# Timely Cache Updating in Parallel Multi-Relay Networks

Priyanka Kaswan, *Student Member, IEEE*, Melih Bastopcu, *Member, IEEE*, and Sennur Ulukus, *Fellow, IEEE*

***Abstract*— We consider a system consisting of a server, which receives updates for $N$ files according to independent Poisson processes. The goal of the server is to deliver the latest version of the files to a user through a parallel network of $K$ caches. We consider an update received by the user successful, if the user receives the same file version that is currently prevailing at the server. We derive an analytical expression for information freshness at the user. We observe that freshness for a file increases with increase in consolidation of rates across caches. To solve the multi-cache problem, we first solve the auxiliary problem of a single-cache system. We then rework this auxiliary solution to our parallel-cache network by consolidating rates to single routes as much as possible. This yields an approximate (sub-optimal) solution for the original problem. We provide an upper bound on the gap between the sub-optimal solution and the optimal solution. We present counterpart expressions and policies for version age of information by employing a stochastic hybrid system approach. Numerical results for both timeliness metrics show that the proposed sub-optimal policy closely follows the optimal policy.**

***Index Terms*— Age of information, information freshness, cache updating systems, parallel relay network.**

## I. Introduction

IN THE information age, users want instant access to up-to-date data. Caching is a popular method of pre-storing data at nodes in a network closer to the users for faster delivery of latest data. In recent years, various papers have explored freshness-optimal policies in different settings. Most works have relied on the age of information (AoI) metric to measure freshness of data. AoI has been considered in a wide range of contexts, such as queueing networks [1], [2], [3], [4], energy harvesting systems [5], [6], [7], [8], [9], [10], [11], [12], web crawling [13], [14], scheduling problems [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], UAV systems [28], [29], remote estimation [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], gossip networks [52], [53], [54]. A more detailed literature review on age of information can be found in references [55], [56], [57].

In this work, we consider information timeliness in parallel relay networks. The parallel multi-cache system model can have important applications in modern vehicular networks. For example, in 5G-enabled vehicular networks where self-sustaining wirelessly connected caching stations are placed to enhance vehicular network capacity has been considered in [58]. Especially with a large number of vehicles in an area and considering in practice that these cache stations may have limited power capacities, in order to provide continuous coverage in a vehicular network, multiple caching stations can be placed such that their coverages may intersect with each other. In these intersection areas, a vehicle may have access to multiple base stations to communicate. In such a wirelessly connected parallel caching systems, in our work, we address how to enable timely communication with the caches that have limited communication capacities.

The works that are most closely related to our work here are [11], [12], [26], [27], [44], [45], [46], [47], [48], [49], [50], [51], [53], and [54]. In [26], a single-server single-cache refresh system is considered, where it is shown that an asymptotically optimal policy updates a cached file in proportion to the square root of its popularity. The work in [26] assumes constant file update durations, which is extended in [27] by considering file update durations to be dependent on the size and the age of the files. While [26], [27] use the AoI metric, reference [44] uses a binary freshness metric in a caching system, and determines the optimum update rates at the user and the cache. [44] also extends the approach to a cascade sequence of cache nodes, and [45] generalizes it to the case of nodes with limited cache capacity. In this paper, we further generalize [44] to a more complex network which is composed of parallel caches, shown in Fig. 1(b), where multiple cache nodes are available for relaying file packets from source to the user. Contrary to the above-mentioned works, which optimize timeliness specifically in line networks, where each node of the network has a unique predecessor node responsible for supplying it with fresh packets, in parallel networks this convenience is not at our disposal anymore, making the analysis more challenging. The stochastic hybrid system (SHS) approach has been used to characterize timeliness of the nodes in gossip networks in [50] and [53] with the version age metric (introduced in [50] and [51]), and in [54] with the binary freshness metric. These earlier works [50], [53], [54] are substantially different from our work and mostly focus on the scaling of information freshness in gossip networks. In this

Manuscript received 22 December 2021; revised 16 May 2022 and 31 October 2022; accepted 25 December 2022. Date of publication 17 January 2023; date of current version 9 January 2024. This work was supported by NSF under Grant CCF 17-13977 and Grant ECCS 18-07348. An earlier version of this paper was presented in part at the IEEE International Symposium on Information Theory, Melbourne, Australia, July 2021. The associate editor coordinating the review of this article and approving it for publication was K. Choi. *(Corresponding author: Sennur Ulukus.)*

Priyanka Kaswan and Sennur Ulukus are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: pkaswan@umd.edu; ulukus@umd.edu).

Melih Bastopcu is with the Coordinated Science Laboratory, University of Illinois Urbana–Champaign, Urbana, IL 61801 USA (e-mail: bastopcu@illinois.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2023.3235971.

Digital Object Identifier 10.1109/TWC.2023.3235971

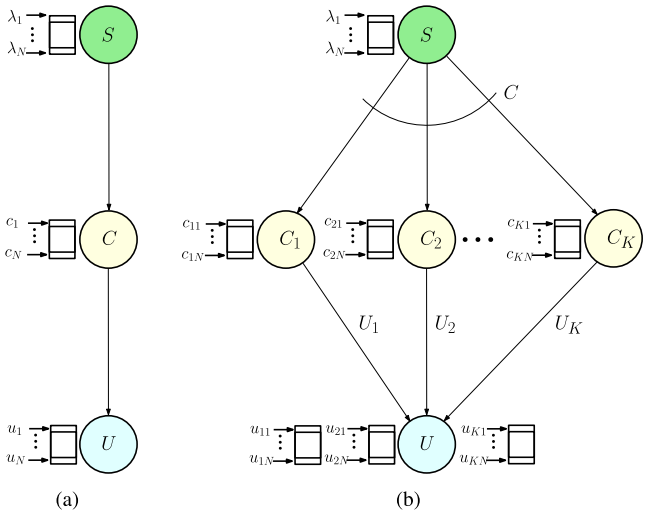


Fig. 1. System models for (a) a single-cache system, and (b) a parallel multi-cache system.

paper, in addition to presenting closed form expressions and policies with binary freshness using first principles, we use the SHS method to set forth parallel formulations for version age in parallel relay networks. Compared to the earlier works, this is the first work that considers characterization and then optimization of the binary freshness and version age metrics in parallel caching networks.

Other related work that use caching and relaying techniques for freshness include: [46] where a tradeoff between content freshness and service latency from the aspect of mobile edge caching is studied; [47] which considers caching policies in opportunistic networks; [11] where a cache-enabled aggregator decides whether to receive a fresh update from an energy harvesting sensor or serve the request with a cached update; [48] where an optimal policy is derived when the current rate of requests for a file is dependent on both the history of requests and the freshness of the file; [49] which considers a two-hop status update system where an optimal scheduling policy is identified by a constrained Markov decision process approach; and [12] where a two-hop system with energy harvesting at source and relay nodes is considered.

The main contributions of our work can be summarized as follows. In this paper, we consider a parallel network with multiple cache routes[1] between a source and a user; see Fig. 1(b). In Section II, we first derive a closed-form expression for freshness at the user. Then, our goal is to maximize the overall freshness at the user subject to the constraints on the total update rates at the caches and at the user. For that, in Section III, we observe from the freshness formula of the two-cache system that lop-sided distribution of rates across the routes supports higher freshness. Further, for the two-route two-file case, restricting at least one of the files to a single route maximizes the overall freshness of the system. Moreover, in a $K$-cache system, restricting a file to fewer routes improves the freshness. Motivated by these properties, in Section IV, we solve an auxiliary problem of a single-cache system and adapt its solution to our parallel cache network to obtain an approximate (sub-optimal) solution for the original problem. We provide an upper bound on the gap between the sub-optimal policy and the optimal policy. The gap is finite and is independent of the number of files. In Section V, we extend our results from the binary freshness metric to the version age of information metric by using the SHS method. Then, we obtain analogous results in parallel networks with the version age of information metric. In Section VI, we optimize the version age metric both in serial and parallel relay networks. Finally, in Section VII, numerical results for both timeliness metrics show that the proposed sub-optimal policies closely approximate the optimal policy.

[1]The file transfer path from the source node to the user node via cache node $k$ is referred to as route $k$ in this paper. Further, the words cache, cache node and relay are used interchangeably.

