Cache Freshness in Information Updating Systems

Melih Bastopcu Sennur Ulukus
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
bastopcu@umd.edu *ulukus@umd.edu*

***Abstract*—We consider a cache updating system with a source, m caches and a user. There are n files. The source keeps the freshest version of the files which are updated with known rates λ_i. The first cache downloads and keeps the freshest version of the files from the source with rates c_{1i} and cache r downloads and keeps the files from cache $r-1$ with rates c_{ri} for $r>1$. The user gets updates from cache m with rates u_i. When the user gets an update, it either gets a fresh update from cache m or the file at cache m becomes outdated by a file update at the source. We find an analytical expression for the average freshness of the files at the user. We provide an alternating maximization based method to find the update rates for the caches, c_{ri}, and for the user, u_i, to maximize the freshness of the files at the user. We note that for a given set of update rates for the user (resp. for the caches), the optimal rate allocation policy for the caches (resp. for the user) is a *threshold policy*, where the optimal update rates for rapidly changing files at the source may be equal to zero.**

I. Introduction

With emerging technologies such as autonomous driving, augmented reality, social networking, high-frequency automated trading, and online gaming, time sensitive information has become ever more critical. Age of information has been proposed as a performance metric to quantify the *freshness* of information in communication networks. Age of information has been studied in the context of web crawling, queueing networks, caching systems, remote estimation, energy harvesting systems, scheduling in networks, and so on [1]–[18].

In this work, we consider a cache updating system that consists of a source, several caches and a user as shown in Fig. 2. The models we study are abstractions of a real-life setting shown in Fig. 1. Specifically, the two-hop serial cache system in Fig. 2 when $m=1$ is an abstraction for the communication system between the cloud, macro base station and user A in Fig. 1; the multi-hop serial cache system in Fig. 2 when $m=2$ is an abstraction for the communication system between the cloud, macro base station, small-cell base station and user B in Fig. 1 (for a three-hop system).

In these system models, the source keeps the freshest version of all the files which are updated with known rates λ_i. The first cache (resp. cache r for $r>1$) downloads the files from the source (resp. from cache $r-1$) and stores the latest downloaded versions of these files. When the first cache downloads a file from the source, the file at the first cache becomes fresh. When cache r downloads a file from cache $r-1$, either cache r gets the fresh file from cache $r-1$ or the file at cache $r-1$ becomes outdated due to a file update at the source. Thus, depending on the file status at cache $r-1$, cache r may get a fresh or an outdated file. This is valid for the user as well, i.e., after cache m obtains a fresh file, either the user gets the fresh file from cache m or the file at cache m becomes outdated. For this system model, we derive an analytical expression for the information freshness at the end-user, and determine the updating frequencies for the intermediate caches and the end-user for maximum freshness at the end-user.

This work was supported by NSF Grants CCF 17-13977 and ECCS 18-07348.

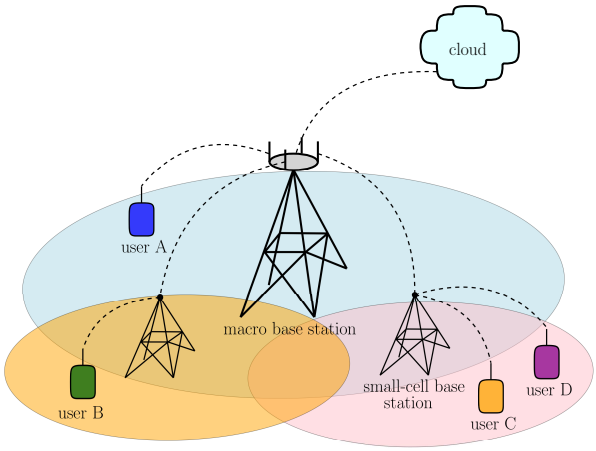


Fig. 1. A cache updating system with a cloud (the source), a macro base station (the first cache), a small-cell base station (the second cache), users.

References that are most closely related to our work are [2] and [7]. Reference [2] studies the problem of finding optimal crawl rates to keep the information in a search engine fresh while maintaining the constraints on crawling rates imposed by the websites and also the total crawl rate constraint of the search engine. Even though the freshness metric used in [2] is similar to ours, the problem settings are different, as here we develop a general freshness expression for a multi-hop caching system, which differentiates our overall work from [2]. Reference [7] considers a similar model to ours where a resource constrained remote server wants to keep the items at a local cache as fresh as possible. Reference [7] shows that the update rates of the files should be chosen proportional to the square roots of their popularity indices. Different from [7] where the freshness of the local cache is considered, we consider the freshness at the user which is connected to the source via multiple caches. Thus, our system model can be thought of as an extended version of the one-hop model in [7]. However, our freshness metric is different than the traditional

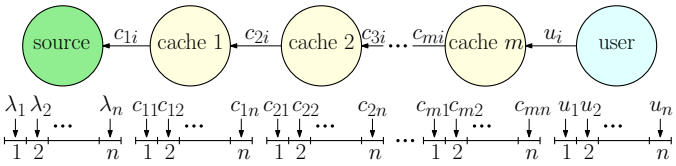


Fig. 2. A cache updating system where there are m serially connected caches in between the source and the user.

age metric used in [7], and hence, our overall work is distinct compared to [7].

