2019

Online Asynchronous Coded Caching

Hooshang Ghasemi
Iowa State University, ghasemi@iastate.edu

Aditya Ramamoorthy
Iowa State University, adityar@iastate.edu

Online Asynchronous Coded Caching

Abstract
Coded caching is a technique for reducing peak data rate in content delivery systems that employ caching. The original formulation of the coded caching problem assumes that the file requests from the users are synchronous, i.e., they arrive at the same time. In this work, we consider the asynchronous setting where the file requests are revealed to the server in an online fashion. We propose a novel online algorithm for this problem building on our prior work for the offline setting (where the server knows the request arrival times and deadlines in advance). Our simulation results demonstrate that our proposed online algorithm allows for a natural tradeoff between the feasibility of the schedule and the rate gains of coded caching.

Disciplines
Electrical and Computer Engineering

Online Asynchronous Coded Caching

Hooshang Ghasemi and Aditya Ramamoorthy
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50010, U.S.A.
Email: {ghasemi, adityar}@iastate.edu

***Abstract*—Coded caching is a technique for reducing peak data rate in content delivery systems that employ caching. The original formulation of the coded caching problem assumes that the file requests from the users are synchronous, i.e., they arrive at the same time. In this work, we consider the asynchronous setting where the file requests are revealed to the server in an online fashion. We propose a novel online algorithm for this problem building on our prior work for the offline setting (where the server knows the request arrival times and deadlines in advance). Our simulation results demonstrate that our proposed online algorithm allows for a natural tradeoff between the feasibility of the schedule and the rate gains of coded caching.**

I. Introduction

Caching is a core component of solving the problem of large scale content delivery over the Internet. Coded caching [1] has emerged as a powerful application of index coding/network coding ideas to the problem of caching. The work of [1] demonstrated that significant reductions in the induced traffic in the caching network were possible using their techniques.

However, the original formulation of the coded caching problem assumes that all file requests from the users arrive at the server at the same time, i.e., it works with an (idealized) synchronized model. From a practical perspective, it is important to consider the case when the requests of the users are not synchronized and users have prescribed deadlines.

In our prior work [2], [3] we considered both the offline and the online variants of this problem. In the offline scenario, where the server knows the arrival times and deadlines of each user in advance, we posed a linear programming (LP) problem which if feasible, allows the server to determine a schedule of transmissions, such that each user can be satisfied within its deadline. We also highlighted certain features of the online scenario that differentiate it qualitatively from the offline scenario.

The delay sensitive coded caching problem was first studied in [4]. They considered the decentralized coded caching model and a setting where each subfile has a specific deadline. Only the online case was considered and heuristics for transmission from the server were proposed. However, the transmission time for each packet was not considered in their formulation. Coded caching of video files taking into account the audience retention rate for each video has been investigated in [5].

The central issue in the online scenario is the dilemma that the server faces in whether or not to transmit a packet at a certain time slot. To see this, suppose that the first user request arrives at the server. Suppose that the server wants to be very risk-averse, i.e., it wants to make sure that the transmission schedule satisfies each user. In this case it should immediately start sending packets to satisfy user 1. However, this strategy will reduce the overall gain of coded caching as the transmissions only benefit one user. Note that if user 1's deadline is not too stringent then it makes sense for the server to wait until another user request comes in. It can then send equations that are potentially useful to two users while still being feasible. The coded caching rate gain and the probability that the transmission schedule is feasible are competing objectives that need to be addressed in the online scenario.

Main contributions: We present a novel heuristic for the online version of the asynchronous coded caching problem. Our proposed algorithm is inspired by our LP formulation for the offline scenario [2]. Roughly speaking, we solve a new offline-like LP every time a new user request comes to the server. Its solution is used to identify equations that simultaneously "benefit" multiple users. Here, the benefit to a given user takes into account the stringency of its deadline. Our simulation results demonstrate that the probability that our online algorithm is feasible is quite high and can be traded off for the rate gains of coded caching by varying a system-defined threshold.

II. Problem Formulation

A coded caching system contains K users and a server with $N \geq K$ files[1], denoted W_n, $n = 1, \ldots, N$, each of size F subfiles, where a subfile is a basic unit of storage. For a positive integer n, let $[n]$ denote the set $\{1, \ldots, n\}$. The subfiles are denoted by $W_{n,f}$ so that $W_n = \{W_{n,f} : f \in [F]\}$. Each user is connected to the server through an error-free, broadcast shared link with a local cache of size MF subfiles. User i's cache is denoted by Z_i where $Z_i \subseteq \{W_{n,f} : \; n \in [N], \; f \in [F]\}$ and Z_i contains at most MF subfiles.

Consider a system with a fixed uncoded cache placement. The asynchronous setting can be formulated as follows. In the delivery phase, suppose that user i requests file W_{d_i}, $d_i \in [N]$, from the server at time T_i. The i-th user specifies a deadline $T_i + \Delta_i$ (where Δ_i is a positive integer) by which he/she

[1]We assume that $N \geq K$ as it corresponds to the worst case rate where each of the K users can request a different file. Furthermore, it is also the more practical scenario.