## II. System Model and Freshness Function

This work is based on the system model depicted in Fig. 1(b). The system consists of a source, $K$ parallel relays and a user. The source has most up-to-date versions of a library of $N$ files. Update packets for file $i$ arrive at the source according to a rate $\lambda_i$ Poisson process. The goal is to update the user through the parallel network of $K$ relays that have cache memories. The source in turn offers updates for file $i$ to cache $k$ according to a rate $c_{ki}$ Poisson process. We assume that there is no delay or information loss in any source-cache links or cache-user links.[2] The source is subject to a total update rate constraint $\sum_{k=1}^{K}\sum_{i=1}^{N} c_{ki} \leq C$, emanating from energy or cost limitations in real-world wireless networks. The cache $k$ sends updates for file $i$ to the user according to a rate $u_{ki}$ Poisson process and is subject to total update rate constraint $\sum_{i=1}^{N} u_{ki} \leq U_k$, for $k = 1, \ldots, K$.

When a file is updated at the source, the stored versions of the same file at the caches and at the user become outdated. Thus, we consider an update received by the user successful, if the user receives a file version that is currently prevailing at the server. This will happen when the source updates the cache and the cache in turn updates the user before the source gets updated with a newer version. In the following subsections, we first derive freshness expression for file $i$ in a single-cache model, and later in a multi-cache model by using a probabilistic approach.[3] For simplicity, we drop subscript $i$ from $\lambda_i$, $c_{ki}$ and $u_{ki}$ in the following subsection since the derivation is valid for all files (for all $i$).

### A. Freshness for File $i$ in the Single-Cache Model

In this subsection, we calculate the freshness expression for file $i$ for the single-cache system shown in Fig. 1(a). First, we characterize the freshness at the cache. In Fig. 2(a), the freshness evolution at the cache is shown between two file updates at the source. We define the freshness function for file $i$ at the cache as follows

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

Let $T_s(i,j)$ denote the $j$th update cycle at the source, i.e., time interval between the $j$th and $(j+1)$th update for

[2]Transmission times and information loss in links considered in [59] can be assumed to be negligible if the distance between the source and the caches are small and/or the file sizes are relatively small compared to the transmission capacity as argued in [44] and [60].

[3]By using the stochastic hybrid system (SHS) approach, we provide an alternative way to characterize binary freshness in Section V-A.

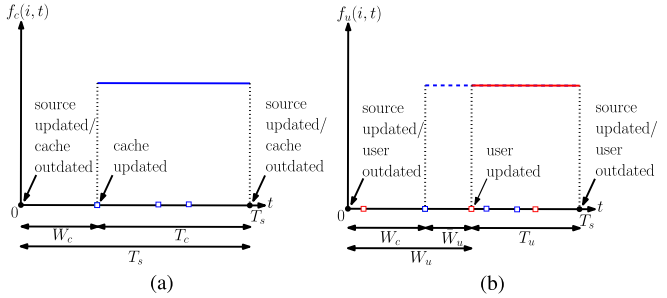


Fig. 2. Freshness function as a function of time (a) at the cache, and (b) at the user.

file $i$. Once the source gets updated, the cache is updated after duration $W_c(i,j)$ and it remains updated for $T_c(i,j) = T_s(i,j) - W_c(i,j)$ duration. For simplicity, we drop index $i$ for variables $T_s(i,j)$, $T_c(i,j)$, and $W_c(i,j)$, as the results in this subsection pertain to file $i$. The long term average freshness of file $i$ at the cache denoted by $F_c(i)$ is equal to

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

Let $M$ be the number of update cycles in time duration $T$. Provided that the system is ergodic, similar to [44], $F_c(i)$ can be equivalently written as

$$F_c(i) = \lim_{T\to\infty} \frac{M}{T}\left(\frac{1}{M}\sum_{j=1}^{M} T_c(j)\right) = \frac{\mathbb{E}[T_c]}{\mathbb{E}[T_s]}. \tag{3}$$

Here, as $T_c(j)$ are independent and identically distributed (i.i.d.) over $j$, we drop the index $j$ and denote $T_c(j)$ with the typical random variable $T_c$. Similarly, $T_s$ and $W_c$ denote the typical random variables for $T_s(j)$ and $W_c(j)$, respectively.

Since the files at the source are updated according to Poisson process, $T_s$ is an exponential random variable with a typical update rate $\lambda$, we have $\mathbb{E}[T_s] = \frac{1}{\lambda}$. We define $W_c'$ as the waiting time of receiving an update at the cache, which is exponentially distributed with rate $c$. Then, we have $W_c = \min\{W_c', T_s\}$, as $W_c$ either takes a value in between 0 and $T_s$, or takes value $T_s$, i.e., $W_c = T_s$, and in this case, the cache is not updated in that cycle. Since $W_c$ is the minimum of two exponential random variables, $W_c$ is an exponential random variable with rate $\lambda + c$ and thus, we obtain $\mathbb{E}[W_c] = \frac{1}{\lambda+c}$. Then, we obtain $\mathbb{E}[T_c]$ as

$$\mathbb{E}[T_c] = \mathbb{E}[T_s] - \mathbb{E}[W_c] = \frac{1}{\lambda} - \frac{1}{\lambda+c} = \frac{c}{\lambda(\lambda+c)}. \tag{4}$$

Finally, by substituting (4) into (3), we obtain $F_c(i)$ as

$$F_c(i) = \frac{\mathbb{E}[T_c]}{\mathbb{E}[T_s]} = \frac{c}{\lambda+c}. \tag{5}$$

Next, we characterize the freshness at the user. Freshness evolution at the user in an update cycle is shown in Fig. 2(b). We define the freshness function for file $i$ at the user as follows

$$f_u(i,t) = \begin{cases} 1, & \text{if file } i \text{ at the user is fresh at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Once file $i$ is updated at the cache after $W_c(j)$, and the same file is updated at the user after $\bar{W}_u(j)$, file $i$ at the user remains fresh for a time period of $T_u(j)$. Thus, the total waiting time for the user to get the freshest version of file $i$ in the $j$th cycle is $W_u = W_c + \bar{W}_u$. We denote $F_u(i)$ as the long term average freshness of file $i$ at the user which is given by

$$\begin{aligned} F_u(i) &= \lim_{T\to\infty} \frac{1}{T}\int_0^T f_u(i,t)dt \\ &= \lim_{T\to\infty} \frac{M}{T}\left(\frac{1}{M}\sum_{j=1}^{M} T_u(j)\right) = \frac{\mathbb{E}[T_u]}{\mathbb{E}[T_s]}, \end{aligned} \tag{7}$$

where $T_u$ denotes the typical random variable for $T_u(j)$ and $M$ is the number of update cycles in time duration $T$.

First, we find $\mathbb{E}[W_u] = \mathbb{E}[\bar{W}_u] + \mathbb{E}[W_c]$ by using nested expectations. Similarly, we define $\bar{W}_u'$ as the waiting time of receiving an update at the user. When the $j$th update arrives at the source, due to memoryless property of the exponential distribution, $W_c'$ and $\bar{W}_u'$ are exponentially distributed with rates $c$ and $u$, respectively. Hence, the distribution of $W_u' = \bar{W}_u' + W_c'$ denoted by $f_{W_u'}(x)$ is equal to the convolution of the distributions of $W_c'$ and $\bar{W}_u'$,

$$\begin{aligned} f_{W_u'}(x) &= f_{\bar{W}_u'}(x) * f_{W_c'}(x) \\ &= \frac{cu}{c-u}\left(e^{-ux} - e^{-cx}\right), \quad 0 \le x < \infty. \end{aligned} \tag{8}$$

For a given update cycle duration $T_s$ at the source, the total waiting time $W_u = \min\{W_u', T_s\}$ either takes a value in between 0 and $T_s$, or $W_u = T_s$. When $W_u = T_s$, we note that the file at the user is not updated in that cycle. Thus, we have

$$\begin{aligned} \mathbb{E}[W_u|T_s = t] &= \int_0^t x f_{W_u}(x)dx + \int_t^\infty t f_{W_u}(x)dx \\ &= \frac{cu}{c-u}\left[\frac{1-e^{-ut}}{u^2} - \frac{1-e^{-ct}}{c^2}\right]. \end{aligned} \tag{9}$$

