Coded Caching under Asynchronous Demands

Eleftherios Lampiris, Hamdi Joudeh, Giuseppe Caire, Petros Elia

***Abstract*— The work focuses on optimizing coded caching under asynchronous demands. We consider a single-stream setting where users are allowed to request content at arbitrary time-slots. Aiming to minimize the total system delay required to serve all users, i.e. from the moment of the first request to the delivery of the last bit of requested information, we design a pair of placement and delivery algorithms and show that the achievable performance is within a multiplicative factor of 2 from the optimal, under the assumption of uncoded placement, and within a multiplicative factor of 4.02 in the general placement case. Interesting characteristics of our algorithms are that i) a placement phase agnostic to the users' arrival times is adequate to provide a near-optimal delay, and ii) the proposed delivery algorithm requires low complexity and, at the same time, requires no non-causal information. Further, we show that systems are able to withstand some degree of asynchronicity without an increase in the delay compared to an equivalent synchronous setting. Finally, we highlight an interesting connection between coded caching under asynchronous demands and coded caching in wireless environments under uneven channel strengths.**

I. Introduction

The seminal work of Maddah-Ali and Niesen [1] explored the fundamental performance of the single-link, bottleneck setting where a server is connected to K cache-aided users. The server has access to a library of N files and each user is able to store the equivalent of M files, i.e. a fraction $\gamma \triangleq \frac{M}{N}$ of the library, while each user *synchronously* requests a single file from the library.

The placement and delivery algorithms constructed in [1] allowed, even if users requested different files, for each transmission to serve $K\gamma + 1$ users simultaneously. The delivery time achieved by the algorithm of [1], normalized with respect to file-size and link-rate, takes the form

$$T_{\text{MAN}} = \frac{K(1-\gamma)}{1+K\gamma}. \tag{1}$$

As proved in [2] (see also [3]) the performance in (1) is exactly optimal under uncoded placement, while it is within a multiplicative factor of 2.01 from the optimal, for general placement schemes [4]. The factor $1 + K\gamma$, appearing at the denominator of (1), is referred to as the *multicast gain*, as it reveals the number of users that receive a useful part of their requested file from a given message.

The initial framework of coded caching has been later adapted to various settings including wireless broadcast channels [5]–[8], multi-antenna channels [9]–[11], and distributed networks [12], [13], as well as incorporating different practical requirements such as user privacy [14]–[16], heterogeneous cache-sizes [17]–[20], decentralized coded caching [21], and revealing the synergy between caching and PHY layer [22]–[24], and the requirement for astronomical file sizes [25]–[27], to name a few[1].

A. Serving asynchronous requests

A key assumption in all the above works is that users demand content at the same time. In reality, though, content requests could begin and end at arbitrary times, causing delays in cache-aided systems. Coded caching capitalizes on the fact that many users ask for content simultaneously, and satisfies these demands through multicast messages that serve many users simultaneously. This desirable characteristic, though, is susceptible to asynchronous requests simply because, a multicast message may conceivably need to be transmitted multiple times so as to convey information to all its intended users, who may not be synchronously active. A consequence of such repetition is the reduction of the multicast gains.

While there are some basic measures one can take to alleviate this problem, such measures tend to be incomplete, heuristic, and at the same time introduce a variety of other drawbacks. A first such measure would be to wait until all K users have placed their requests. The obvious drawback is that the overall system delay increases, while users who arrive first end up being penalized by latecomers.

A second basic measure would be to divide each file into multiple smaller files. For example, one can imagine a movie being divided into multiple one-minute segments. The delivery algorithm of [1] would be applied for the requests of the first users that appear in the system, communicating to those users the first one-minute of their movie. After this transmission has been completed, the system would apply the algorithm of [1] in the first set of users plus the new set of users. The first set of users would receive the second one-minute segment of their movie, while the second set of users would receive the first one-minute segment. This solution would retain the coded caching gains, but requires to further subpacketize the files, thus further exacerbating the problem of finite file size and the related performance loss with respect to the ideal case of very large files.