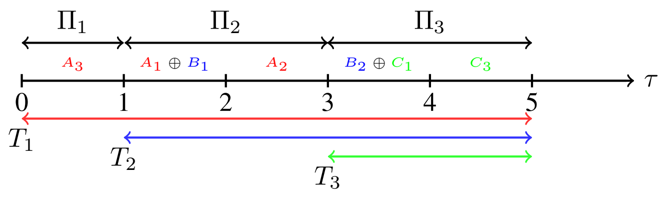

Fig. 1: Offline solution corresponding to the Example 1. The double-headed arrows show the active time slots for each user. The transmitted equations are shown above the timeline.

expects the request to be satisfied[2]. We let $\Omega^{(i)}$ denote the indices of the subfiles that are not present in the i-th user's cache. Thus, $\Omega^{(i)} = \{f : \ f \in [F], \ W_{d_i,f} \notin Z_i\}$. We assume that time $\tau \geq 0$ is slotted and that the size of each subfile is such that it needs r time-slots to be transmitted over the shared link.

In [2], we formulated the offline and the online versions of the asynchronous setting. In the offline case, the arrival time, deadline, and the requested file of each user are known to the server at the very beginning, i.e., time $\tau = T_1$. In the more challenging online case, information about $\{T_i, \Delta_i, d_i\}$ is revealed to the server only at time T_i for $i \in [K]$. In [2] we presented an optimal algorithm for the offline case when the server transmits *all-but-one* equations.

In this work we study the online scenario under the assumption of all-but-one equations. Given a sequence of $\{T_i, \Delta_i, d_i\}$ values that are revealed to the server over time, the output of our algorithm is a specification of the transmissions at each time slot. Without loss of generality, we assume that $T_1 \leq T_2 \leq \ldots \leq T_K$. Let $T_{\max} = \max_i(T_i + \Delta_i)$. As in [2], upon sorting the set of arrival times and deadlines, we can divide the interval $[T_1, T_{\max})$ into $\beta \leq 2K - 1$ intervals denoted $\Pi_1, \ldots, \Pi_\beta$ (see Fig. 1 for an example). We define,

$$\begin{aligned} U_\ell &= \{i \in [K] : \ [T_i, T_i + \Delta_i) \cap \Pi_\ell = \Pi_\ell\}, \text{and} \\ D_\ell &= \{d_i \in [N] : \ i \in U_\ell\}. \end{aligned}$$

Thus, U_ℓ is the set of active users in time interval Π_ℓ and D_ℓ is the corresponding set of active file requests.

For each user $i \in U$ we let $\mathcal{F}_{\{i,U\}}$ denote the indices of all missing subfiles of the i-th user that have been stored in the cache of the other users in U. That is, $\mathcal{F}_{\{i,U\}} = \{f \in \Omega^{(i)} : W_{d_i,f} \in Z_j \text{ for all } j \in U \setminus \{i\}\}$. We call U a user group if $\mathcal{F}_{\{i,U\}} \neq \emptyset$ so that there is at least one all-but-one type of equation corresponding to U.

We let $\mathcal{U}_\ell$ be a subset of the power set of U_ℓ (i.e. the set of all subsets of U_ℓ) such that each member of $\mathcal{U}_\ell$ is a user group. Also, we let $\mathcal{I}_U \subseteq [\beta]$ denote the indices of all time intervals that users in user group U are simultaneously active. That is $\mathcal{I}_U = \{\ell \in [\beta] : \ U \in \mathcal{U}_\ell\}$. For each user $i \in [K]$ and a missing subfile $f \in \Omega^{(i)}$ there may be more that one user group U so that $f \in \mathcal{F}_{\{i,U\}}$. We let $\mathcal{U}_{\{i,f\}}$ be a set of such user groups. Therefore, $\mathcal{U}_{\{i,f\}} = \{U \in \mathcal{U}_\ell : \ \ell \in [\beta], U \ni i, \ f \in \mathcal{F}_{\{i,U\}}\}$.

[2]It is not too hard to see that in the absence of deadlines the server can simply wait for enough user requests to arrive before starting transmission. Thus, the deadline-free case essentially reduces to the synchronous setting.

Example 1: Consider a system with $N = 3$ files with $W_1 = A$, $W_2 = B$, and $W_3 = C$. Each file is divided to three subfiles, so that $F = 3$. Also, there are $K = 3$ users with the following cache content.

$$\begin{aligned} Z_1 &= \{W_{2,1}, W_{2,2}, W_{3,3}\} = \{B_1, B_2, C_3\}, \\ Z_2 &= \{W_{1,1}, W_{2,3}, W_{3,1}\} = \{A_1, B_3, C_1\}, \text{ and} \\ Z_3 &= \{W_{2,2}, W_{3,2}, W_{2,1}\} = \{B_2, C_2, B_1\}. \end{aligned}$$

Thus, $M = 1$. The arrival times are $T_1 = 0$, $T_2 = 1$, $T_3 = 3$, and deadlines are $\Delta_1 = 5$, $\Delta_2 = 4$, and $\Delta_3 = 2$ (see Fig. 1 for an illustration). The i-th user requests file W_i, for $i = 1, 2, 3$. Therefore, $\Omega^{(1)} = \{1, 2, 3\}$, $\Omega^{(2)} = \{1, 2\}$, and $\Omega^{(3)} = \{1, 3\}$. An offline solution for this system is illustrated in Fig. 1. In [2], we demonstrated that in the online scenario, the server should code across missing subfiles intended for a given user; this is not required in the synchronous setting.