By using $\mathbb{E}[W_u] = \mathbb{E}[\mathbb{E}[W_u|T_s]]$, we obtain $\mathbb{E}[W_u]$ as

$$\begin{aligned} \mathbb{E}[W_u] &= \int_0^\infty \frac{cu}{c-u}\left[\frac{1-e^{-ut}}{u^2} - \frac{1-e^{-ct}}{c^2}\right]\lambda e^{-\lambda t}dt \\ &= \frac{\lambda + c + u}{(\lambda+u)(\lambda+c)}. \end{aligned} \tag{10}$$

Using (10), we obtain $\mathbb{E}[T_u]$ as

$$\begin{aligned} \mathbb{E}[T_u] &= \mathbb{E}[T_s] - \mathbb{E}[W_u] = \frac{1}{\lambda} - \frac{\lambda+c+u}{(\lambda+u)(\lambda+c)} \\ &= \frac{uc}{\lambda(\lambda+u)(\lambda+c)}. \end{aligned} \tag{11}$$

Finally, by substituting (11) into (7), we obtain $F_u(i)$ as

$$F_u(i) = \frac{\mathbb{E}[T_u]}{\mathbb{E}[T_s]} = \frac{u}{\lambda+u}\frac{c}{\lambda+c}, \tag{12}$$

which is equal to the freshness expression in [44]. Above, we have provided an alternative method (to [44]) to derive freshness, which will be useful in the multi-cache system next.

### *B. Freshness of File $i$ in the Multi-Cache Model*

In this subsection, we find the freshness expression of file $i$ for a multi-cache system shown in Fig. 1(b). Again, for simplicity, we drop file index $i$ from all variables in this subsection.

Each cache sends its updates to the user independent of other caches. After the file at the source is updated for the $j$th time, the file at the user becomes fresh again by the first successful update by any one of the caches. The file at cache $k$ is updated after $W'_{c_k}$ duration. Cache $k$ updates the same file at the user after $\bar{W}'_{u_k}$ duration. We denote the random variable $X_k = W'_{c_k} + \bar{W}'_{u_k}$ as the total waiting time for cache $k$ to send a successful update to the user. As $W'_{c_k}$ and $\bar{W}'_{u_k}$ are exponentially distributed with rates $c_k$ and $u_k$, respectively, similar to (8), we have $f_{X_k}(x) = \frac{c_k u_k}{c_k - u_k}\left(e^{-u_k x} - e^{-c_k x}\right)$ for $x \geq 0$. For a given update cycle $T_s = t$, the user is updated after $W_u$ given by

$$W_u = \min\{t, X_1, X_2, \ldots, X_K\}, \tag{13}$$

where $W_u = t$ denotes the case where the user is not updated in that update cycle. The complementary cumulative distribution function (ccdf) of $X_k$ is given by

$$\mathbb{P}(X_k > x) = \begin{cases} \frac{c_k u_k}{c_k - u_k}\left(\frac{e^{-u_k x}}{u_k} - \frac{e^{-c_k x}}{c_k}\right), & x \geq 0, \\ 1, & x < 0. \end{cases} \tag{14}$$

Since $W_u$ takes only positive values, $\mathbb{E}[W_u]$ can be found by integrating its ccdf as follows

$$\begin{aligned}\mathbb{E}[W_u|T_s = t] &= \int_0^\infty \mathbb{P}(W_u > x)dx \\ &= \int_0^t \mathbb{P}(X_1 > x)\mathbb{P}(X_2 > x)\cdots\mathbb{P}(X_K > x)dx.\end{aligned} \tag{15}$$

For ease of exposition, let $p_v = (p_i)_{i\in[k]} \in \Pi_k\{c_k, u_k\} = V_p$, and $S_c = \sum_{k=1}^K \mathbb{1}\{p_k = c_k\}$. Then, we have

$$\begin{aligned}\mathbb{E}[W_u|T_s = t] = &\frac{\prod_k c_k \prod_k u_k}{\prod_k (c_k - u_k)} \\ &\times \left(\sum_{p_v \in V_p} \frac{(-1)^{S_c}\left(1 - e^{-t\left(\sum_k p_k\right)}\right)}{\left(\sum_k p_k\right)\prod_k p_k}\right).\end{aligned} \tag{16}$$

Next, we find $\mathbb{E}[W_u] = \mathbb{E}[\mathbb{E}[W_u|T_s]]$ as follows

$$\mathbb{E}[W_u] = \frac{\prod_k c_k \prod_k u_k}{\prod_k (c_k - u_k)}\left(\sum_{p_v \in V_p} \frac{(-1)^{S_c}}{\prod_k p_k}\frac{1}{\lambda + \sum_k p_k}\right). \tag{17}$$

As given in (7), the long term average freshness of file $i$ at the user is

$$\begin{aligned}F_u(i) = \frac{\mathbb{E}[T_u]}{\mathbb{E}[T_s]} &= 1 - \lambda\mathbb{E}[W_u] \\ &= 1 - \frac{\prod_k c_k \prod_k u_k}{\prod_k (c_k - u_k)}\left(\sum_{p_v \in V_p} \frac{(-1)^{S_c}}{\prod_k p_k}\frac{1}{1 + \frac{\sum_k p_k}{\lambda}}\right).\end{aligned} \tag{18}$$

Our goal is to maximize the overall freshness at the user, i.e., $\sum_{i=1}^N F_u(i)$, under the given total update rate constraints at the caches and at the user. We note that when $K = 1$, i.e., a single-cache system, the user freshness in (18) reduces to the expression in (12), given by

$$\begin{aligned}F_u(i) &= 1 - \frac{cu}{(c-u)}\left(\frac{1}{u}\frac{1}{1+\frac{u}{\lambda}} - \frac{1}{c}\frac{1}{1+\frac{c}{\lambda}}\right) \\ &= \left(\frac{u}{\lambda+u}\right)\left(\frac{c}{\lambda+c}\right).\end{aligned} \tag{19}$$

When $K = 2$, i.e., a two-cache system, the user freshness in (18) reduces to

$$\begin{aligned}F_u(i) = 1 &- \frac{(c_1c_2)(u_1u_2)}{(c_1-u_1)(c_2-u_2)} \\ &\times \left(\frac{1}{c_1c_2}\frac{1}{1+\frac{c_1+c_2}{\lambda}} - \frac{1}{u_1c_2}\frac{1}{1+\frac{u_1+c_2}{\lambda}}\right. \\ &\left. + \frac{1}{u_1u_2}\frac{1}{1+\frac{u_1+u_2}{\lambda}} - \frac{1}{c_1u_2}\frac{1}{1+\frac{c_1+u_2}{\lambda}}\right)\end{aligned}$$

which can be equivalently written as

$$\begin{aligned}F_u(i) = &\frac{(u_1+u_2)(c_1+c_2)}{(\lambda+u_1+u_2)(\lambda+c_1+c_2)} \\ &- \frac{\lambda}{(\lambda+u_1+u_2)(\lambda+c_1+c_2)} \\ &\times \left(\frac{u_2c_1}{\lambda+u_1+c_2} + \frac{u_1c_2}{\lambda+u_2+c_1}\right).\end{aligned} \tag{20}$$

Interestingly, comparing (12) and (20), we note that freshness in a two-cache system with update rates $(c_1, c_2)$ from the source to the caches and $(u_1, u_2)$ from caches to the user, yields a smaller freshness than in a single-cache system with an update rate $c = c_1 + c_2$ from the source to a cache and $u = u_1 + u_2$ from the cache to the user due to the negative term in (20).

We observe that as $\lambda$ becomes very large, $F_u(i)$ in (20) approaches zero, i.e., $\lim_{\lambda\to\infty} F_u(i) = 0$. This supports the intuition that it is difficult to maintain fresh information at the user when the source keeps getting newer update packets with high frequency. On the other hand, as $\lambda$ approaches zero, which corresponds to the scenario where files never get updated at the source, $F_u(i)$ becomes 1, since the user needs to receive each file only once and from that point onwards, the user forever remains fresh. Further, interestingly, setting either one of the variables $u_k$ and $c_k$ for $k = 1, 2$ to zero transforms (20) into (19), rendering the route $k$ bereft of any use and reducing the model to a single cache system.