E. Lampiris and G. Caire are with the Electrical Engineering and Computer Science Department, Technical University of Berlin, 10587, Germany, email: {lampiris, caire}@tu-berlin.de.
H. Joudeh is with the Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, email: h.joudeh@tue.nl.
P. Elia is with the Communication Systems Department at EURECOM, Sophia Antipolis, 06410, France, email: elia@eurecom.fr.
The work is supported by the European Research Council under the EU Horizon 2020 research and innovation program / ERC grant agreement no. 725929. (ERC project DUALITY) and by the European Research Council under the EU Horizon 2020 research and innovation program / ERC grant 789190 - CARENET.

[1]We note that the above is not a comprehensive bibliography of coded caching, something that would be beyond the scope of this paper.

B. State of art

Designing placement and delivery algorithms that can jointly facilitate asynchronous demands is a problem that has attracted attention, with multiple works seeking to design algorithms which can counter the aforementioned loss in coding gains [28]–[31].

The work in [29] considered a setting where users are able to request files in arbitrary time-slots while they may end their session before the requested file has been fully communicated. The work in [30] considers a single-stream setting, where users can potentially have heterogeneous cache-sizes, request content asynchronously and pose delay constraints on the reception of their requested content. The authors propose an optimization algorithm that outputs the multicast message order which would minimize the delivery time and abide by the delay constraints imposed by the users. Finally, the work in [31] considers a fog access network where each access point has a cache, storing in a decentralized manner. For this setting, the authors design coded caching transmission schemes with the goal of reducing the impact of asynchronous requests.

C. Contributions and paper outline

In this work, we focus on the delay required to satisfy all K user requests i.e., from the point when the first user arrives to the point the last bit is transmitted. While we recognise that the metric of choice may remain impractical for some systems (see discussion in Sec. IV), it nonetheless constitutes a first attempt into addressing this problem. For the proposed problem, we develop an algorithm that minimizes this delay and show that its performance is near-optimal; optimal within a multiplicative factor of 2 under uncoded placement and within a factor of 4.02 under any placement scheme.

We summarize some interesting outcomes of this work.

- We show that coded caching is able to withstand some degree of asynchronicity, in the sense that it can achieve the same performance as the equivalent synchronous system even when users place requests asynchronously. The two parameters that determine if the two delays are the same are i) the multicasting gain, and ii) the activation pattern of the users, i.e. the arrival times of each user in the system. The lower the nominal gain, the higher the potential to fully stave off asynchronicity. The set of asynchronous activation patterns that bare no asynchronicity penalty, are fully captured in our work.
- Users who request content last are not necessarily those responsible for the increase in the delay. As we will show, the increase in the overall delay can be typically attributed to a user that arrives "somewhere in the middle". It is this *threshold user* that defines the overall delay, and as a consequence, this delay would not reduce had the remaining users arrived earlier than they actually did.
- A very important outcome of our work is that the original placement algorithm of [1], which does not take into account such demand asynchronicity, is capable of achieving a near-optimal delay.
- Finally, the proposed placement and delivery schemes have the advantage of being computationally efficient. While previous works such as [29], [30] have proposed schemes with complexity growing as a function of the subpacketization, i.e. exponential in the number of users, we instead show that our scheme requires very low complexity.

II. Setting and notation

We consider a single-stream shared-link noise-less channel, where one server is connected to a set of K users. The server has access to a library of N files, $\{W^n\}_{n=1}^K$, each of size F bits, while each user is equipped with a cache of size $M \cdot F$ bits, i.e. a cache of normalized size $\gamma = \frac{M}{N}$. Each user requests a single file from the library, in some arbitrary time. Index d_k denotes that file W^{d_k} is requested by user k.

We assume that time is slotted, and requests are placed at any arbitrary time-slot. Further, the time difference between any two time-slots is equal to the time required to transmit $F/\binom{K}{K\gamma}$ bits. We denote with $t_k,\ k \in [K]$ the time-slot during which user k places its request, where without loss of generality $t_i \leq t_j,\ \forall i > j$ while we set $t_K = 0$, i.e. the user placing a request first is user K, followed by user $K-1$ and so on. The session is completed once all users are served, and we are interested in minimizing the overall session time.