In this paper, we consider a system where there is a source, caches and a user (Fig. 2). We find an analytical expression for the average freshness of the files at the user. We impose total update rate constraints for the caches and also for the user due to limited nature of resources. Our aim is to find the update rates for the caches and also for the user such that the total freshness of the files at the user is maximized. We provide an alternating maximization based solution where the update rates of the user (resp. of the caches) are optimized for a given set of update rates of the caches (resp. of the user). We observe that for a given set of parameters, such as update rates of the user, the optimal rate allocation policy for the other set of parameters, such as update rates at the caches, is a *threshold policy*, where the files that are refreshed frequently at the source may not be updated by the corresponding entity.

II. System Model

We consider a cache updating system where there is an information source, m caches and a user as shown in Fig. 2. The information source keeps the freshest version of n files where the ith file is updated (refreshed) with exponential inter-arrival times with rate λ_i. The file updates at the source are independent of each other. Caches are capable of storing the latest downloaded versions of all files. The first cache gets the fresh files from the source, and cache r gets files from cache $r-1$ for $r > 1$. The channels are assumed to be perfect and the transmission times are negligible. Thus, if cache r requests an update for the ith file, it receives the file from cache $r-1$ (or from the source if $r = 1$) right away. The inter-update request times of cache r for the ith file are exponential with rate c_{ri}. Each cache is subject to a total update rate constraint as in [7] as it is resource-constrained, i.e., $\sum_{i=1}^{n} c_{ri} \leq C_r$. The user requests files from cache m. The inter-update request times of the user for the ith file are exponential with rate u_i. Similarly, the channel between the user and cache m is perfect and the transmission times are negligible. There is a total update rate constraint for the user, i.e., $\sum_{i=1}^{n} u_i \leq U$.

We note that each file at the source is always *fresh*. However, when a file is updated at the source, the stored versions of the same file at the caches and at the user become *outdated*. When the first cache gets an update for an outdated file from the source, the updated file in the first cache becomes *fresh* again until the next update arrival at the source. When cache r requests an update for an outdated file, it might still receive an outdated version if the file at cache $r-1$ is not fresh. This is valid for the user as well. We note that since the caches and

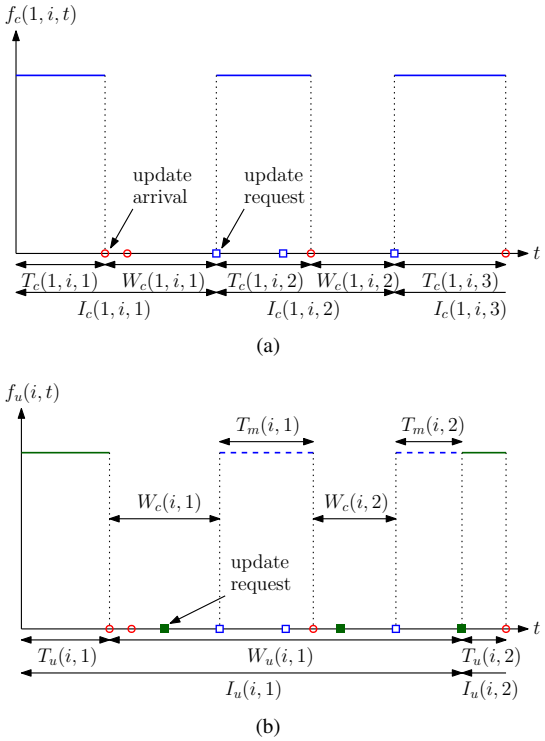


Fig. 3. Sample evolution of freshness of the ith file (a) at the cache and (b) at the user for a cache updating system with a source, a single cache and a user. Red circles represent the update arrivals at the source, blue squares represent the update requests from the cache, and green filled squares represent the update requests from the user.

the user are unaware of the file updates at the source, they do not know whether they have the freshest versions of the files or not. Thus, they may still unknowingly request an update even though they have the freshest version of a file.

In order to quantify the *freshness*, we define the freshness function of file i at cache r $f_c(r,i,t)$ shown in Fig. 3(a) as,

$$f_c(r,i,t) = \begin{cases} 1, & \text{if file } i \text{ at cache } r \text{ is fresh at time } t, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

i.e., the instantaneous freshness function is a binary function taking values of fresh, "1", or not fresh, "0", at any time t.

We denote file updates which replace an outdated file with the freshest version of the file as *successful* updates. We define the time interval between the jth and the $(j+1)$th successful updates for the ith file at cache r as the jth update cycle and denote it by $I_c(r,i,j)$. We denote the time duration when the ith file at cache r is fresh during the jth update cycle as $T_c(r,i,j)$. Then, we define the long term average freshness of the ith file at cache r as

$$F_c(r,i) = \lim_{T\to\infty} \frac{1}{T} \int_0^T f_c(r,i,t)dt. \quad (2)$$

Similar to [1], $F_c(r,i)$ is equal to

$$F_c(r,i) = \lim_{T\to\infty} \frac{N}{T} \left(\frac{1}{N} \sum_{j=1}^{N} T_c(i,j) \right) = \frac{\mathbb{E}[T_c(r,i)]}{\mathbb{E}[I_c(r,i)]}, \quad (3)$$

where N is the number of update cycles in time duration T.