## III. Structure of the Optimal Policy

In this section, we find the optimum update rate allocation structure for the most general system with $K$ caches and $N$ files. First, we consider the system with $K = 2$ caches and $N = 2$ files. We denote route $k$ as the file update path from source through cache $k$ to the user. Again dropping file index $i$ for the remainder of the section, let user update rates for file $i$ be $u_1$ and $u_2$ in route 1 and route 2, respectively, also let cache update rates in route 1 and route 2 be $c_1$ and $c_2$, respectively, as shown in Fig. 3. We define the average update rates arriving to user as $\bar{u} = \frac{u_1+u_2}{2}$ and arriving to caches as $\bar{c} = \frac{c_1+c_2}{2}$, and deviation from these averages as $b = \frac{u_2-u_1}{2}$ and $a = \frac{c_2-c_1}{2}$. Thus, $u_1 = \bar{u} - b$, $u_2 = \bar{u} + b$, $c_1 = \bar{c} - a$, and $c_2 = \bar{c} + a$.

In the next lemma, for given user rates $u_1$ and $u_2$ (therefore, given $\bar{u}$ and $b$), and the total cache rate $2\bar{c}$, we find the optimal distribution of cache rates to maximize the freshness at the user, that is, we find the optimal $a$, $a^*$, in terms of $b$, $\bar{u}$ and $\bar{c}$.

*Lemma 1:* In a cache update system with $K = 2$ parallel caches, for given user rates $u_1$ and $u_2$, and the total cache rate

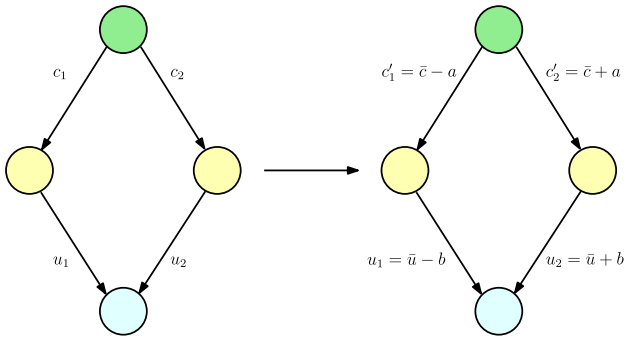


Fig. 3. Finding optimal cache rates in each route for single file by reshuffling while keeping total cache rate constant.

$2\bar{c}$, the optimal cache rates are $c_1^* = \bar{c} - a^*$ and $c_2^* = \bar{c} + a^*$, where

$$a^* = \min\Bigg\{b + \frac{(\bar{c}+\lambda+\bar{u})}{b(2\bar{c}+\lambda)}\Bigg(\bar{u}(2\bar{c}+\lambda+\bar{u}) - b^2 - \sqrt{(\bar{u}^2-b^2)((2\bar{c}+\lambda+\bar{u})^2-b^2)}\Bigg), \bar{c}\Bigg\}. \quad (21)$$

*Proof:* We write (20) equivalently as follows after inserting $\bar{u}$, $\bar{c}$, $a$ and $b$,

$$F_u(i) = \frac{4\bar{c}\bar{u}}{(\lambda+2\bar{c})(\lambda+2\bar{u})} - \frac{\lambda}{(\lambda+2\bar{c})(\lambda+2\bar{u})} \times \left(\frac{(\bar{u}-b)(\bar{c}+a)}{\bar{c}+\lambda+\bar{u}+b-a} + \frac{(\bar{u}+b)(\bar{c}-a)}{\bar{c}+\lambda+\bar{u}+a-b}\right). \quad (22)$$

Since $\bar{u}$ and $\bar{c}$ are fixed, the first term and pre-factor of the second term in (22) are fixed. Taking the derivative of $F_u(i)$ with respect to $a$ yields

$$\frac{dF_u(i)}{da} = \frac{\lambda}{(\lambda+2\bar{c})(\lambda+2\bar{u})}\left(\frac{(b-\bar{u})}{(b-a+\bar{c}+\lambda+\bar{u})} + \frac{(b+\bar{u})}{(a-b+\bar{c}+\lambda+\bar{u})} + \frac{(a+\bar{c})(b-\bar{u})}{(b-a+\bar{c}+\lambda+\bar{u})^2} - \frac{(b+\bar{u})(a-\bar{c})}{(a-b+\bar{c}+\lambda+\bar{u})^2}\right). \quad (23)$$

Setting $\frac{dF_u(i)}{da} = 0$ yields two solutions. Since $a$ values are restricted in range $0 < a < \bar{c}$, there is only one valid solution which is the first part of the min in (21).

The second derivative of freshness with respect to $a$ is given by

$$\frac{d^2F_u(i)}{da^2} = \frac{\lambda}{(\lambda+2\bar{c})(\lambda+2\bar{u})}\left(\frac{2(b-\bar{u})}{(b-a+\bar{c}+\lambda+\bar{u})^2} - \frac{2(b+\bar{u})}{(a-b+\bar{c}+\lambda+\bar{u})^2} + \frac{2(a+\bar{c})(b-\bar{u})}{(b-a+\bar{c}+\lambda+\bar{u})^3} + \frac{2(b+\bar{u})(a-\bar{c})}{(a-b+\bar{c}+\lambda+\bar{u})^3}\right). \quad (24)$$

Again, as $0 < b < \bar{u}$, the valid solution yields $\frac{\partial^2 F_u(i)}{\partial a^2} < 0$, from which we conclude that $a^*$ in (21) maximizes the freshness at the user.

Furthermore, given $0 \le x, y \le 1$ , the inequality $1 - xy - \sqrt{(1-x^2)(1-y^2)} \ge 0$ holds true, hence, by substituting $x = b/\bar{u}$ and $y = b/(2\bar{c}+\lambda+\bar{u})$ in (21), we get

$$a^* = b + \frac{(\bar{c}+\lambda+\bar{u})}{b(2\bar{c}+\lambda)}\bar{u}(2\bar{c}+\lambda+\bar{u}) \times \left(1 - xy - \sqrt{(1-x^2)(1-y^2)}\right) \ge b. \quad (25)$$

Differentiating both sides with respect to $b$ gives $\frac{\partial a^*}{\partial b} \ge 0$, and thus, $a^*$ increases monotonically with $b$, till it reaches $\bar{c}$, after which $a^*$ is equal to $\bar{c}$, yielding the result provided in (21). ■

Next, we define $\tilde{F}_u(i)$ as the *cache-update-rate-optimized* freshness, where for fixed $u_1$, $u_2$, we insert the optimal cache update rates $c_1^*$ and $c_2^*$ in (20). Note that $\tilde{F}_u(i)$ is a function of $b$, $\bar{u}$ and $\bar{c}$. In the following lemma, we show that as $u_1$, $u_2$ get more *lopsided*, i.e., as the difference $(u_2 - u_1)$ increases, cache-update-rate-optimized freshness $\tilde{F}_u(i)$ increases.

*Lemma 2:* $\tilde{F}_u(i)$ is an increasing function of $b$.

*Proof:* We prove the statement by showing $\frac{d\tilde{F}_u(i)}{db} > 0$. Here, we consider two different cases for the value of $a$.

