} Z_i - \mathbf{g}_i\mathbf{B} \\ \mathbf{g}_j\mathbf{A} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{g}_i \\ \mathbf{g}_j \end{bmatrix}}_{2 \times 2 \text{ full rank matrix}} \mathbf{A},$$

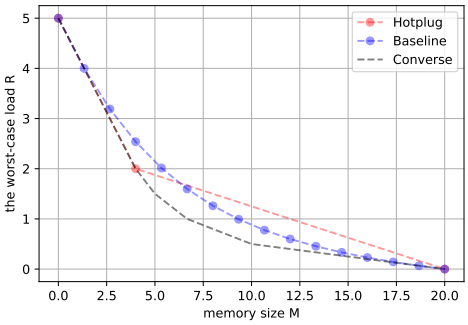
and similarly for user j requesting file $\mathbf{B}$. Thus we can serve any pair of users, regardless of the demand, by $\mathsf{R}^{\text{new2}} = 1$.

Case $\mathsf{K} \geq 3$*:* We showed that we can achieve all the corner points of the converse bound in [1] (which does not depends on K), thus we have the optimal coded caching strategy for the case $(\mathsf{K}, \mathsf{K}', \mathsf{N}) = (3, 2, 2)$. The same approach extends to any $\mathsf{K} \geq 3$, $\mathsf{K}' = \mathsf{N} = 2$ by using the general achievable schemes in [12, Appendix B and Appendix C].

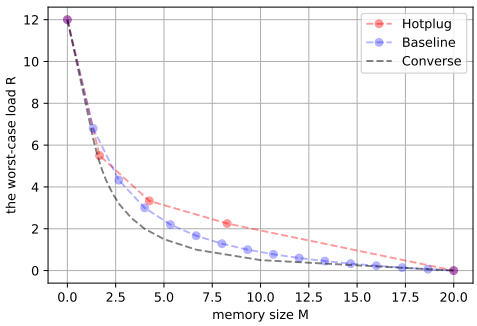
V. Numerical Evaluations

We conclude with some examples, to illustrate the performance of our new schemes.

Case $(\mathsf{K}', \mathsf{N}) = (5, 20)$: Fig. 2 shows the memory-load tradeoffs for the case $(\mathsf{K}', \mathsf{N}) = (5, 20)$ and various K. The performance of the first new scheme and of the converse bound does not depend on the value of K, while that of the baseline scheme worsen as K increases.



(a) Case $(\mathsf{K}, \mathsf{K}', \mathsf{N}) = (10, 5, 20)$.



(b) Case $(\mathsf{K}, \mathsf{K}', \mathsf{N}) = (15, 12, 20)$.

Fig. 3: Memory-load tradeoffs for the hotplug system with $(\mathsf{K}, \mathsf{N}) = (15, 20)$ and different values of K'.

Case $(\mathsf{K}, \mathsf{N}) = (15, 20)$: Fig. 3 shows the memory-load tradeoffs for two different values of K' for fixed $(\mathsf{K}, \mathsf{N}) = (15, 20)$. For $\mathsf{M} \in [0, \mathsf{N}/\mathsf{K}']$ in Fig. 3a, the first new scheme with MDS coded placement in Theorem 1 outperforms the baseline scheme in the small memory regime, and it is exactly optimal in the small memory regime.

VI. Conclusion

In this paper, we introduced the novel hotplug coded caching model to address a practical limitation of the original coded caching system, namely, to allow the server to start the delivery phase for a subset of active users, while the reaming users are offline. We proposed new coded caching schemes with MDS coded placement that are optimal in the small memory regime when some conditions hold. In general, our scheme is optimal to within a factor of 2. This shows that load savings are possible when the system is aware that only a subset of users will be active. Interestingly, when optimality can be be shown, the optimal performance only depends on the number active users and not on the total number of users; we are tempted to conjecture that this is true in general. Current work includes further extending optimality results and to consider other statistical models for the users' activity.

Acknowledgemnt

This work was supported in part by NSF Award 1910309.

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," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, 2017.

[3] K. Wan, D. Tuninetti, and P. Piantanida, "An index coding approach to caching with uncoded cache placement," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1318–1332, 2020.

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Characterizing the rate-memory tradeoff in cache networks within a factor of 2," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 647–663, 2018.

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

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

[7] C. Tian, "Symmetry, outer bounds, and code constructions: A computer-aided investigation on the fundamental limits of caching," *Entropy*, vol. 20, no. 8, p. 603, 2018.

[8] H. Ghasemi and A. Ramamoorthy, "Asynchronous coded caching with uncoded prefetching," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2146–2159, 2020.

[9] J. Liao and O. Tirkkonen, "Fundamental rate-memory tradeoff for coded caching in presence of user inactivity," *arXiv preprint arXiv:2109.14680*, 2021.

[10] Ç. Yapar, K. Wan, R. F. Schaefer, and G. Caire, "On the optimality of d2d coded caching with uncoded cache placement and one-shot delivery," *IEEE Transactions on Communications*, vol. 67, no. 12, pp. 8179–8192, 2019.

[11] 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, 2014.

[12] Y. Ma and D. Tuninetti, "On coded caching systems with offline users," *arXiv preprint arXiv:2202.01299*, 2022.