We define the time interval between the jth and the $(j+1)$th successful updates for the ith file at the user as the jth update cycle and denote it by $I_u(i,j)$. We denote the time duration when the ith file at the user is fresh during the jth update cycle as $T_u(i,j)$. Similarly, we define $f_u(i,t)$ as the freshness function of the ith file at the user shown in Fig. 3(b). Then, the long term average freshness of the ith file at the user becomes

$$F_u(i) = \frac{\mathbb{E}[T_u(i)]}{\mathbb{E}[I_u(i)]}. \tag{4}$$

We define the total freshness over all files at the user F_u as

$$F_u = \sum_{i=1}^{n} F_u(i). \tag{5}$$

Our aim is to find the optimal update rates for the caches, c_{ri}, and for the user, u_i, such that the total average freshness of the user F_u is maximized while satisfying the constraints on the total update rates for caches, $\sum_{i=1}^{n} c_{ri} \leq C_r$, and for the user, $\sum_{i=1}^{n} u_i \leq U$. Thus, our optimization problem is,

$$\begin{aligned} \max_{\{c_{ri},u_i\}} \quad & F_u \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_{ri} \leq C_r, \quad r = 1,\dots,m \\ & \sum_{i=1}^{n} u_i \leq U \\ & c_{ri} \geq 0, \quad u_i \geq 0. \end{aligned} \tag{6}$$

In the following section, we find analytical expressions for the long term average freshness of the ith file at the caches, $F_c(r,i)$, and at the user, $F_u(i)$, as a function of the update rate at the source λ_i, the update rates at the caches c_{ri}, and the update rate at the user u_i. Once we find $F_u(i)$, this will determine the objective function of (6) via (5).

III. Average Freshness Analysis

We consider a system where there are m caches placed in between the source and the user, as shown in Fig. 2. We note that due to the memoryless property of the exponential distribution, $T_c(r,i,j)$ which is the time duration when the ith file at cache r is fresh during the jth update cycle is exponentially distributed with parameter λ_i. Since $T_c(r,i,j)$ are independent and identically distributed (i.i.d.) over j, we drop index j, and denote a typical $T_c(r,i,j)$ as $T_c(r,i)$. Thus, we have $\mathbb{E}[T_c(r,i)] = \frac{1}{\lambda_i}$. Let $W_c(r,i,j)$ be the total duration when the ith file at cache r is outdated during the jth update cycle, i.e., $W_c(r,i,j) = I_c(r,i,j) - T_c(r,i,j)$. Note that $W_c(r,i,j)$ is also equal to the time passed until the fresh version of the ith file is obtained after the file is outdated at the cache. We denote typical random variables for $W_c(r,i,j)$ and $I_c(r,i,j)$ by $W_c(r,i)$ and $I_c(r,i)$, respectively.

Next, we find an analytical expression for the average freshness of the ith file at the rth cache $F_c(r,i)$ and at the user $F_u(i)$. We note that the first cache always obtains a fresh file from the source. Thus, as the update request times for the first cache are exponentially distributed with rate c_{1i}, we have $\mathbb{E}[W_c(1,i)] = \frac{1}{c_{1i}}$. In order to obtain a fresh file at cache r, the file at cache $r-1$ needs to be fresh for $r > 1$ which takes $W_c(r-1,i,j)$ time as defined earlier. After the file at cache $r-1$ becomes fresh, either cache r gets the fresh update from cache $r-1$ or the file at the source is updated, and thus the files at all caches become outdated again. We denote the earliest time that one of these two cases happens as $T_m(r,i)$, i.e., $T_m(r,i) = \min\{T_c(r-1,i), \bar{W}_c(r,i)\}$ where $\bar{W}_c(r,i)$ is the time for cache r to obtain a new update from cache $r-1$ which is exponentially distributed with rate c_{ri}. Thus, $T_m(r,i)$ is also exponentially distributed with rate $c_{ri}+\lambda_i$. We note that $\mathbb{P}[T_m(r,i) = \bar{W}_c(r,i)] = \frac{c_{ri}}{c_{ri}+\lambda_i}$ and $\mathbb{P}[T_m(r,i) = T_c(r-1,i)] = \frac{\lambda_i}{c_{ri}+\lambda_i}$.