In this case we have $\mathcal{F}_{\{1,\{1,2\}\}} = \{1\}$ and $\mathcal{F}_{\{2,\{1,2\}\}} = \{1, 2\}$ therefore user group $U = \{1, 2\}$ can be used to transmit two equations $W_{\{d_1,1\}} \oplus W_{\{d_2,1\}}$ and $W_{\{d_1,1\}} \oplus W_{\{d_2,2\}}$. Since user 2 has two missing subfiles in $\mathcal{F}_{\{1,\{1,2\}\}}$, the server can also transmit a linear combination of these subfiles along with $W_{\{d_1,1\}}$, e.g., $W_{\{d_1,1\}} \oplus W_{\{d_2,1\}} \oplus W_{\{d_2,2\}}$. The advantage of the last equation is that this equation will be useful for the second user if the server decides to transmit either of $W_{\{d_2,1\}}$ or $W_{\{d_2,2\}}$ later.
In what following, each missing subfile will be treated as an element of a large enough finite field $\mathbb{F}$ by an appropriate embedding.

III. Online Asynchronous Coded Caching

Our proposed algorithm is inspired by our LP formulation of the offline scenario. In particular, we solve an LP which is similar to [2] each time a new user request comes into the system. Following this, we find a candidate list of feasible user groups that can be chosen for transmission. We calculate a metric for each feasible user group U depending upon the stringency of the deadlines of the users in U. If this metric is above a system-defined threshold, we transmit an equation corresponding to this user group in the time-slot following the user's request. Subsequently, we update certain variables and the process continues for each time slot thereafter. When the next user request arrives into the system, the history of the variable assignments is used to solve a new LP (similar to [2]), and the process continues recursively.

In what follows, we explain the working of this algorithm and demonstrate that it has excellent performance in simulation. We begin by recapitulating the offline asynchronous coded caching that we have introduced in [2]. Let $|\Pi_\ell|$ denote the length of the time interval Π_ℓ. For each time interval Π_ℓ with $\ell = 1, \ldots, \beta$ and for each $U \in \mathcal{U}_\ell$ we define variable $x_U(\ell) \in [0, |\Pi_\ell|]$ that represents the portion of time interval Π_ℓ that is allocated to an equation that benefits user group U. For each missing subfile $W_{\{d_i,f\}}$ and each $U \in \mathcal{U}_{\{i,f\}}$ we define variable $y_{\{i,f\}}(U) \in [0, r]$ that represents the portion of the missing subfile $W_{\{d_i,f\}}$ transmitted within some or all of the equations associated with $x_U(\ell)$ for $\ell \in \mathcal{I}_U$.

In [2], we proposed the following LP that minimizes the overall rate of transmission from the server in offline case while respecting all the deadline constraints of the users.

$$\min \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \tag{1}$$

$$\text{s.t. :}$$

$$\sum_{U \in \mathcal{U}_\ell} x_U(\ell) \le |\Pi_\ell|, \quad \text{for } \ell = 1, \dots, \beta,$$

$$\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \le \sum_{\ell \in \mathcal{I}_U} x_U(\ell), \quad \text{for } i \in U,\ U \in \cup_{\ell=1}^{\beta} \mathcal{U}_\ell,$$

$$\sum_{U \in \mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = r, \quad \text{for } f \in \Omega^{(i)},\ i \in [K],$$

$$x_U(\ell),\ y_{\{i,f\}}(U) \ge 0.$$

The solution of (1) can be interpreted as a set of equations to satisfy the demand of each user within deadline. A detailed explanation can be found in [2]. Applying the interpretation in [2] to the solution of (1) for the system in Example 1 will result in, e.g., the set of equations in Fig. 1.

In the online scenario we solve a LP similar to (1) every time a new user request comes to the server. However, the interpretation of the solution in terms of deciding the equations to be transmitted is different. Moreover, some of variables are already fixed based on the previous choices made by the server. Consider a time $\tau = T_k$ when the request of the k-th user arrives at the server. We let $\mathcal{U}_{\text{sent}}(\tau)$ be a set of user groups associated with the previously transmitted equations. We also let $z_U(\tau)$ be the total time allocated to equations corresponding to user group U, i.e., $z_U(\tau) = \sum_{\ell \le \ell'} x_U(\ell)$ where ℓ' is the current time interval. The time intervals $\Pi_{1,k}, \dots, \Pi_{\beta_k,k}$ are formed the set of times in

$$\{T_k\} \{T_i + \Delta_i : i \in [k],\ T_i + \Delta_i > T_k\}.$$

As in the offline case in (1), the sets of active users $U_{\ell,k}$, user groups $\mathcal{U}_{\ell,k}$ and $\mathcal{I}_U^{(k)}$ are defined corresponding to these time intervals. Moreover, $\mathcal{V}_k$ is a set of user groups that either already have been transmitted or might be transmitted after $\tau = T_k$. That is $\mathcal{V}_k = \mathcal{U}_{\text{sent}}(\tau) \cup \{\mathcal{U}_{\ell,k} : \ \ell \in \beta_k\}$. The variables $x_U(\ell)$'s and $y_{\{i,f\}}(U)$'s have the same interpretation as the offline case. With these variables, the server solves the following LP.