In the first case, we consider $a = a^* < \bar{c}$. Assuming $a = a^*$ in (22) to get $\tilde{F}_u(i)$ and using the chain rule, we have

$$\frac{d\tilde{F}_u(i)}{db} = \frac{\partial \tilde{F}_u(i)}{\partial b} + \frac{\partial \tilde{F}_u(i)}{\partial a}\frac{da}{db} = \frac{\partial \tilde{F}_u(i)}{\partial b}, \quad (26)$$

where (26) follows from the fact that $\frac{\partial \tilde{F}_u(i)}{\partial a} = 0$ when $a = a^*$. We define $F_1$ as follows

$$F_1 = \frac{(\bar{u}-b)(\bar{c}+a)}{(\bar{c}+\lambda+\bar{u}+b-a)} + \frac{(\bar{u}+b)(\bar{c}-a)}{(\bar{c}+\lambda+\bar{u}+a-b)}, \quad (27)$$

which can be equivalently written as

$$F_1 = \frac{(2\bar{u}+\bar{c}+\lambda-a)(\bar{c}+a)}{(\lambda+\bar{u}+\bar{c}+b-a)} + \frac{(2\bar{u}+\bar{c}+\lambda+a)(\bar{c}-a)}{(\lambda+\bar{u}+\bar{c}+a-b)} - 2\bar{c}. \quad (28)$$

Comparing (28) with (22), we note that proving $\frac{\partial \tilde{F}_u(i)}{\partial b} \ge 0$ is the same as proving $\frac{\partial F_1}{\partial b} \le 0$,

$$\frac{\partial F_1}{\partial b} = \big(-(\lambda+\bar{u}+\bar{c}+a-b)^2(2\bar{u}+\bar{c}+\lambda-a)(\bar{c}+a) + (\lambda+\bar{u}+\bar{c}+b-a)^2(2\bar{u}+\bar{c}+\lambda+a)(\bar{c}-a)\big) \times \frac{1}{((\lambda+\bar{u}+\bar{c})^2-(a-b)^2)^2}. \quad (29)$$

Denoting $A = (\lambda+\bar{u}+\bar{c})^2 + (a-b)^2$, $B = 2(\lambda+\bar{u}+\bar{c})(a-b)$, $C = 2\bar{u}+\bar{c}+\lambda$, $D = a$ and $E = \bar{c}$, the expression for $\frac{\partial F_1}{\partial b}$ in (29) becomes

$$\frac{\partial F_1}{\partial b} = -\frac{2(B(CE-D^2)+AD(C-E))}{((\lambda+\bar{u}+\bar{c})^2-(a-b)^2)^2} \le 0, \quad (30)$$

where (30) follows from the fact that $B \ge 0$ due to (25), $D^2 \le CE$ due to $D \le C$ and $D \le E$ as $a < \bar{c}$, $A \ge 0$ and $E \le C$, which completes the proof of the first case.

In the second case, we consider $a = a^* = \bar{c}$. For this case, $F_1$ in (28) becomes

$$F_1 = \left(\frac{(2\bar{u}+\lambda)}{(\lambda+\bar{u}+b)} - 1\right)2\bar{c}. \quad (31)$$

Then, the derivative of $F_1$ in (31) with respect to $b$ becomes

$$\frac{\partial F_1}{\partial b} = \frac{-(2\bar{u}+\lambda)2\bar{c}}{(\lambda+\bar{u}+b)^2} \le 0. \quad (32)$$

As a result, for both cases, we show that $\frac{\partial F_1}{\partial b} \leq 0$ which equivalently implies that $\frac{\partial \tilde{F}_u(i)}{\partial b} \geq 0$, and thus, $\tilde{F}_u(i)$ increases with $b$. ■

Lemma 2 implies that lopsided update rates at the user increase the freshness. Next, for a $K = 2$ cache system with $N = 2$ files, we show that we should restrict at least one of the files to a single route, that is, lopside at least one of the files to an extreme.

*Lemma 3:* In a cache update system with $K = 2$ caches and $N = 2$ files, in the optimal policy, we need to restrict at least one file to a single route.

*Proof:* Let the average rates at the caches and at the user hold values $\bar{c}_i = \frac{c_{1i}+c_{2i}}{2}$ and $\bar{u}_i = \frac{u_{1i}+u_{2i}}{2}$ for $i = 1, 2$ which fixes total user rates and total cache rates. Similarly, we have $u_{1i} = \bar{u}_i - b_i$ and $u_{2i} = \bar{u}_i + b_i$ which satisfies the total update rate constraints $u_{11} + u_{12} = U_1$ and $u_{21} + u_{22} = U_2$. Then, we change the update rates at the user to $u'_{11} = \bar{u}_1 - b_1 - \delta_1$, $u'_{21} = \bar{u}_1 + b_1 + \delta_1$, $u'_{12} = \bar{u}_2 - b_2 + \delta_2$ and $u'_{22} = \bar{u}_2 + b_2 - \delta_2$ such that we have $|\delta_1| = |\delta_2|$, $u'_{11} + u'_{12} = U_1$, and $u'_{21} + u'_{22} = U_2$ still hold. We analyze two cases of shuffling, shown in Fig. 4.

In the first case, increasing $b_i$ for one file leads to increasing $b_i$ value for the other file as shown in Fig. 4(a). As distribution of user rates for both files become lopsided simultaneously, it is a win-win situation for both files. For this case, we increase $b_i$ values of files till one file is completely in a single route. For example, in Fig. 4(a), the user rates for the second file (shown in yellow) are $\bar{u}_2 - b_2$ and $\bar{u}_2 + b_2$ in route 1 and route 2, respectively. Then, we increase $b_2$ till $\bar{u}_2 - b_2 = 0$ in route 1 and the second file is completely restricted to route 2. Such shuffling also leads to a simultaneous increase in $b_1$.

In the second case, increasing $b_i$ value of one file decreases $b_i$ value of the other file. This case is shown in Fig. 4(b) where both files have larger user update rates in route 2. In order to determine which file to prioritize, we compare $\frac{d\tilde{F}_u(i)}{db_i}$ for both files. If $\frac{d\tilde{F}_u(1)}{db_1} > \frac{d\tilde{F}_u(2)}{db_2}$, then we prioritize improving freshness of file 1. We observe that $\frac{d^2\tilde{F}_u(i)}{db_i^2} > 0$. Thus, the increase in freshness of file 1 is always larger than the decrease in freshness of file 2. Similarly, if $\frac{d\tilde{F}_u(2)}{db_2} > \frac{d\tilde{F}_u(1)}{db_1}$, then we increase the freshness of the second file which decreases the freshness of the first file. Thus, we need to restrict at least one file to a single route to obtain the optimum freshness. ■

Thus, for a $K = 2$ cache and $N = 2$ file system with a given set of update rates $u_{11}$, $u_{21}$, $u_{12}$ and $u_{22}$, we can shuffle these rates to increase the total freshness while keeping average rates $\bar{u}_1$, $\bar{u}_2$, $\bar{c}_1$, and $\bar{c}_2$ the same. In this process, we always end up restricting one of the files to only one route. Extending this result to a $K = 2$ cache but arbitrary $N$ files case, we iteratively choose a pair of files and increase freshness of the pair by restricting one of these files to a single route. We repeat this process until we restrict $N - 1$ files to a single route each. Thus, for a $K = 2$ cache, arbitrary $N$ file system, only at most one file will be updated through both relays, and the remaining $N - 1$ files will settle to a single relay.

*Lemma 4:* Freshness of a file in a $K$-cache system with update rates at the cache $(c_1, c_2, c_3, \ldots, c_K)$, and at the user $(u_1, u_2, u_3, \ldots, u_K)$ is smaller than the freshness in a $(K-1)$-cache system with update rates at the cache $(c_1 + c_2, c_3, \ldots, c_K)$, and at the user $(u_1 + u_2, u_3, \ldots, u_K)$.

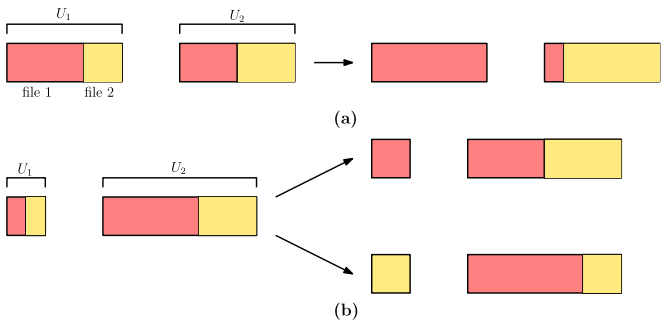


Fig. 4. Shuffling user rates for improving freshness. (a) Freshness of both files improve (file 2 only in route 2). (b) In upper branch, freshness of file 1 decreases and of file 2 increases (file 2 only in route 2). In lower branch, freshness of file 1 increases and of file 2 decreases (file 1 only in route 2).

*Proof:* With the notation of Section II, since freshness $F_u(i) = 1 - \lambda E[W_u]$, where $E[W_u] = \int_0^\infty E[W_u|T_s = t]\lambda e^{-\lambda t}dt$, we prove the lemma by showing $E[W_u^K|T_s = t] - E[W_u^{K-1}|T_s = t] \geq 0$, where $K$ in $W_u^K$ denotes the total waiting time for the user in a $K$-cache system.