Note that if cache r gets the fresh update before the file at cache $r-1$ becomes outdated which happens with probability $\mathbb{P}[T_m(r,i) = \bar{W}_c(r,i)]$, an update cycle of the ith file at cache r is completed, and thus $f_c(r,i,t)$ is equal to 1 again. However, if the file at the source is updated before cache r gets the fresh update from cache $r-1$, then this process repeats itself, i.e., cache $r-1$ initially needs to receive a fresh update which takes another $W_c(r-1,i,j)$ time and so on, until cache r receives a fresh update from cache $r-1$. Thus, we have

$$W_c(r,i,j) = \sum_{\ell=1}^{K_r} W_c(r-1,i,\ell) + T_m(r,i,\ell), \tag{7}$$

where K_r is a geometric random variable with rate $\frac{c_{ri}}{c_{ri}+\lambda_i}$. Then, given that $K_r = k$, we write $\mathbb{E}[W_c(r,i)|K_r = k]$ as

$$\mathbb{E}[W_c(r,i)|K_r = k] = k\mathbb{E}[W_c(r-1,i)] + \frac{k}{c_{ri}+\lambda_i}. \tag{8}$$

Then, we find $\mathbb{E}[W_c(r,i)] = \mathbb{E}\left[\mathbb{E}\left[W_c(r,i)|K_r\right]\right]$ as

$$\mathbb{E}[W_c(r,i)] = \frac{c_{ri}+\lambda_i}{c_{ri}}\mathbb{E}[W_c(r-1,i)] + \frac{1}{c_{ri}}, \tag{9}$$

which is equal to $\mathbb{E}[W_c(1,i)] = \frac{1}{c_{1i}}$ if $r = 1$, and to

$$\mathbb{E}[W_c(r,i)] = \frac{1}{c_{ri}} + \sum_{\ell=1}^{r-1} \frac{1}{c_{\ell i}} \prod_{p=\ell+1}^{r} \frac{c_{pi}+\lambda_i}{c_{pi}}, \quad r = 2,\dots,m. \tag{10}$$

Then, by using $\mathbb{E}[I_c(r,i)] = \mathbb{E}[T_c(r,i)] + \mathbb{E}[W_c(r,i)]$, we have

$$\mathbb{E}[I_c(r,i)] = \begin{cases} \frac{1}{\lambda_i} + \frac{1}{c_{1i}}, & r = 1, \\ \left(\frac{1}{\lambda_i} + \frac{1}{c_{1i}}\right) \prod_{\ell=2}^{r} \frac{c_{\ell i}+\lambda_i}{c_{\ell i}}, & 2 \leq r \leq m. \end{cases} \tag{11}$$

Thus, the average freshness for the ith file at cache r is

$$F_c(r,i) = \frac{\mathbb{E}[T_c(r,i)]}{\mathbb{E}[I_c(r,i)]} = \prod_{\ell=1}^{r} \frac{c_{\ell i}}{c_{\ell i}+\lambda_i}, \quad r = 1,\dots,m. \tag{12}$$

We use $I_u(i)$ to denote the typical random variable for $I_u(i,j)$. Similar to $\mathbb{E}[I_c(r,i)]$, we find $\mathbb{E}[I_u(i)]$ as

$$\mathbb{E}[I_u(i)] = \left(\frac{1}{\lambda_i} + \frac{1}{c_{1i}}\right) \frac{u_i+\lambda_i}{u_i} \prod_{r=2}^{m} \frac{c_{ri}+\lambda_i}{c_{ri}}. \tag{13}$$

Then, the average freshness of the ith file at the user is

$$F_u(i) = \frac{\mathbb{E}[T_u(i)]}{\mathbb{E}[I_u(i)]} = \frac{u_i}{u_i+\lambda_i}\prod_{r=1}^{m}\frac{c_{ri}}{c_{ri}+\lambda_i}. \tag{14}$$

To give a specific example, for $m = 2$, we have

$$F_u(i) = \frac{u_i}{u_i+\lambda_i}\frac{c_{1i}}{c_{1i}+\lambda_i}\frac{c_{2i}}{c_{2i}+\lambda_i}. \tag{15}$$

The freshness of the ith file at the user $F_u(i)$ in (14) depends not only on the update rate of the user u_i and file update rate at the source λ_i but also the update rates of all the caches c_{ri} as the user obtains the fresh files through caches. We note that $F_u(i)$ is an increasing function of u_i and c_{ri}, but a decreasing function of λ_i. We observe that $F_u(i)$ is an individually concave function of u_i and c_{ri} but not jointly concave in u_i and c_{ri}, as u_i and c_{ri} terms appear as a multiplication in (14). From (12), the average freshness experienced by cache r for $r > 1$ is equal to the multiplication of the freshness of cache $r-1$ with the freshness of cache r when cache r is directly connected to the source, i.e., $\frac{c_{ri}}{c_{ri}+\lambda_i}$. We observe from (14) that this is also valid for the freshness of the user. As $\frac{c_{ri}}{c_{ri}+\lambda_i} < 1$, the freshness of the user when connected to the source via caches is smaller than the freshness it would achieve if it was connected to the source directly.

In the following section, we solve the optimization problem in (6) by using the freshness expression $F_u(i)$ found in (14).