$$\min \sum_{\ell=1}^{\beta_k} \sum_{U \in \mathcal{U}_{\ell,k}} x_U(\ell) \tag{2}$$

$$\text{s.t. :}$$

$$\sum_{U \in \mathcal{U}_{\ell,k}} x_U(\ell) \le |\Pi_{\ell,k}|, \quad \text{for } \ell = 1, \dots, \beta_k$$

$$\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \le \sum_{\ell \in \mathcal{I}_U^{(k)}} x_U(\ell) + z_U(T_k) \quad \text{for } i \in U,\ U \in \mathcal{V}_k$$

$$\sum_{U \in \mathcal{U}_{\{i,f\}}^{(k)}} y_{\{i,f\}}(U) = r, \quad \text{for } f \in \Omega^{(i)},\ \forall\ i \in \cup_{\ell=1}^{\beta_k} U_{\ell,k},$$

$$x_U(\ell),\ y_{\{i,f\}}(U) \ge 0.$$

An important feature of time intervals $\Pi_{1,k}, \dots, \Pi_{\beta_k,k}$ is that these time intervals end at a deadline and except the first time interval $\Pi_{1,k}$ that starts with arrival time T_k, the other time intervals start with a deadline. Therefore, we have $U_{\ell+1,k} \subset U_{\ell,k}$, i.e., the set of active users in interval $\Pi_{l+1,k}$ is a subset of the active users in interval $\Pi_{l,k}$.

Upon solving the LP in (2), the server makes a decision on the equation that will be transmitted in time slot $[T_k, T_k + 1)$. Towards this end, it creates a list of candidate user groups. Let $\{x_U^*(\ell),\ \forall\ U \in \mathcal{U}_\ell,\ \ell = 1, \dots, \beta_k\}$ be the solution of (2) and that $\mathcal{X}^* = \{x_U^*(\ell) :\ x_U^*(\ell) \ge 1\}$. The elements of $\mathcal{X}^*$ are first ordered based on time intervals. Then, among the elements with the same time interval, they are ordered based on length of user group. Therefore, for two elements $x_U^*(\ell), x_{U'}^*(\ell') \in \mathcal{X}^*$ we say $x_U^*(\ell)$ is before $x_{U'}^*(\ell')$ if $\ell < \ell'$, or if $\ell = \ell'$ and $|U| \ge |U'|$. We let $\mathcal{X}_{\text{sorted}}^*$ to be the sorted version of $\mathcal{X}^*$ in this way. Let $v_i(\tau)$ be the number of missing subfiles that have been transmitted for user i until time τ; this value is tracked in Algorithm 1.

Next, we compute a metric $\eta_U(\tau)$ for each $U \in \mathcal{X}_{\text{sorted}}^*$ that measures the overall benefit of transmitting user group U. Note that user i is missing $\big(|\Omega^{(i)}| - v_i(\tau)\big)$ subfiles and it has $T_i + \Delta_i - \tau$ time slots to obtain them. The ratio of these quantities measures the stringency of user i's deadline. If a user group U is chosen for transmission, it may in general benefit different users differently. For instance, if U has been used for transmission in the past, then the current transmission may be less beneficial to some of the users or of no benefit.

Towards this end, we need a measure of how useful a given U is to a user i where $i \in U$. This can naturally be described in terms of a related LP that we now describe. For each element $x_{\tilde{U}}^*(\ell) \in \mathcal{X}_{\text{sorted}}^*$ let $w_{\{i,\tilde{U}\}}(\tau)$ denote the maximum number of missing subfiles that are recovered by user i at time $\tau + 1$ if user group $\tilde{U}$ was chosen for transmission at time τ. Let us assume that the server chooses $\tilde{U}$ to transmit an equation at the next time slot and let $\tilde{\mathcal{U}}_{\text{sent}}(\tau) = \mathcal{U}_{\text{sent}}(\tau) \cup \{\tilde{U}\}$. Under this assumption we let $\tilde{z}_U(\tau)$, for $U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)$, be the new set of variables $z_U(\tau)$ so that $\tilde{z}_U(\tau) = z_U(\tau)$ for $U \in \mathcal{U}_{\text{sent}}(\tau) \setminus \{\tilde{U}\}$ and if $\tilde{U} \in \mathcal{U}_{\text{sent}}(\tau)$ then $\tilde{z}_{\tilde{U}}(\tau) = z_{\tilde{U}}(\tau) + 1$ otherwise $\tilde{z}_{\tilde{U}}(\tau) = 1$. Consider the set of all user groups, including $\tilde{U}$. For each user group U in this set there are $\tilde{z}_U(\tau)$ time slots available. To compute $w_{\{i,\tilde{U}\}}(\tau)$, we need to find an assignment of the missing subfiles in $\Omega^{(i)}$ to each of these time slots so that number of the recovered missing subfiles of the i-th user is maximized. We let $y_{\{i,f\}}(U)$, for $U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)$ with $U \ni i$, to have the same interpretation as (1). This is equivalent to finding $y_{\{i,f\}}(U)$'s that maximize $\sum_{U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)\ U \ni i} \sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U)$ under the following constraints. Since for each user group $U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)$ there are $\tilde{z}_U(\tau)$ time slots available therefore we have $\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \le \tilde{z}_U(\tau)$. Each missing subfile in $\Omega^{(i)}$

needs r time slots but not all of them might be recoverable. Therefore, we have $\sum_{\mathcal{U}_{\{i,f\}}\cap\tilde{\mathcal{U}}_{\text{sent}}(\tau)} y_{\{i,f\}}(U) \leq r$. Thus, $w_{\{i,\tilde{U}\}}(\tau)$ can be obtained as the solution to the following LP.