Notation: We use $[K] \triangleq \{1, 2, ..., K\}$ for the set of the first K integers, while $|\cdot|$ denotes the cardinality of a set. For natural numbers n, k the binomial coefficient is defined as

$$\binom{n}{k} = \begin{cases} \frac{n!}{(n-k)!k!}, & n \geq k \\ 0, & n < k. \end{cases}$$

Remark 1. *We note that despite the fact that the arrival process is represented as a function of the subpacketization, this is done purely for the sake of simplicity of representing the different t_k values in integer form. In principle the arrival process is represented in basic units of time.*

III. Main results

Theorem 1. *The achievable overall delivery time for the single-stream channel where K users ask for files at arbitrary time-slots t_k and where each user is equipped with a cache of normalized size γ is given by*

$$T = \max_{w \in [K]} \left\{ \frac{t_w + \binom{K}{K\gamma+1} - \binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}} \right\} \quad (2)$$

and is within a multiplicative factor of 2 from the optimal delay under the assumption of uncoded placement, and within 4.02 under general placement schemes.

Proof. The achievability part of the proof is described in Section III-A, while the converse is proved in Section III-B. □

Remark 2. *From Theorem 1 we see that the achieved delay is not, necessarily, dependent on the user who arrives last, but rather on some intermediate user w who maximizes (2).*

Corollary 1. *The proposed algorithm is able to serve every request with delay equal to the synchronous system while each user $w \in [K]$ places a request to the system with delay*

$$t_w \leq \binom{K-w}{K\gamma+1}. \tag{3}$$

Proof. The proof is described in Section III-C. □

A. Achievable scheme

In this section we propose a new, order-optimal delivery scheme that minimizes the overall delay, from the time when the first user arrives to the system until the last bit of requested information has been communicated.

Remark 3. *The near-optimal placement and delivery algorithms presented in the next section do not require knowledge of the users' arrival times.*

a) Placement algorithm: The placement algorithm is borrowed from [1], thus we only describe this process here in short. Each file is divided into $S = \binom{K}{K\gamma}$ subpackets as

$$W^n \rightarrow \{W^n_\tau \;:\; \tau \subset [K],\; |\tau| = K\gamma\},\; \forall n \in [N] \tag{4}$$

such that each subfile is described by a $K\gamma$-length index whose elements belong in set $[K]$. Subsequently, user $k \in [K]$ stores all subfiles whose index contains k, and thus cache $\mathcal{Z}_k$ of user k, is filled as follows

$$\mathcal{Z}_k = \{W^n_\tau \;:\; \tau \ni k, \forall n \in [N]\}. \tag{5}$$

b) Delivery algorithm: During the delivery phase, each user is allowed to request a single file at an arbitrary time-slot. Without loss of generality we assume that users place requests in a descending order[2], i.e. user K arrives first, then user $K-1$ and so on.

The multicast messages that will eventually convey the requested files to the users are formed as in the algorithm of [1]. Hence, a multicast message aimed for users of set $\sigma \subseteq [K]$, $|\sigma| = K\gamma + 1$, takes the form

$$X_\sigma = \bigoplus_{k \in \sigma} W^{d_k}_{\sigma \setminus \{k\}}. \tag{6}$$

Remark 4. *We note that any message X_σ is just a placeholder which is then evaluated during the time of the transmission. If all users in σ are active then the message is formed as in (6). In a different case the message is formed using the subset of users from σ that are currently active.*

During each iteration of the algorithm (time-slot t) the algorithm uses sets $P(t)$ and $\mathcal{K}^{\text{act}}$. Set $P(t)$ stores all the remaining multicast messages that need to be communicated, and set $\mathcal{K}^{\text{act}}$ represents the set of active users. At each time-slot t the algorithm is responsible of selecting a new multicast message to communicate to the users. This message is picked from $P(t)$ such that the set of its intended users is a maximal subset of the set of active users. In other words, the algorithm picks a message that would be useful to as many active users as possible.

[2]In a different case we can simply rename the users to reflect this ordering.