IV. Freshness Maximization

In this section, we consider the optimization problem in (6) for the system in Fig. 2. Using $F_u(i)$ in (14) and F_u in (5), we rewrite the freshness maximization problem as

$$\begin{aligned}\max_{\{c_{ri},u_i\}} \quad & \sum_{i=1}^{n}\frac{u_i}{u_i+\lambda_i}\prod_{r=1}^{m}\frac{c_{ri}}{c_{ri}+\lambda_i} \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_{ri} \le C_r, \quad r = 1,\dots,m \\ & \sum_{i=1}^{n} u_i \le U \\ & c_{ri} \ge 0, \quad u_i \ge 0. \end{aligned} \tag{16}$$

We introduce the Lagrangian function [19] for (16) as

$$\begin{aligned}\mathcal{L} = &-\sum_{i=1}^{n}\frac{u_i}{u_i+\lambda_i}\prod_{r=1}^{m}\frac{c_{ri}}{c_{ri}+\lambda_i} + \sum_{r=1}^{m}\beta_r\left(\sum_{i=1}^{n}c_{ri} - C_r\right) \\ &+ \theta\left(\sum_{i=1}^{n}u_i - U\right) - \sum_{r=1}^{m}\sum_{i=1}^{n}\nu_{ri}c_{ri} - \sum_{i=1}^{n}\eta_i u_i, \end{aligned} \tag{17}$$

where $\beta_r \ge 0$, $\theta \ge 0$, $\nu_{ri} \ge 0$ and $\eta_i \ge 0$. Then, we write the KKT conditions as

$$\frac{\partial\mathcal{L}}{\partial c_{ri}} = -\frac{u_i}{u_i+\lambda_i}\prod_{\ell\neq r}\frac{c_{\ell i}}{c_{\ell i}+\lambda_i}\frac{\lambda_i}{(c_{ri}+\lambda_i)^2} + \beta_r - \nu_{ri} = 0, \tag{18}$$

$$\frac{\partial\mathcal{L}}{\partial u_i} = -\frac{\lambda_i}{(u_i+\lambda_i)^2}\prod_{r=1}^{m}\frac{c_{ri}}{c_{ri}+\lambda_i} + \theta - \eta_i = 0, \tag{19}$$

for all r and i. The complementary slackness conditions are

$$\beta_r\left(\sum_{i=1}^{n}c_{ri} - C_r\right) = 0, \tag{20}$$

$$\theta\left(\sum_{i=1}^{n}u_i - U\right) = 0, \tag{21}$$

$$\nu_{ri}c_{ri} = 0, \tag{22}$$

$$\eta_i u_i = 0. \tag{23}$$

The objective function in (16) is not jointly concave in c_{ri} and u_i. However, for given c_{ri}s, the objective function in (16) is concave in u_i, and for a given u_i and $c_{\ell i}$s for all $\ell \neq r$, the objective function in (16) is concave in c_{ri}. Thus, we apply an alternating maximization based method [20] to find $(c_{1i},\dots,c_{ri},u_i)$ tuples such that (18) and (19) are satisfied for all r and i. Starting with initial u_i and $c_{\ell i}$s for $\ell \neq r$, we find the optimum update rates for cache r, c_{ri}s, such that the total update rate constraint for cache r, i.e., $\sum_{i=1}^{n}c_{ri} \le C_r$, and the feasibility of the update rates, i.e., $c_{ri} \ge 0$ for all i, are satisfied. We repeat this step for all r. Then, for given c_{ri}s, we find the optimum update rates for the user, u_is, such that the total update rate constraint for the user, i.e., $\sum_{i=1}^{n}u_i \le U$, and the feasibility of the update rates, i.e., $u_i \ge 0$ for all i, are satisfied. We repeat these steps until the KKT conditions in (18) and (19) are satisfied.

For given u_is with $u_i > 0$, and $c_{\ell i}$s with $c_{\ell i} > 0$, for $\ell \neq r$, we rewrite (18) as

$$(c_{ri}+\lambda_i)^2 = \frac{\sigma_i\lambda_i}{\beta_r - \nu_{ri}}, \tag{24}$$

where $\sigma_i = \frac{u_i}{u_i+\lambda_i}\prod_{\ell\neq r}\frac{c_{\ell i}}{c_{\ell i}+\lambda_i}$. If $c_{ri} > 0$, we have $\nu_{ri} = 0$ from (22). Thus, we have

$$c_{ri} = \left(\sqrt{\frac{\sigma_i\lambda_i}{\beta_r}} - \lambda_i\right)^+, \tag{25}$$

for all i with $u_i > 0$ and $c_{\ell i} > 0$ for $\ell \neq r$, where $(x)^+ = \max(x,0)$. Note that $c_{ri} > 0$ requires $\frac{\sigma_i}{\lambda_i} > \beta_r$, i.e., $\frac{1}{\lambda_i}\frac{u_i}{u_i+\lambda_i}\prod_{\ell\neq r}\frac{c_{\ell i}}{c_{\ell i}+\lambda_i} > \beta_r$ which also implies that, if $\frac{1}{\lambda_i}\frac{u_i}{u_i+\lambda_i}\prod_{\ell\neq r}\frac{c_{\ell i}}{c_{\ell i}+\lambda_i} \le \beta_r$, then we must have $c_{ri} = 0$. Thus, for given u_is and $c_{\ell i}$s for $\ell \neq r$, we see that the optimal rate allocation policy for cache r is a *threshold policy* in which the optimal update rates are equal to zero when the files are refreshed too frequently at the source, i.e., when the corresponding λ_is are too large. We repeat this step for all r.