For file $i$ in the $K$-cache system, $W_u$ in (13) states that

$$W_u^K = \min\{t, X_1, X_2, X_3, \ldots, X_K\}. \tag{33}$$

Therefore, similar to (15), we have

$$\begin{aligned}\mathbb{E}[W_u^K|T_s = t] &= \int_0^\infty \mathbb{P}(W_u^K > x)dx \\ &= \int_0^t \mathbb{P}(X_1 > x)\mathbb{P}(X_2 > x)\cdots\mathbb{P}(X_K > x)dx,\end{aligned} \tag{34}$$

where $X_k$ is the total waiting time for cache $k$ to send a successful update to the user which has the following distribution

$$\mathbb{P}(X_k > x) = \begin{cases} \dfrac{c_k e^{-u_k x} - u_k e^{-c_k x}}{(c_k - u_k)}, & x \geq 0, \\ 1, & x < 0. \end{cases} \tag{35}$$

For file $i$ in $(K-1)$-cache system, we have $W_u^{K-1} = \min\{t, X_p, X_3, \ldots, X_K\}$ where $X_p$ is the total waiting time in the route having update rate $c_1 + c_2$ from source to cache and $u_1 + u_2$ from cache to user, backed by the following distribution

$$\mathbb{P}(X_p > x) = \begin{cases} \dfrac{(c_1+c_2)e^{-(u_1+u_2)x} - (u_1+u_2)e^{-(c_1+c_2)x}}{((c_1+c_2)-(u_1+u_2))}, & x \geq 0, \\ 1, & x < 0. \end{cases} \tag{36}$$

Then, we write $\mathbb{E}[W_u^{K-1}|T_s = t]$ as

$$\begin{aligned}\mathbb{E}[W_u^{K-1}|T_s = t] &= \int_0^\infty \mathbb{P}(W_u^{K-1} > x)dx \\ &= \int_0^t \mathbb{P}(X_p > x)\mathbb{P}(X_3 > x)\cdots\mathbb{P}(X_K > x)dx.\end{aligned} \tag{37}$$

Hence, we obtain $\mathbb{E}[W_u^K|T_s=t]-\mathbb{E}[W_u^{K-1}|T_s=t]$ as

$$\int_0^t \left(\mathbb{P}(X_1>x)\mathbb{P}(X_2>x)-\mathbb{P}(X_p>x)\right) \times \mathbb{P}(X_3>x)\cdots\mathbb{P}(X_K>x)dx. \quad (38)$$

We prove $\mathbb{E}[W_u^K|T_s=t]-\mathbb{E}[W_u^{K-1}|T_s=t]\geq 0$ by showing $G(x)\geq 0$ for all $x\geq 0$, where $G(x)=\mathbb{P}(X_1>x)\mathbb{P}(X_2>x)-\mathbb{P}(X_p>x)$ is given by

$$G(x)=e^{-(c_1+c_2)x}\left(\frac{(c_1e^{(c_1-u_1)x}-u_1)(c_2e^{(c_2-u_2)x}-u_2)}{(c_1-u_1)(c_2-u_2)} - \frac{(c_1+c_2)e^{((c_1+c_2)-(u_1+u_2))x}-(u_1+u_2)}{(c_1+c_2)-(u_1+u_2)}\right). \quad (39)$$

Proving $G(x)\geq 0$ for all $x\geq 0$ is equivalent to proving $G_1(x)\geq 0$ for all $x\geq 0$, where

$$G_1(x)=\left(\frac{(c_1e^{(c_1-u_1)x}-u_1)(c_2e^{(c_2-u_2)x}-u_2)}{(c_1-u_1)(c_2-u_2)} - \frac{(c_1+c_2)e^{((c_1+c_2)-(u_1+u_2))x}-(u_1+u_2)}{(c_1+c_2)-(u_1+u_2)}\right). \quad (40)$$

Since $G_1(0)=0$, we prove $G_1(x)\geq 0$ for $x\geq 0$ by showing that $\frac{dG_1(x)}{dx}\geq 0$ for $x\geq 0$. Then, we obtain $\frac{dG_1(x)}{dx}=e^{((c_1+c_2)-(u_1+u_2))x}G_2(x)$ where

$$G_2(x)=\left(\frac{c_1\left(c_2-u_2e^{-(c_2-u_2)x}\right)}{(c_2-u_2)} + \frac{c_2\left(c_1-u_1e^{-(c_1-u_1)x}\right)}{c_1-u_1}-(c_1+c_2)\right), \quad (41)$$

for $x\geq 0$. Since $G_2(0)=0$ and $\frac{dG_2(x)}{dx}=c_1u_2e^{-(c_2-u_2)x}+c_2u_1e^{-(c_1-u_1)x}\geq 0$, we show that $G_2(x)\geq 0$ for $x\geq 0$ which proves that $G(x)\geq 0$ for $x\geq 0$, and thus, completes the proof. ■

Hence, for given total update rates $\sum_{k=1}^K u_{ki}$ and $\sum_{k=1}^K c_{ki}$ for file $i$, the maximum freshness is obtained by concentrating the rates in a single route to the extent possible. In the next section, we provide an approximate way of finding total update rates for files and scheduling them to individual links.

## IV. The Optimization of Binary Freshness in Parallel Relay Networks

Based on the insights obtained for the optimal caching structure in Section III, in this section, we provide a sub-optimal solution that finds the cache and user update rates for each file, and then, propose a way to allocate these rates to the cache routes. We first write the freshness maximization problem for our system as,

$$\max_{c_{ki},u_{ki}} \sum_{i=1}^N F_u(i)$$
$$\text{s.t.} \sum_{k=1}^K\sum_{i=1}^N c_{ki}\leq C$$
$$\sum_{i=1}^N u_{ki}\leq U_k,\quad k=1,\ldots,K,$$
$$c_{ki}\geq 0,\quad u_{ki}\geq 0,\ k=1,\ldots,K,\ i=1,\ldots,N. \quad (42)$$

We note that the problem in (42) is not a convex optimization problem since the objective function in (42) is not concave. Moreover, this parallel cache problem is significantly more complex than the cascade cache problem in [44]. A Lagrangian approach as in [44] seems prohibitive as it results in highly nonlinear KKT conditions, and thus, it is difficult to derive closed form expressions for $c_{ki}$ and $u_{ki}$ that satisfy the KKT conditions. For this reason, we pursue an approximate solution approach utilizing the properties of the optimal solution found in Lemma 3.

First, we construct a single-cache problem by bringing all relays together, where the source-to-cache total update constraint is $C$ and the relay-to-user total update constraint is $U=\sum_{k=1}^K U_k$. The optimal solution of this single-cache problem forms an upper bound for the optimum solution of our multi-cache problem, as it allows distributed relays to share update rate capacities. We denote this upper bound by $F_{ub}$.

Second, we extract a feasible solution for our multi-cache problem from the optimum solution of the constructed single-cache problem. We know from Lemma 4 that files need to be restricted to single routes for maximum freshness. Thus, our approximate solution takes the optimum solution of the constructed single-cache problem, and distributes the update rates in the multi-cache setting in such a way that each file is updated only through a single relay to the extent possible. Let the solution of the single-cache problem be $u_i$ which is $u_i=\sum_{k=1}^K u_{ki}$. We assign the files in order of decreasing $u_i$ to one of the routes. We start with the first route and fit fully as many files as possible, till we reach a file which will not fit completely and we make it split rates with the last route (route $K$). We follow this for $K-1$ routes. If a file rate $u_i$ exceeds route capacity $U_k$, we first fill maximal full routes with it, then try to fit the remaining rate fully in the remaining routes. This leaves us with at most $K-1$ files that split rates between two routes. The remaining files go to route $K$. This approximate solution gives us a sub-optimal freshness $F_{so}$. Denoting the optimal freshness in our problem in (42) as $F^*$, we have

$$F_{so}<F^*<F_{ub} \quad (43)$$

which means $F^*-F_{so}\leq F_{ub}-F_{so}$, i.e., the gap between the sub-optimal solution and the optimal solution is bounded by the gap between the upper bound and the sub-optimal solution.