Algorithm 1: Communicating under asynchronous requests

1 Initialize: $P(0) = \{X_\sigma, \sigma \subseteq [K], |\sigma| = K\gamma + 1\}$, $\mathcal{K}^{\text{act}} = \emptyset,\;\; t = 0$
2 **while** $\mathcal{K}^{act} \neq [K]$ & $P(t) \neq \emptyset$ **do**
3   **if** *user k arrives* **then**
4         $$\mathcal{K}^{\text{act}} = \mathcal{K}^{\text{act}} \cup \{k\}$$
5   Pick multicast message X_σ such that
$$X_\sigma \in P(t) \quad : \sigma \in \arg \max_{\substack{\phi \subseteq [K] \\ |\phi| = K\gamma+1}} \left\{ \left|\phi \cap \mathcal{K}^{\text{act}}\right| \right\}$$
6   Transmit X_σ.
7   **if** $\sigma \subseteq \mathcal{K}^{act}$ **then**
8         $P(t) = P(t) \setminus \{X_\sigma\}$.
9     $t = t + 1$
10     $P(t) = P(t-1)$

c) Algorithm description: Algorithm 1 begins by initializing sets $P(0)$ and $\mathcal{K}^{\text{act}}$. The main part of the algorithm is governed by a *While* loop, where each iteration corresponds to a single time-slot.

Inside the *While* loop is an *If* condition, which is used to ascertain whether a new user has become active, in which case set $\mathcal{K}^{\text{act}}$ is updated to include this new user (Step 4).

Further, in Step 5 the algorithm selects one multicast message $X_\sigma \in P(t)$, such that it forms a maximal subset of the active-users set, i.e. is aimed to serve as many users as possible. If all users of the selected multicast message are active, then the algorithm removes X_σ from set $P(t)$ (Steps $7-8$). Finally, in Step 9 the the time index is updated as well as the message set $P(t)$ (Step 10) and the algorithm proceeds with the next iteration.

Remark 5. *The complexity of the algorithm at some time-slot t depends on identifying one multicast message which belongs in set $P(t)$ and serves as many active users as possible.*

d) Delay calculation: The metric of interest is the delay required to serve all K users. In each time-slot t the algorithm is responsible for selecting one multicast message for transmission, such that it serves the maximum possible active users. At each time-slot t_k the size of set $P(t_k)$ is

$$|P(t_k)| \geq \binom{K}{K\gamma+1} - \binom{K-k}{K\gamma+1} \tag{7}$$

because the algorithm cannot remove from $P(t)$ messages for users that have not appeared. At any given slot t_k the algorithm will need to iterate a minimum of $|P(t_k)|$ slots such that it would serve the remaining messages in $P(t_k)$. Thus, the achieved delay needs to be greater or equal to

$$T \geq \max_{m \in [K]} \left\{ \frac{t_m + \binom{K}{K\gamma+1} - \binom{K-m}{K\gamma+1}}{\binom{K}{K\gamma}} \right\}. \tag{8}$$

Naming the user who maximizes the left-hand-side of (10) as user w we observe that

$$t_w+\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}\geq t_y+\binom{K}{K\gamma+1}-\binom{K-y}{K\gamma+1}$$
$$\binom{K-y}{K\gamma+1}-\binom{K-w}{K\gamma+1}\geq t_y-t_w,\ \forall y<w. \quad (9)$$

From (9) we see that the number of messages intended to at least one user from the set of users $\{y+1,...,w\}$ are less than the number of time-slots between t_w and t_y. Hence, in each time-slot $t\geq t_w$ the algorithm will transmit and then remove one message from set $P(t)$. This further means that the achievable delay is given by

$$T=\max_{w\in[K]}\left\{\frac{t_w+\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}}\right\}. \quad (10)$$

□

We proceed with two examples that aim to illustrate the mechanics of Algorithm 1.

***Example* 1.** *Let us consider a single-stream setting where $K=4$ users will request files from a library of $N=4$ files, and Each user has a cache of normalized size $\gamma=\frac{1}{4}$. The cached content at user $k\in[4]$ takes the form*

$$\mathcal{Z}_k=\{W_k^n,\forall n\in[N]\}. \quad (11)$$

We assume[3] that the arrival times of the users are as follows

$$(t_4,t_3,t_2,t_1)=(0,0,1,3). \quad (12)$$

According to (2)*, the achievable delay is given by*

$$T=\frac{1}{4}\cdot\max_{k\in[4]}\big\{6+t_4,6+t_3,5+t_2,3+t_1\big\}=\frac{3}{2} \quad (13)$$

i.e., equal to the synchronous case.