Next, we solve for u_is for given c_{ri}s for all r with $c_{ri} > 0$. We rewrite (19) as

$$(u_i+\lambda_i)^2 = \frac{\rho_i\lambda_i}{\theta - \eta_i}, \tag{26}$$

where $\rho_i = \prod_{r=1}^{m}\frac{c_{ri}}{c_{ri}+\lambda_i}$. If $u_i > 0$, we have $\eta_i = 0$ from (23). Thus, for all i with $c_{ri} > 0$ for all r, we have

$$u_i = \left(\sqrt{\frac{\rho_i\lambda_i}{\theta}} - \lambda_i\right)^+. \tag{27}$$

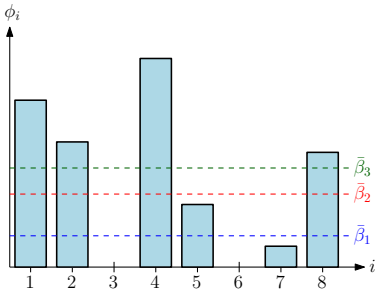
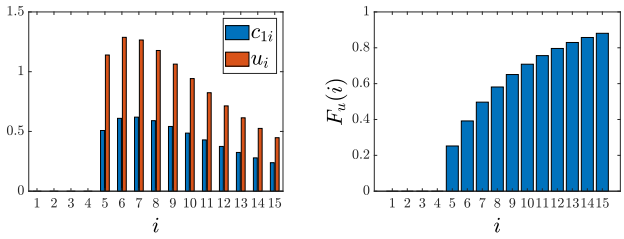


Fig. 4. For given u_is and $c_{\ell i}$s for $\ell \neq r$, we show ϕ_is in (28) for $n = 8$.



(a) (b)

Fig. 5. A single cache system, i.e., $m = 1$. (a) Update rate allocation for the cache and the user for each file, and (b) the corresponding freshness $F_u(i)$.

Similarly, $u_i > 0$ requires $\frac{\rho_i}{\lambda_i} > \theta$, i.e., $\frac{1}{\lambda_i}\prod_{r=1}^{m}\frac{c_{ri}}{c_{ri}+\lambda_i} > \theta$ which implies that if $\frac{1}{\lambda_i}\prod_{r=1}^{m}\frac{c_{ri}}{c_{ri}+\lambda_i} \leq \theta$, then we must have $u_i = 0$. Thus, for given c_{ri}s, the optimal rate allocation policy for the user is a threshold policy and the update rates for the user are equal to zero for the files with very large λ_is.

One can show that in the optimal policy, if $c_{ri} = 0$ for some r, then we must have $c_{\ell i} = 0$ for all $\ell \neq r$ and $u_i = 0$. Similarly, if $u_i = 0$ for some i, then we must have $c_{ri} = 0$ for all r. These are true since c_{ri} and u_i come as multiplicative terms in the objective function of (16); see [21]. One can also show that the total update rate constraints for cache r, i.e., $\sum_{i=1}^{n} c_{ri} \leq C_r$, for all r and for the user, i.e., $\sum_{i=1}^{n} u_i \leq U$, should be satisfied with equality as the objective function in (16) is an increasing function of c_{ri} and u_i. Thus, in the optimal policy, we have $\sum_{i=1}^{n} c_{ri} = C_r$ for all r and $\sum_{i=1}^{n} u_i = U$; see [21].

In the following lemma, we identify a property of the optimal update rates for cache r c_{ri} for given update rates of the user u_i and of the remaining caches $c_{\ell i}$ for all $\ell \neq r$. To that end, let us define ϕ_i as

$$\phi_i = \frac{\sigma_i}{\lambda_i}, \quad i = 1, \ldots, n. \tag{28}$$

This lemma is used to solve for c_{ri} given u_i and $c_{\ell i}$ for $\ell \neq r$.

Lemma 1 *For given u_is and $c_{\ell i}$s for $\ell \neq r$, if $c_{ri} > 0$ for some i, then we have $c_{rj} > 0$ for all j with $\phi_j \geq \phi_i$.*

Proof: As we noted earlier, from (25), $c_{ri} > 0$ implies $\phi_i > \beta_r$. Thus, if $\phi_j \geq \phi_i$, then we have $\phi_j > \beta_r$, which further implies $c_{rj} > 0$. ■

Next, we describe the overall solution. We start with a set of initial u_is and $c_{\ell i}$s for $\ell \neq r$. We obtain ϕ_i from u_i and $c_{\ell i}$s for $\ell \neq r$ using (28). We will utilize Fig. 4 to describe the steps of the solution visually. We plot ϕ_i in Fig. 4. Note that if $u_i = 0$ or $c_{\ell i}$s for $\ell \neq r$ for some ℓ then $\phi_i = 0$, and vice versa. First, we choose $c_{ri} = 0$ for the files with $u_i = 0$ or $c_{\ell i} = 0$ for some ℓ as discussed above, i.e., in Fig. 4, we choose c_{r3} and c_{r6} as zero. Next, we find the remaining c_{ri}s with $u_i > 0$ and $c_{\ell i} > 0$ for all $\ell \neq r$. We rewrite (25) as

$$c_{ri} = \frac{\lambda_i}{\sqrt{\beta_r}}\left(\sqrt{\phi_i} - \sqrt{\beta_r}\right)^+ \tag{29}$$