$$\max \sum_{U\in\tilde{\mathcal{U}}_{\text{sent}}(\tau), U\ni i}\ \sum_{f\in\mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \tag{3}$$

$$\text{s.t. :}$$

$$\sum_{f\in\mathcal{F}_{\{i,\ U\}}} y_{\{i,f\}}(U) \leq \tilde{z}_U(\tau) \text{ for } U\in\tilde{\mathcal{U}}_{\text{sent}}(\tau), \tilde{U}\ni i$$

$$\sum_{U\in\mathcal{U}_{\{i,f\}}\cap\tilde{\mathcal{U}}_{\text{sent}}(\tau)} y_{\{i,f\}}(U) \leq r \text{ for } f\in\Omega^{(i)},$$

$$y_{\{i,f\}}(U)\geq 0.$$

The metric $\eta_U(\tau)$ is obtained by the following weighted sum.

$$\eta_U(\tau) = \sum_{i\in U} \frac{\left(|\Omega^{(i)}| - v_i(\tau)\right) r}{T_i+\Delta_i-\tau}(w_{\{i,U\}}(\tau) - v_i(\tau)).$$

At time $\tau = T_k$, the server picks the first element $x^*_U(\ell) \in \mathcal{X}^*_{\text{sorted}}$ such that $\eta_U(\tau) \geq \eta_0$ for some threshold η_0 and transmits an equation corresponding to it. Unlike the synchronous case, we need to choose a random linear combination of all missing subfiles of user i that can be transmitted by user group U. Thus, the server transmits

$$\sum_{i\in U}\sum_{f\in\mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}},$$

where m denotes the m-th equation transmitted by the server and $\alpha_{\{i,f,m\}}$ are chosen independently and uniformly at random from the finite field $\mathbb{F}$. If none of the elements in $x^*_U(\ell) \in \mathcal{X}^*_{\text{sorted}}$ satisfy $\eta_U(\ell) \geq \eta_0$ then nothing will be transmitted for the next time slot.

If a new user request does not come at time T_k+1, then the server updates the user group values and then solves (3) again to decide the user group for the time slot $[T_k+1, T_k+2)$. The process continues this way until the next user request comes when the LP in (2) is solved. The complete details are provided in Algorithm 1.

In general, there is no guarantee that Algorithm 1 will return a feasible schedule if the corresponding offline schedule is feasible. In that sense, Algorithm 1 can be viewed as a heuristic with good experimental performance. However, if Algorithm 1 does not return "INFEASIBLE", we can show that a feasible solution for the corresponding offline LP can be identified. This fact coupled with a usage of the Schwartz-Zippel Lemma allows us to conclude that our algorithm works with high probability if it does not return "INFEASIBLE".

Claim 1: For a set of user requests, $\{T_i, \Delta_i, d_i\}$, where $i \in [K]$, if online Algorithm 1 does not return "INFEASIBLE" then there exists a feasible solution for the offline LP in (1).

Proof: We will construct $x_U(\ell)$ and $y_{\{i,f\}}(U)$ variables for the offline LP from the decisions made in Algorithm 1. Note that we update the set $\mathcal{X}_{\text{off}}$ with the user groups chosen in Algorithm 1. It is not difficult to verify that for any $\tilde{x}_U(\ell) \in \mathcal{X}_{\text{off}}$ user group U is a member of $\mathcal{U}_\ell$. Now, for any $U \in \mathcal{U}_\ell$ in (1), we set $x_U(\ell) = \tilde{x}_U(\ell)$ if $\tilde{x}_U(\ell) \in \mathcal{X}_{\text{off}}$ and $x_U(\ell) = 0$ otherwise. Since at each time only one equation in transmitted in Algorithm 1, the first condition $\sum_{U\in\mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|$ holds for all $\ell \in [\beta]$.

For each $i \in [K]$ we define τ_i to be the last time that an equation has been sent to user i. Clearly we have that $v_i(\tau_i+1) \geq r|\Omega^{(i)}|$ otherwise Algorithm 1 will be infeasible at $\tau = T_i + \Delta_i$. We let $U_{i,\text{last}}$ to be the user group associated with this equation and since user i benefits from this equation, we have $i \in U_{i,\text{last}}$. We note that $w_{\{i,U_{i,\text{last}}\}}(\tau_i - 1) = v_i(\tau_i)$.

Note that Algorithm 1, tracks a set $\mathcal{U}_{\text{sent}}$ that contains all the user groups that have been used by the algorithm thus far. In what follows, we use $\mathcal{U}_{\text{sent}}(\tau)$ to denote this set at time τ. We let $\bar{y}_{\{i,f\}}(U)$, $f \in \mathcal{F}_{\{i,U\}}$ and $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ with $U \ni i$, to be the solution of (3) when solving it for $w_{\{i,U_{i,\text{last}}\}}(\tau_i - 1)$. Then, for each $U \in \cup_{\ell=1}^{\beta}\mathcal{U}_\ell$ with $U \ni i$ and for each $f \in \mathcal{F}_{\{i,U\}}$ we assign $y_{\{i,f\}}(U) = \bar{y}_{\{i,f\}}(U)$ if $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ and $y_{\{i,f\}}(U) = 0$ otherwise. We apply this assignment for all $i \in [K]$. With these assignments, we now demonstrate that the second and third conditions in (1) hold.