The delivery phase begins when users $3,4$ *arrive to the system. The set of active users $\mathcal{K}^{act}$ is updated, while the set of desired multicast messages is now[4]*

$$P(0)=\{D_3\oplus C_4,\ D_1\oplus A_4,\ D_2\oplus B_4,\ C_2\oplus B_3,$$
$$C_1\oplus A_3,A_2\oplus B_1\}.$$

During some time-slot t the algorithm selects one multicast message from set $P(t)$ such that the biggest possible number of users associated with this message also appear in set $\mathcal{K}^{act}$. Thus, in time-slot $t=0$ the only message that satisfies the above requirement is $D_3\oplus C_4$. This message is sent, and subsequently removed from set $P(0)$ so that a new iteration of the algorithm starts.

At the beginning of time-slot 1*, user* 2 *arrives at the system, which prompts the update of the active users set to $\mathcal{K}^{act}=\{2,3,4\}$. There are two possible messages that can be selected for transmission, $C_2\oplus B_3$ and $D_2\oplus B_4$, since both contain the maximal number of active users. One of those messages is selected at random, it is transmitted, and then is removed from set $P(1)$. At the beginning of time-slot $t=2$, and given that no new user joins, the other of the above two messages is selected, sent, and removed from $P(2)$.*

At time-slot $t=3$ all users are active, hence the algorithm can proceed with the transmission of the remaining messages in an arbitrary order. Completing the transmission required 6 *time-slots, thus the delay of the algorithm is $T=\frac{6}{4}$.*

***Example* 2.** *We consider the same system of $K=4$ users, but now the arrival times of the users are*

$$(t_4,t_3,t_2,t_1)=(0,1,2,3). \quad (14)$$

Similarly, the achievable delay becomes

$$T=\frac{1}{4}\cdot\max_{k\in[4]}\big\{6+t_4,6+t_3,5+t_2,3+t_1\big\}=\frac{7}{4}. \quad (15)$$

When user 4 *places a request the set of active users $\mathcal{K}^{act}$ is updated, while the set of desired multicast messages is*

$$P(0)=\{D_3\oplus C_4,\ D_1\oplus A_4,\ D_2\oplus B_4,\ C_2\oplus B_3,$$
$$C_1\oplus A_3,A_2\oplus B_1\}.$$

Time-slot t = 0*: The algorithm select one of the messages $D_3\oplus C_4$, $D_1\oplus A_4$, $D_2\oplus B_4$. This message is sent, but not removed from set $P(0)$ due to the fact that not all of its intended users are active.*

Time-slot $t=1$*: User* 3 *arrives at the system, which prompts the update of the active-users set to $\mathcal{K}^{act}=\{3,4\}$. The only message that corresponds to a maximal subset of active users is $D_3\oplus C_4$, which is transmitted and removed from $P(1)$.*

Time-slot $t=2$*: The set of active-users is $\mathcal{K}^{act}=\{2,3,4\}$, hence the algorithm selects one of $D_1\oplus A_4$, $D_2\oplus B_4$ messages for transmission and then removes it from set $P(2)$.*

Time-slot $t=3$*: At this point all users are active hence, the algorithm can proceed with the transmission of the remaining messages in an arbitrary order. Completing the transmission required* 7 *time-slots, thus the delay of the algorithm is $T=\frac{7}{4}$.*

B. Converse

We consider a hypothetical *augmented* system comprised of w users, where w corresponds to the user that maximizes expression (2). These w users place requests simultaneously at time-slot t_w and which requests can be served using the algorithm of [1], with delay

$$T_{[w]}=\frac{t_w}{\binom{K}{K\gamma}}\frac{w(1-\gamma)}{1+w\gamma} \quad (16)$$

which is exactly optimal under uncoded placement schemes [2], [3] and within a multiplicative factor of 2.01 for general placement schemes [4]. Hence, a lower bound on the above augmented system is also a lower bound on the achievable result of Theorem 1. This bound takes the form

$$T\geq\frac{t_w}{\binom{K}{K\gamma}}+\frac{1}{b}\frac{w(1-\gamma)}{1+w\gamma} \quad (17)$$

[3]For simplicity we name the requests of each user by consecutive letters, i.e. $A\triangleq W^{d_1}$, $B\triangleq W^{d_2}$, and so on.