As we discussed earlier, in the optimal policy, we must have $\sum_{i=1}^{n} c_{ri} = C_r$. Assuming that $\phi_i \geq \beta_r$ for all i, i.e., by ignoring $(\cdot)^+$ in (25) and (29), we solve $\sum_{i=1}^{n} c_{ri} = C_r$ for β_r. Then, we compare the smallest ϕ_i with β_r. If the smallest ϕ_i is larger than or equal to β_r, it implies that $c_{ri} > 0$ for all i due to Lemma 1, and we have obtained the optimal c_{ri} values for given u_is and $c_{\ell i}$s for all $\ell \neq r$. If the smallest ϕ_i is smaller than β_r, it implies that the corresponding c_{ri} was negative and it must be chosen as zero. In this case, we choose $c_{ri} = 0$ for the smallest ϕ_i. In the example in Fig. 4, if the β_r we found is $\bar{\beta}_1$, then since $\phi_7 < \bar{\beta}_1$ we choose c_{r7} as zero. Then, we repeat this process again until the smallest ϕ_i among the remaining c_{ri}s satisfies $\phi_i \geq \beta_r$. For example, in Fig. 4, the next β_r found by using only indices $1, 2, 4, 5, 8$ may be $\bar{\beta}_2$. Since $\phi_5 < \bar{\beta}_2$, we choose $c_{r5} = 0$. In the next iteration, the β_r found by using indices $1, 2, 4, 8$ may be $\bar{\beta}_3$. Since $\phi_8 > \bar{\beta}_3$, we stop the process and find c_{ri} for $i = 1, 2, 4, 8$ from (25) or (29) by using $\bar{\beta}_3$ in Fig. 4. This concludes finding c_{ri}s for given u_is and $c_{\ell i}$s for all $\ell \neq r$. We repeat this step for $r = 1, \ldots, m$. Next, for given c_{ri}s, we find u_is by following a similar procedure. We keep updating these parameters until $(c_{1i}, \ldots, c_{ri}, u_i)$ tuples converge.

V. Numerical Results

In this section, we provide three numerical results. For these results, we use the following update arrival rates at the source

$$\lambda_i = bq^i, \quad i = 1, \ldots, n, \tag{30}$$

where $b > 0$ and $0 < q \leq 1$ such that $\sum_{i=1}^{n} \lambda_i = a$. With the update arrival rates in (30), we have $\lambda_i \geq \lambda_j$ for $i \leq j$.

In the first example, we take $a = 10$, $q = 0.7$, and $n = 15$ in (30). For this example, we consider the system with a single cache, i.e., $m = 1$. We choose the total update rate constraint for the cache as $C_1 = 5$, and for the user as $U = 10$. We apply the alternating maximization method in Section IV to find the update rates for the cache and for the user. We see in Fig. 5(a) that the first four files which are refreshed most frequently at the source are not updated by the cache and the user. As these files change too frequently at the source, their stored versions at the user become obsolete very quickly. Even though the update rates of the cache and the user for the slowly changing files are low, the freshness of the slowly varying files is higher compared to the rapidly changing files as shown in Fig. 5(b).

In the second example, we consider the same system as in the first example but this time we examine the effect of

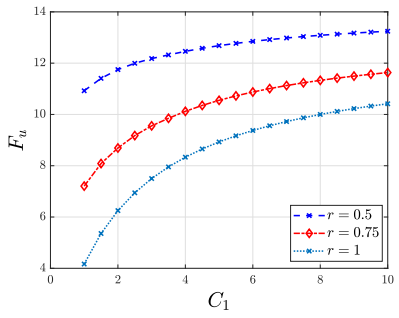


Fig. 6. A single cache system, $m = 1$. Total freshness of the user with respect to C, with λ_i given in (30), for $a = 2$, $q = 0.5, 0.75, 1$ and $n = 15$.

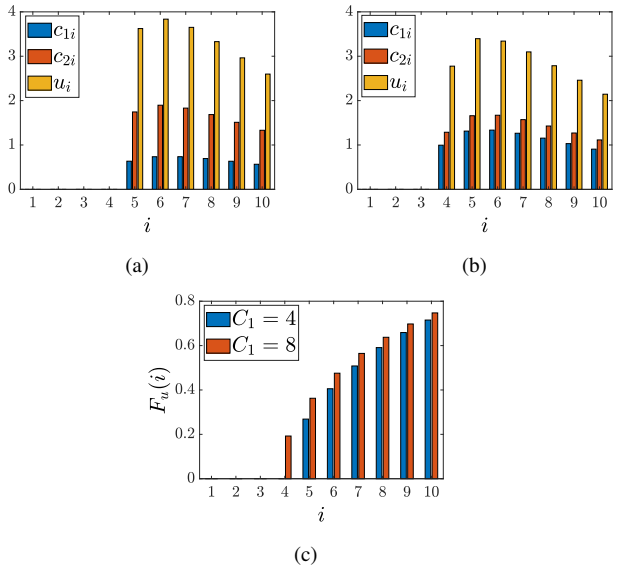


Fig. 7. A two-cache system, i.e., $m = 2$. The update rates of the caches and the user when the total update rate constraint for the user is $U = 20$, for the second cache is $C_2 = 10$ and for the first cache is (a) $C_1 = 4$, and (b) $C_1 = 8$. The freshness of the files at the user is shown in (c).