For the second condition we note that if $U \notin \mathcal{U}_{\text{sent}}(\tau_i)$ then $y_{\{i,f\}}(U) = 0$ and we have nothing to show. For $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ we have that $y_{\{i,f\}}(U) = \bar{y}_{\{i,f\}}(U)$. Recall that $\bar{y}_{\{i,f\}}(U)$ is the solution of (3) at time $\tau = \tau_i - 1$. By the way that $z_U(\tau)$ has been updated in Algorithm 1, we have $z_U(\tau_i) \leq z_U(T_{\max})$. Therefore, we have $z_U(\tau_i) \leq z_U(T_{\max}) = \sum_{\ell\in\mathcal{I}_U} \tilde{x}_U(\ell)$ and from (3) for $w_{\{i,U_{i,\text{last}}\}}(\tau_i)$,

$$\begin{aligned}\sum_{f\in\mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) &= \sum_{f\in\mathcal{F}_{\{i,U\}}} \bar{y}_{\{i,f\}}(U)\\ &\leq \tilde{z}_U(\tau_i - 1) = z_U(\tau_i)\\ &\leq \sum_{\ell\in\mathcal{I}_U} \tilde{x}_U(\ell) = \sum_{\ell\in\mathcal{I}_U} x_U(\ell).\end{aligned}$$

For the third condition, consider any user $i \in [K]$ and any $f \in \Omega^{(i)}$. Recalling the definition of τ_i and $w_{\{i,U_{i,\text{last}}\}}(\tau_i)$. We know that $w_{\{i,U_{i,\text{last}}\}}(\tau_i) = v_i(\tau_i+1) \geq r|\Omega^i|$ which implies that in (3), we have

$$\begin{aligned} r|\Omega^{(i)}| &\leq \sum_{U\in\mathcal{U}_{\text{sent}(\tau_i)}, U\ni i}\ \sum_{f\in\mathcal{F}_{\{i,U\}}} \bar{y}_{\{i,f\}}(U)\\ &= \sum_{f\in\Omega^{(i)}}\ \sum_{U\in\mathcal{U}_{\{i,f\}}\cap\mathcal{U}_{\text{sent}}(\tau_i)} \bar{y}_{\{i,f\}}(U)\\ &\leq \sum_{f\in\Omega^{(i)}} r = r|\Omega^{(i)}|,\end{aligned}$$

where the last inequality comes from the second constraint in (3). The first equality holds by counting arguments for missing subfiles $f \in \Omega^{(i)}$ and user groups in $U \in \mathcal{U}_{\text{sent}}(\tau)$. To verify this, consider a bipartite graph in which the left and right nodes correspond to $f \in \Omega^{(i)}$ and $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ with $U \ni i$ respectively. There is an edge between nodes corresponding to f and U if and only if $f \in \mathcal{F}_{\{i,U\}}$. We let $\bar{y}_{\{i,f\}}(U)$ to be label of this edge. By the definition of $\mathcal{U}_{\{i,f\}}$ we have that $f \in \mathcal{F}_{\{i,U\}}$ implies $U \in \mathcal{U}_{\{i,f\}}$. Therefore, outgoing edges

Algorithm 1 Recursive LP Algorithm

```
Input: Caches $Z_i$ for $i \in [K]$, $\eta_0$, $\{T_i, \Delta_i\}$, for $i \in [K]$.
 1: Initialization:
 2: set $\mathcal{U}_{\text{sent}} \leftarrow \emptyset$, $\mathcal{X}_{\text{off}} \leftarrow \emptyset$, $\ell_{\text{off}} \leftarrow 1$, $m \leftarrow 1$ and $k \leftarrow 1$.
 3: set $\mathcal{M}_i = \emptyset$, and $v_i = 0$ for $i = 1, \ldots, K$.
 4: for $\tau = 0, 1, 2, \ldots, T_{\max}$ do
 5:     if $\tau = T_i + \Delta_i$ for some $i \in [k]$ then
 6:         if $v_i < r|\Omega^{(i)}|$ return INFEASIBLE.
 7:         $\ell_{\text{off}} \leftarrow \ell_{\text{off}} + 1$
 8:     end if
 9:     if $\tau = T_k$ (a new user makes request) then
10:         Solve LP (2), get $\mathcal{X}^*$ then set $\mathcal{X}^*_{\text{sorted}} =$ Sort($\mathcal{X}^*$)
11:         $k \leftarrow k + 1$
12:     end if
13:     if $\mathcal{X}^*_{\text{sorted}} \neq \emptyset$ then
14:         pick first in order $x^*_U(\ell) \in \mathcal{X}^*_{\text{sorted}}$ with $\eta_U(\tau) \geq \eta_0$, $x^*(\ell) \geq 1$ and $\tau \leq \max \Pi_{\ell,k}$.
15:         randomly select $\alpha_{\{i,f,m\}}$'s from $\mathbb{F}$ and send
                $$\sum_{i \in U} \sum_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$$
16:         If $U \in \mathcal{U}_{\text{sent}}$ then $z_U \leftarrow z_U + 1$, otherwise $z_U = 1$ and $\mathcal{U}_{\text{sent}} \leftarrow \mathcal{U}_{\text{sent}} \cup \{U\}$
17:         If $\tilde{x}_U(\ell_{\text{off}}) \in \mathcal{X}_{\text{off}}$ then $\tilde{x}_U(\ell_{\text{off}}) \leftarrow \tilde{x}_U(\ell_{\text{off}}) + 1$, otherwise $\tilde{x}_U(\ell_{\text{off}}) = 1$ and $\mathcal{X}_{\text{off}} \leftarrow \mathcal{X}_{\text{off}} \cup \{\tilde{x}_U(\ell_{\text{off}})\}$
18:         set $x^*_U(\ell) \leftarrow x^*_U(\ell) - 1$, and $z_U \leftarrow z_U + 1$
19:         For all $i \in U$, compute $w_{\{i,U\}}$ and set $v_i \leftarrow w_{\{i,U\}}$, set $\mathcal{M}_i \leftarrow \mathcal{M}_i \cup \{m\}$, then $m \leftarrow m + 1$
20:     end if
21: end for
```