Next, we bound $F_{ub}-F_{so}$. We note that, in the sub-optimal policy, we assign at most $K-1$ files to two routes. From Lemma 2, freshness for file $i$ increases when $b_i$ increases, with minimum at $b_i=0$ ($a^*=0$) and maximum at $b_i=\bar{u}_i$ ($a^*=\bar{c}$). Hence, using (22), we find an upper bound on the maximum freshness loss ratio $\rho$ possible for a file due to splitting as

$$\rho=\frac{F_u(i)|_{(b_i,a_i^*)=(\bar{u}_i,\bar{c}_i)}-F_u(i)|_{(b_i,a_i^*)=(0,0)}}{F_u(i)|_{(b_i,a_i^*)=(\bar{u}_i,\bar{c}_i)}} =\frac{\lambda_i}{2(\lambda_i+\bar{u}_i+\bar{c}_i)}<0.5. \quad (44)$$

Since $F_{ub}(i)<1$, the optimality gap is $F^*-F_{so}\leq\rho(K-1)F_{ub}(i)<0.5(K-1)$. These $K-1$ files have low $u_i$s, owing

to very high or very low $\lambda_i$s, as observed in [44]. In the former case, $F_{ub}(i)$ is low, while in the latter case, $\rho$ is very low.

## V. The SHS Method to Characterize Timeliness in Parallel Cache Systems

The stochastic hybrid system (SHS) approach has been introduced in [3] as an alternative way to characterize the timeliness in communication systems. Specifically, by using the SHS method, timeliness of the nodes in gossip networks is analyzed with the version age metric in [50] and [53], and with the binary freshness metric in [54]. In this section, first, we use the SHS method to provide a different way to find the binary freshness which is derived in Section II by using a probabilistic approach. Then, we utilize the SHS method to characterize the version age in parallel relay networks.

### *A. The Characterization of Binary Freshness by Using the SHS Method*

We define the set $\mathcal{K}$ as $\mathcal{K} \triangleq \{1, 2, \ldots, K\}$ to denote cache indices. Then, we denote the collection of all the subsets of $\mathcal{K}$ with $m$ elements as $\mathcal{K}_m$. For example, we have $\mathcal{K}_1 = \{\{1\}, \{2\}, \ldots, \{K\}\}$. Next, we drop file index $i$ in the remainder of the section and denote the freshness of cache $k$ as $F_k(t)$ and the freshness of any set $S \subseteq \mathcal{K} \cup \{u\}$, where $u$ denotes the index for the user, as $F_S(t) \triangleq \max_{j\in S} F_j(t)$ at time $t$. We define the average freshness of the set $S$ as $F_S = \lim_{t\to\infty} \mathbb{E}[F_S(t)]$. Next, by using the [54, Thm. 1], we write the freshness of the user in $K$-cache system as

$$F_u = \frac{\sum_{k_1=1}^{K} u_{k_1} F_{\{u\}\cup\{k_1\}}}{\lambda + \sum_{k_1=1}^{K} u_{k_1}}. \tag{45}$$

Note that $F_{\{u\}\cup\{k_1\}}$ denotes the freshness of the set consisting of the user and the cache $k_1$, where $k_1 \in \mathcal{K}$. Then, we write $F_{\{u\}\cup\{k_1\}}$, or simply $F_{\{u,k_1\}}$, as

$$F_{\{u,k_1\}} = \frac{c_{k_1} + \sum_{k_2\in\mathcal{K}-\{k_1\}} u_{k_2} F_{\{u,k_1\}\cup\{k_2\}}}{\lambda + c_{k_1} + \sum_{k_2\in\mathcal{K}-\{k_1\}} u_{k_2}}, \quad \{k_1\}\in\mathcal{K}_1, \tag{46}$$

where $\mathcal{K} - \{k_1\}$ denotes the subtraction of $\{k_1\}$ from the set $\mathcal{K}$. Then, we write the freshness expressions for $F_{\{u,k_1,k_2\}}$ in terms of $F_{\{u,k_1,k2\}\cup\{k_3\}}$, and so on. If we continue to write these sequential equations for the $\ell$th time, i.e., $F_{\{u,k_1,k_2,\ldots,k_\ell\}}$, we obtain (47), shown at the bottom of the next page, for $\{k_1, k_2, \ldots, k_\ell\} \in \mathcal{K}_\ell$ where $\ell = 1, 2, \ldots, K-1$. At the last step, we have

$$F_{\{u,1,2,\ldots,K\}} = \frac{\sum_{k=1}^{K} c_k}{\lambda + \sum_{k=1}^{K} c_k}. \tag{48}$$

After obtaining all these freshness expressions, we start inserting the value of $F_{\{u,1,2,\ldots,\ell+1\}}$ back into $F_{\{u,1,2,\ldots,\ell\}}$ for $\ell = 1, 2, \ldots, K-1$, and finally, we obtain the freshness of the user $F_u$ in (45) by inserting the $F_{\{u,k_1\}}$ values. With the SHS method, in order to find the freshness of the user $F_u$ in a parallel $K$-cache system, we need to find $2^K$ different freshness expressions. For example, in a $K = 2$ cache system, we find the freshness at the user $F_u$ by using 4 equations which are given as follows

$$F_u = \frac{u_1 F_{\{u,1\}} + u_2 F_{\{u,2\}}}{\lambda + u_1 + u_2}, \tag{49}$$

where $F_{\{u,1\}}$ and $F_{\{u,2\}}$ are given by

$$F_{\{u,1\}} = \frac{c_1 + u_2 F_{\{u,1\}\cup\{2\}}}{\lambda + c_1 + u_2},$$
$$F_{\{u,2\}} = \frac{c_2 + u_1 F_{\{u,2\}\cup\{1\}}}{\lambda + c_2 + u_1}. \tag{50}$$

Then, we find $F_{\{u,1,2\}}$ as

$$F_{\{u,1,2\}} = \frac{c_1 + c_2}{\lambda + c_1 + c_2}. \tag{51}$$

Finally, by using $F_{\{u,1\}}$ and $F_{\{u,2\}}$ in (50), $F_{\{u,1,2\}}$ in (51), we obtain the freshness expression at the user $F_u$ for $K = 2$ cache system as

$$F_u = \frac{u_1 c_1 + u_2 c_2}{(\lambda+u_1+u_2)(\lambda+c_1+c_2)} + \frac{1}{(\lambda+u_1+u_2)(\lambda+c_1+c_2)} \times \left( \frac{u_2 c_1 (u_1+c_2)}{\lambda+u_1+c_2} + \frac{u_1 c_2 (u_2+c_1)}{\lambda+u_2+c_1} \right) \tag{52}$$

which is the same as (20). Therefore, the SHS method can be used as an alternative way to characterize the binary freshness at the user in a parallel $K$-cache system.

In the following subsection, we use the SHS method to characterize the version age of information in a parallel $K$-cache system.

### *B. The Characterization of Version Age by Using the SHS Method*

The version age of information or simply the version age has been recently introduced in [50] and [51]. Here, each update at the source is denoted as a new version of the information, and thus, the version age measures how many versions a particular node is behind compared to the most current version at the source. We denote the version of the information at the source as $N_s(t)$, at cache $k$ as $N_k(t)$, and at the user as $N_u(t)$ at time $t$. Then, the version age is defined at cache $k$ as $\Delta_k(t) \triangleq N_s(t) - N_k(t)$, and at the user as $\Delta_u(t) \triangleq N_s(t) - N_u(t)$. When the information at the source is updated, the version age at the caches and at the user increases by 1, i.e., $\Delta'_k(t) = \Delta_k(t) + 1$, for all $k \in \mathcal{K}$ and $\Delta'_u(t) = \Delta_u(t) + 1$, where $\Delta'(t)$ represents the version age after the transition. When cache $k$ gets an update from the source, its version age becomes 0, i.e., $\Delta'_k(t) = 0$, as the source has the latest version of the update. However, when the user gets an update from cache $k$, its version age becomes $\Delta'_u(t) = \min\{\Delta_u(t), \Delta_k(t)\}$. In other words, the user updates its information only if cache $k$ has more recent version of the information. We define the version age of any set $S$ as $\Delta_S(t) \triangleq \min_{j\in S} \Delta_j(t)$ at time $t$. In addition, we define the average version age of any set $S$ as $\Delta_S = \lim_{t\to\infty} \mathbb{E}[\Delta_S(t)]$. Correspondingly, $\Delta_k$ and $\Delta_u$ denote the average version age at cache $k$ and at the user. Next, using the [50, Thm. 1], we write the version age of the user in a $K$-cache system as

$$\Delta_u = \frac{\lambda + \sum_{k_1=1}^{K} u_{k_1} \Delta_{\{u\}\cup\{k_1\}}}{\sum_{k_1=1}^{K} u_{k_1}}. \tag{53}$$