[4]The multicast messages of set $P(t)$ are formed using the requests of the active users, and are updated every time a user arrives at the system.

where factor $b = 1$ when considering uncoded placement, while $b = 2.01$ for general placement schemes. Comparing (17) with the achievable delay in (2), we get

$$\frac{\frac{t_w+\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}}}{\frac{t_w}{\binom{K}{K\gamma}}+\frac{1}{b}\frac{w(1-\gamma)}{1+w\gamma}} \overset{(i)}{\leq} \frac{\frac{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}}}{\frac{1}{b}\frac{w(1-\gamma)}{1+w\gamma}} \overset{(ii)}{\leq} 2b \tag{18}$$

where inequality (i) is achieved by using the fact that the ratio is maximized when $t_w = 0$, while inequality (ii) uses a result from [7], and which concludes the proof. □

C. Intuition on the results

An interesting characteristic of our algorithm is that it can—in certain cases—maintain the fully synchronous performance corresponding to the scenario where all the requests come at the very beginning (time $t = 0$). As we showed above, by prioritising at any given time-slot the multicast messages intended for a maximal subset of users who are simultaneously active, we are able, to a certain extent achieve the same delay as the synchronous system.

In this section we characterize the maximum delay that is allowed for any user to place a request while still allowing the achieved delay to be equal to the synchronous case.

Using the result of Theorem 1 and requiring the delay to be equal to that of the synchronous case (cf. (1)) we have

$$\frac{\binom{K}{K\gamma+1}}{\binom{K}{K\gamma}} \geq \frac{t_w+\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}}, \ \forall w \in [K] \tag{19}$$

hence, for the asynchronous system to enjoy the same delay as the synchronous one, then the arrival time-slot of any user may not exceed

$$t_w \leq \binom{K-w}{K\gamma+1}. \tag{20}$$

In Fig. 1 we display the maximum allowed, per-user delay which can still achieve the delay of the synchronous problem.

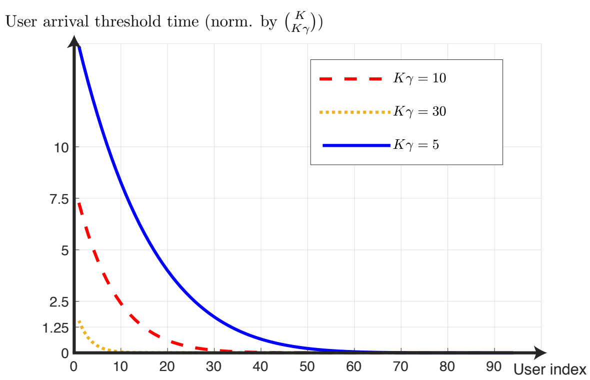

Fig. 1. The delay thresholds that achieve the synchronous delay in settings with $K = 100$ users.

Remark 6. *Let us focus on the setting of Figure 1 that is comprised of $K = 100$ users with caches of normalized size $\gamma = \frac{5}{100}$ (corresponding to $K\gamma = 5$, solid blue line). We notice that even if 30 users place requests with delay $1.5\binom{100}{5}$ time-slots, the system will still experience the synchronous system delay, thus incurring no penalty from asynchronicity. Similarly for the setting with $\gamma = \frac{1}{10}$ we observe that 12 users can become active at time-slot $1.5\binom{100}{10}$ while allowing the system to experience the same delay as the synchronous one.*

IV. Final remarks & future work

It is interesting to observe the striking similarity between the current setting and the (wireless) coded caching setting with non-identical link capacities (cf. [7], [8]). In this latter, degraded channel setting each user—instead of having an equally strong channel of unit normalized capacity (as in [1])—experiences a reduced strength link of some capacity $\alpha_k \in (0, 1]$. As shown in [7], [8] the order-optimal delay takes the form

$$T_{\text{deg}} = \max_{w\in[K]} \left\{ \frac{1}{\alpha_w} \frac{\binom{K}{K\gamma+1}-\binom{K-w}{K\gamma+1}}{\binom{K}{K\gamma}} \right\} \tag{21}$$

which nicely resembles (2) and which draws a parallel between the effects of channel asymmetry and temporal asynchronicity.

Future work: We note that in the longer version of this work we explore and describe the following scenarios as well:

- Completing user-requests in a "first-in-first-out" manner and minimizing the time each user spends in the system.
- Designing coded caching for delay-sensitive applications, where each request is accompanied by a strict completion time-frame.