from the node corresponding to f are the edges between f and the nodes $U \in \mathcal{U}_{\{i,f\}} \cap \mathcal{U}_{\text{sent}}(\tau_i)$. Similarly, the outgoing edges between node $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ with $U \ni i$ are the edges between U and $f \in \mathcal{F}_{\{i,U\}}$. By counting $\bar{y}_{\{i,f\}}(U)$ two ways, from the left and right nodes we have the required equality. Therefore, we have that $\sum_{U \in \mathcal{U}_{\{i,f\}} \cap \mathcal{U}_{\text{sent}}} \bar{y}_{\{i,f\}}(U) = r$ for any $f \in \Omega^{(i)}$. This further implies that $\sum_{U \in \mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = r$ for all $f \in \Omega^{(i)}$ and ends the proof.

The following lemma shows that if Algorithm 1 does not return "INFEASIBLE" then with high probability each user is able to recover its missing subfiles from the transmitted equations. Roughly speaking, we set up the a system of equations for each user and claim that under the random choices made by Algorithm 1, each of the system of equations is simultaneously non-singular with high probability. Towards this end, we use Claim 1, to claim that the determinants of the corresponding matrices are not identically zero and then use the Schwartz-Zippel lemma.

Lemma 1: If Algorithm 1 does not return "INFEASIBLE" then with probability at least $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{rKF}$ all requests will be satisfied within their deadline.

Proof: By the way that $\mathcal{M}_i$ and v_i are updated in Algorithm 1, we have $|\mathcal{M}_i| = v_i$. Since the algorithm returns feasible solution thus $v_i = r|\Omega^{(i)}|$ for all $i \in [K]$. Therefore, each user $i \in [K]$ benefits from $r|\Omega^{(i)}|$ equations. For a $m \in \mathcal{M}_i$, let's $\sum_{i \in U} \sum_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$ represents m-th equation. User $i \in U$ can recover $\sum_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$ from this equation since the missing subfiles $W_{\{d_j,f'\}}$, for $f' \in \mathcal{F}_{\{j,U\}}$ and $j \in U \setminus \{i\}$, exist in the cache of user i.

For simplicity, in the discussion below we assume that $r = 1$. The proof for $r > 1$ follows in straightforward manner. For each user $i \in [K]$ we define matrix $\mathbf{B}_i \in \mathbb{F}^{|\Omega^{(i)}| \times |\Omega^{(i)}|}$ whose rows and columns correspond to equation numbers in $\mathcal{M}_i$ and missing subfiles in $\Omega^{(i)}$ respectively. For $m \in \mathcal{M}_i$, assume that m-th equation is associated with user group U, where $i \in U$. Then, the entry of $\mathbf{B}_i$ for the row and column corresponding to $m \in \mathcal{M}_i$ and $f \in \Omega^{(i)}$ is $\alpha_{\{i,f,m\}}$ if $f \in \mathcal{F}_{\{i,U\}}$ and zero otherwise. Recall that user $i \in U$ can recover $\sum_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$ from m-th equation. Therefore, if matrix $\mathbf{B}_i$ is invertible then user i can recover all the missing subfile $W_{\{d_i,f\}}$, for $f \in \Omega^{(i)}$, from equations $\sum_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$ for $m \in \mathcal{M}_i$. Thus, we need to show that determinant of $\mathbf{B}_i$ is nonzero for all $i \in [K]$ with high probability.

Towards this end, let $h_i(\{\alpha_{\{i,f,m\}},\ f \in \Omega^{(i)},\ m \in \mathcal{M}_i\})$ denote the determinant of $\mathbf{B}_i$; we treat the $\{\alpha_{\{i,f,m\}},\ f \in \Omega^{(i)},\ m \in \mathcal{M}_i\}$ as indeterminates at this point. Note that since Algorithm 1 did not return "INFEASIBLE", we have a feasible solution for the corresponding offline LP (*cf.* Claim 1). Therefore, there is a setting for coefficients $\alpha_{\{i,f,m\}}$ with $\alpha_{\{i,f,m\}} \in \{0, 1\}$ such that the multivariate polynomial h_i evaluates to a non-zero value over $\mathbb{F}$, i.e., h_i is not identically zero. This further implies that $h = \prod_{i \in [K]} h_i$ is not identically zero. Now, since each $\alpha_{\{i,f,m\}}$ appears only one time in $\mathbf{B}_i$ thus its degree in polynomial h_i is one. Also, h_i is a polynomial of degree $|\Omega^{(i)}| \leq F$ thus h is a polynomial of degree at most KF. Therefore, we can use Lemma 4 in [6] to show that by choosing $\alpha_{\{i,f,m\}}$'s independently and uniformly at random from $\mathbb{F}$ determinants of $\mathbf{B}_i$'s, $i \in [K]$, are nonzero with probability at least $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{KF}$.

When $r > 1$, we will need to split a missing subfile $W_{\{d_i,f\}}$ in r sub-packets and code over these as well. Thus, the corresponding system of equations will be of size $\mathbb{F}^{r|\Omega^{(i)}| \times r|\Omega^{(i)}|}$ leading to the bound $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{rKF}$.