file update rates at the source over the information freshness at the user. We take λ_i in (30) with $a = 2$, $n = 15$ for $q = 0.5, 0.75, 1$. We note that a smaller q corresponds to a less even (more polarized) distribution of file change rates at the source. We choose the total update rate for the user as $U = 10$ and vary the total update rate for the cache as $C_1 = 1, 1.5, \ldots, 10$. We see in Fig. 6 that when the distribution of the change rates of the files are more polarized, i.e., when q is small, the overall information freshness at the user is larger as the freshness contribution from the slowly varying files can be utilized more with the more polarized distributions of file change rates at the source. We also note that for a fixed λ_i distribution, the freshness of the user increases with the total update rate at the cache C_1 due to the fact that the user gets fresh files more frequently from the cache as the freshness of the files at the cache increases with C_1.

In the third example, we consider a system with two caches, i.e., $m = 2$, with λ_i as given in (30) with $a = 10$, $q = 0.7$, and $n = 10$. We take the total update rate constraint for the user as $U = 20$, for the second cache as $C_2 = 10$ and for the first cache as $C_1 = 4, 8$. We apply the alternating maximization method in Section IV. We observe that the update rates for the caches and for the user have similar trends as shown in Fig. 7(a) for $C_1 = 4$ and in Fig. 7(b) for $C_1 = 8$. Even though the total update rate constraints for the second cache and for the user remain the same in Fig. 7(a)-(b), we see that the update rates at the second cache and at the user change depending on the update rates at the first cache. We observe in Fig. 7(c) that increasing the total update rate constraint for the first cache improves the freshness of every file except the first three files as the total update rate constraints for the caches and for the user are not high enough to update the most rapidly changing files. Furthermore, the improvement on the freshness of the rapidly changing files is more significant than the others.

REFERENCES

[1] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4):390–426, December 2003.

[2] A. Kolobov, Y. Peres, E. Lubetzky, and E. Horvitz. Optimal freshness crawl under politeness constraints. In *ACM SIGIR Conference*, pages 495–504, July 2019.

[3] S. K. Kaul, R. D. Yates, and M. Gruteser. Real-time status: How often should one update? In *IEEE Infocom*, March 2012.

[4] M. Costa, M. Codrenau, and A. Ephremides. Age of information with packet management. In *IEEE ISIT*, June 2014.

[5] A. Soysal and S. Ulukus. Age of information in G/G/1/1 systems: Age expressions, bounds, special cases, and optimization. May 2019. Available on arXiv: 1905.13743.

[6] W. Gao, G. Cao, M. Srivatsa, and A. Iyengar. Distributed maintenance of cache freshness in opportunistic mobile networks. In *IEEE ICDCS*, June 2012.

[7] R. D. Yates, P. Ciblat, A. Yener, and M. Wigger. Age-optimal constrained cache updating. In *IEEE ISIT*, June 2017.

[8] C. Kam, S. Kompella, G. D. Nguyen, J. Wieselthier, and A. Ephremides. Information freshness and popularity in mobile caching. In *IEEE ISIT*, June 2017.

[9] J. Zhong, R. D. Yates, and E. Soljanin. Two freshness metrics for local cache refresh. In *IEEE ISIT*, June 2018.

[10] S. Zhang, J. Li, H. Luo, J. Gao, L. Zhao, and X. S. Shen. Towards fresh and low-latency content delivery in vehicular networks: An edge caching aspect. In *IEEE WCSP*, October 2018.

[11] H. Tang, P. Ciblat, J. Wang, M. Wigger, and R. D. Yates. Age of information aware cache updating with file- and age-dependent update durations. September 2019. Available on arXiv: 1909.05930.

[12] L. Yang, Y. Zhong, F. Zheng, and S. Jin. Edge caching with real-time guarantees. December 2019. Available on arXiv:1912.11847.

[13] M. Wang, W. Chen, and A. Ephremides. Reconstruction of counting process in real-time: The freshness of information through queues. In *IEEE ICC*, July 2019.

[14] Y. Sun, Y. Polyanskiy, and E. Uysal-Biyikoglu. Remote estimation of the Wiener process over a channel with random delay. In *IEEE ISIT*, June 2017.

[15] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor. Age-minimal transmission for energy harvesting sensors with finite batteries: Online policies. *IEEE Transactions on Information Theory*, 66(1):534–556, 2020.

[16] M. Bastopcu and S. Ulukus. Age of information with soft updates. In *Allerton Conference*, October 2018.

[17] B. Buyukates, A. Soysal, and S. Ulukus. Age of information scaling in large networks. In *IEEE ICC*, May 2019.

[18] M. Bastopcu and S. Ulukus. Who should Google Scholar update more often? In *IEEE Infocom*, July 2020.

[19] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[20] D. P. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs: Prentice Hall, 1989.

[21] M. Bastopcu and S. Ulukus. Information freshness in cache updating systems. *IEEE Transactions on Wireless Communications*. To appear. Also available on arXiv:2004.09475.