References

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. on Information Theory*, vol. 60, pp. 2856–2867, May 2014.

[2] 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.

[3] 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, pp. 1281–1296, Feb 2018.

[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, pp. 647–663, Jan 2019.

[5] J. Zhang and P. Elia, "Wireless coded caching: A topological perspective," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 401–405, June 2017.

[6] E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 2960–2964, June 2017.

[7] E. Lampiris, J. Zhang, O. Simeone, and P. Elia, "Fundamental limits of wireless caching under uneven-capacity channels," in *Internation Zurich Seminar (IZS)*, Feb 2020.

[8] H. Joudeh, E. Lampiris, P. Elia, and G. Caire, "Fundamental limits of wireless caching under mixed cacheable and uncacheable traffic," in *IEEE International Symposium on Information Theory (ISIT)*, June 2020.

[9] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *IEEE Transactions on Information Theory*, vol. 62, pp. 7253–7271, Dec 2016.

[10] E. Lampiris, A. Bazco-Nogueras, and P. Elia, "Resolving the feedback bottleneck of multi-antenna coded caching," *arXiv preprint arXiv:1811.03935*, 2018.

[11] E. Lampiris and P. Elia, "Bridging two extremes: Multi-antenna coded caching with reduced subpacketization and CSIT," *SPAWC*, 2019.

[12] M. Bayat, R. K. Mungara, and G. Caire, "Achieving spatial scalability for coded caching over wireless networks," *arXiv preprint arXiv:1803.05702*, 2018.

[13] M. Bayat, K. Wan, and G. Caire, "Coded caching over multicast routing networks," 2020.

[14] F. Engelmann and P. Elia, "A content-delivery protocol, exploiting the privacy benefits of coded caching," in *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pp. 1–6, 2017.

[15] K. Wan and G. Caire, "On coded caching with private demands," *arXiv 1908.10821*, Aug. 2019.

[16] Q. Yan and D. Tuninetti, "Fundamental limits of caching for demand privacy against colluding users." arXiv 2008.03642, Aug. 2020.

[17] A. Sengupta, R. Tandon, and T. C. Clancy, "Layered caching for heterogeneous storage," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, pp. 719–723, Nov 2016.

[18] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Coded caching for heterogeneous systems: An optimization perspective," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5321–5335, 2019.

[19] E. Lampiris and P. Elia, "Full coded caching gains for cache-less users," *IEEE Transactions on Information Theory*, Aug 2020.

[20] M. Mohammadi Amiri, Q. Yang, and D. Gündüz, "Decentralized caching and coded delivery with distinct cache capacities," *IEEE Transactions on Communications*, vol. 65, pp. 4657–4669, Nov 2017.

[21] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, pp. 1029–1040, Aug 2015.

[22] S. P. Shariatpanahi, G. Caire, and B. Hossein Khalaj, "Physical-layer schemes for wireless coded caching," *IEEE Transactions on Information Theory*, vol. 65, pp. 2792–2807, May 2019.

[23] A. Tölli, S. P. Shariatpanahi, J. Kaleva, and B. H. Khalaj, "Multi-antenna interference management for coded caching," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2091–2106, 2020.

[24] E. Lampiris, P. Elia, and G. Caire, "Bridging the gap between multiplexing and diversity in finite snr multiple antenna coded caching," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1272–1277, 2019.

[25] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., "Finite-length analysis of caching-aided coded multicasting," *IEEE Transactions on Information Theory*, vol. 62, pp. 5524–5537, Oct 2016.

[26] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Transactions on Information Theory*, vol. 63, pp. 5821–5833, Sep. 2017.

[27] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 36, pp. 1176–1188, June 2018.

[28] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *2015 IEEE International Conference on Communications (ICC)*, pp. 5559–5564, 2015.

[29] Q. Yang, M. Mohammadi Amiri, and D. Gündüz, "Audience-retention-rate-aware caching and coded video delivery with asynchronous demands," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7088–7102, 2019.

[30] H. Ghasemi and A. Ramamoorthy, "Asynchronous coded caching with uncoded prefetching," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2020.

[31] Y. Jiang, W. Huang, M. Bennis, and F. Zheng, "Decentralized asynchronous coded caching design and performance analysis in fog radio access networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 540–551, 2020.