IV. Simulation Results

We now present simulation results that demonstrate the performance of our algorithm and its comparison with the corresponding offline algorithm and the work of [4]. We note here that the algorithm of [4] works with deadlines for subfiles and does not account for the transmission time for a given packet. We have adapted it for our setting here.

The system we consider has $N = K = 6$ and $M = 2$ with the placement scheme of [1]. The arrival times $\{T_i,\ i \in [K]\}$ are generated according to a Poisson process with parameter λ. In our simulations, we generate Δ_i, $i \in [K]$, from a uniform distribution between $[\Delta_{\min}, \Delta_{\max}]$. We set $\Delta_{\min} = (KM/N)F$ and $\Delta_{\max} = KF$.

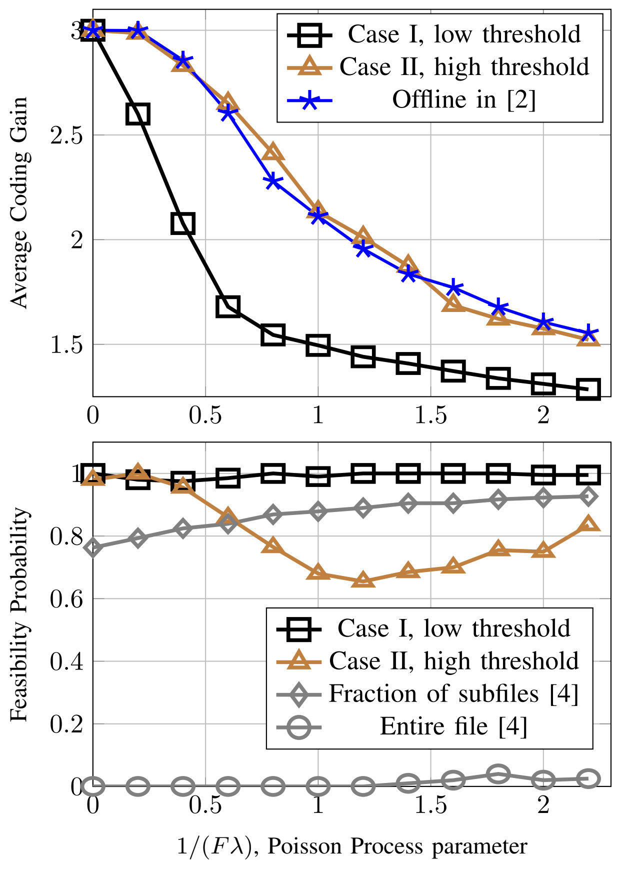

Fig. 2: Centralized Placement in [1]: (a) average Coding Gain over all feasible offline problem instances, (b) feasibility probability of the online algorithm conditioned on feasibility of the offline problem. The placement has been fixed for all trials and at each trial a new arrival time and deadline is generated. In this simulation, we set $\eta_0 = 0.4 - \frac{0.5}{\lambda}$ and $\eta_0 = 0.8 - \frac{0.2}{\lambda}$ in Case I and II respectively.

For each set of generated arrivals, we first run the offline LP to check whether it is feasible. The proposed online algorithm is run only if the offline LP is feasible. We run the online algorithm with a low threshold (case I) and a high threshold (case II). The coding gain is defined as the ratio of the rate achieved by the system to the uncoded rate. Fig. 2 (a) depicts a plot of the coding gain vs. $1/(F\lambda)$. As λ decreases the arrivals are spaced further apart on average, and the coding gain of any scheme is expected to reduce. The coding gain is computed by taking an average overall all instances where a given scheme is feasible. For the offline scheme this means that we take the average of all instances where it is feasible. For the case II of the online algorithm, some of arrival patterns may result in infeasibility; these instances were not taken into account when computing the average coding gain. This explains why the coding gain of case II sometimes appears to be higher than the offline algorithm. However, the coding gain of case I is significantly lower, because of its low threshold.

The feasibility probability of a scheme vs. the arrival rate is plotted in Fig. 2 (b). As expected the low threshold online algorithm has a very high feasibility probability ≈ 1 for a range of arrival parameters. On the other hand, the higher threshold of case II, affects its feasibility probability when the arrival rates are lower as it tends to wait for larger user groups before transmitting.

For both plots, we also include the results of [4]. Note that the heuristic in [4] attempts a best effort delivery based on the deadline of each subfile. In our system the deadline for all subfiles of a given user are the same. The feasibility probability of [4] is quite poor. Accordingly we also plot the fraction of subfiles that meet the deadline; this is somewhat better. The coding gain numbers for [4] are also quite unreliable as the algorithm is infeasible in most cases. Thus, we do not plot it.

References

[1] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. on Info. Th.*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[2] H. Ghasemi and A. Ramamoorthy, "Asynchronous coded caching," *IEEE Intl. Symposium on Info. Th.*, 2017.

[3] ——, "Algorithms for asynchronous coded caching," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct 2017, pp. 636–640.

[4] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *IEEE Intl. Conf. Comm.*, 2015, pp. 5559–5564.

[5] Q. Yang, M. M. Amiri, and D. Gündüz, "Audience-retention-rate-aware caching and coded video delivery with asynchronous demands," *arXiv preprint arXiv:1808.04835*, 2018.

[6] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. on Info. Th.*, vol. 52, no. 10, pp. 4413–4430, 2006.