Here, $\Delta_{\{u\}\cup\{k_1\}}$, or simply $\Delta_{\{u,k_1\}}$, denotes the version age of the set containing cache $k_1$ and the user. Next,

we write $\Delta_{\{u,k_1\}}$ as

$$\Delta_{\{u,k_1\}}=\frac{\lambda+\sum_{k_2\in\mathcal{K}-\{k_1\}} u_{k_2}\Delta_{\{u,k1\}\cup\{k_2\}}}{c_{k_1}+\sum_{k_2\in\mathcal{K}-\{k_1\}} u_{k_2}}, \quad k_1\in\mathcal{K}_1. \tag{54}$$

Continuing to write sequential equations for the version age for the $\ell$ caches, i.e., $\Delta_{\{u,k_1,k_2,\ldots,k_\ell\}}$, we get

$$\Delta_{\{u,k_1,k_2,\ldots,k_\ell\}} = \frac{\lambda+\sum_{k_{\ell+1}\in\mathcal{K}-\{k_1,k_2,\ldots,k_\ell\}} u_{k_{\ell+1}}\Delta_{\{u,k_1,k_2,\ldots,k_\ell\}\cup\{k_{\ell+1}\}}}{\sum_{\bar{k}_{\ell+1}\in\{k_1,k_2,\ldots,k_\ell\}} c_{\bar{k}_{\ell+1}}+\sum_{k_{\ell+1}\in\mathcal{K}-\{k_1,k_2,\ldots,k_\ell\}} u_{k_{\ell+1}}}, \tag{55}$$

for $\{k_1,k_2,\ldots,k_\ell\}\in\mathcal{K}_\ell$ where $\ell=1,2,\ldots,K-1$. In the final step, we have

$$\Delta_{\{u,1,2,\ldots,K\}}=\frac{\lambda}{\sum_{k=1}^{K} c_k}. \tag{56}$$

After obtaining all these version age expressions, by substituting $\Delta_{\{u,1,2,\ldots,\ell+1\}}$ back into $\Delta_{\{u,1,2,\ldots,\ell\}}$ for $\ell=1,2,\ldots,K-1$, and substituting $\Delta_{\{u,1\}}$ in (54) into $\Delta_u$ in (53), we obtain the version age at the user. Thus, characterizing version age at the user by using the SHS method requires solving $2^K$ different equations sequentially.

Next, as an example, we find the version age in a $K=2$ cache system. For this system, the version age at the user is given by

$$\Delta_u=\frac{\lambda+u_1\Delta_{\{u,1\}}+u_2\Delta_{\{u,2\}}}{u_1+u_2}, \tag{57}$$

where $\Delta_{\{u,1\}}$ and $\Delta_{\{u,2\}}$ are given by

$$\begin{aligned}\Delta_{\{u,1\}}&=\frac{\lambda+u_2\Delta_{\{u,1\}\cup\{2\}}}{c_1+u_2},\\ \Delta_{\{u,2\}}&=\frac{\lambda+u_1\Delta_{\{u,2\}\cup\{1\}}}{c_2+u_1}.\end{aligned} \tag{58}$$

Next, we find $\Delta_{\{u,1,2\}}$ as

$$\Delta_{\{u,1,2\}}=\frac{\lambda}{c_1+c_2}. \tag{59}$$

Finally, we obtain the version age at the user $\Delta_u$ in (57) by using $\Delta_{\{u,1\}}$ and $\Delta_{\{u,2\}}$ in (58), and $\Delta_{\{u,1,2\}}$ in (59) as

$$\Delta_u=\frac{\lambda}{u_1+u_2}+\frac{\lambda}{c_1+c_2}+\frac{\lambda}{(u_1+u_2)(c_1+c_2)}\times\left(\frac{u_2c_1}{u_1+c_2}+\frac{u_1c_2}{u_2+c_1}\right). \tag{60}$$

In the next section, we optimize the update rates and select file routes that minimize the average version age of the user in a parallel relay network.

## VI. The Optimization of Version Age in Parallel Relay Networks

In this section, we consider the problem of minimizing the average version age in parallel relay networks. For this purpose, before considering the parallel relay problem, let us first consider the minimization of version age in a serially connected cache system as shown in Fig. 1(a). In Section VI-A, we first find the optimal update rates to minimize the version age for the serially connected network. We also utilize these rates for the optimization of the version age in parallel relay networks which we consider in Section VI-B. Here, we first show that it is better to consolidate update rates of the files to a single route, and propose an approximate solution to minimize the version age for the multi-cache system.

### A. Optimization of Version Age in Serially Connected Cache Systems

By using the SHS method provided in Section V-B based on recursive use of [50, Thm. 1], we write the version age of file $i$ at the user for the serially connected cache system as $\Delta_u(i)=\frac{\lambda_i}{u_i}+\Delta_{\{c\}\cup\{u\}}(i)=\frac{\lambda_i}{u_i}+\Delta_c(i)$, where $\Delta_c(i)$ denotes the version age of file $i$ at the cache and $\Delta_{\{c\}\cup\{u\}}(i)=\min\{\Delta_c(i),\Delta_u(i)\}=\Delta_c(i)$ as the user obtains its information only via the cache. Then, we find $\Delta_c(i)=\frac{\lambda_i}{c_i}$. Thus, the version age of file $i$ at the user for a single-cache system is $\Delta_u(i)=\frac{\lambda_i}{u_i}+\frac{\lambda_i}{c_i}$. We note that the version age at the user is equal to the sum of the version age at the cache $\frac{\lambda_i}{c_i}$ and the version age of the user with respect to the cache $\frac{\lambda_i}{u_i}$. Therefore, for a serially connected cache system, the version age becomes additive. Thus, for a serially connected $K$-cache system, the version age at the user is

$$\Delta_u(i)=\sum_{k=1}^{K}\frac{\lambda_i}{c_{ki}}+\frac{\lambda_i}{u_i}, \tag{61}$$

where the first term is equal to the version age at cache $k$. When we have a serially connected cache system, we see in (19) that the binary freshness at the user has a multiplicative structure whereas the version age in (61) has an additive structure.[4]

[4]If we use the traditional age of information metric for the serially connected $K$-cache system with dynamically changing source instead of a source which is updated according to a Poisson process, the average age of file $i$ at the user denoted by $\hat{\Delta}_u(i)$ becomes equal to $\hat{\Delta}_u(i)=\sum_{k=1}^{K}\frac{1}{c_{ki}}+\frac{1}{u_i}$ as in [52, (43)]. We note that we can obtain $\hat{\Delta}_u(i)$ by substituting $\lambda_i=1$ in the version age expression in (61). This result is intuitive as the traditional age metric increases linearly over time with a unit rate as if we have $\lambda_i=1$ in the case of version age. Thus, the version age and the traditional age metrics are closely related.

$$F_{\{u,k_1,k_2,\ldots,k_\ell\}}=\frac{\sum_{\bar{k}_{\ell+1}\in\{k_1,k_2,\ldots,k_\ell\}} c_{\bar{k}_{\ell+1}}}{\lambda+\sum_{\bar{k}_{\ell+1}\in\{k_1,k_2,\ldots,k_\ell\}} c_{\bar{k}_{\ell+1}}+\sum_{k_{\ell+1}\in\mathcal{K}-\{k_1,k_2,\ldots,k_\ell\}} u_{k_{\ell+1}}}+\frac{\sum_{k_{\ell+1}\in\mathcal{K}-\{k_1,k_2,\ldots,k_\ell\}} u_{k_{\ell+1}}F_{\{u,k_1,k_2,\ldots,k_\ell\}\cup\{k_{\ell+1}\}}}{\lambda+\sum_{\bar{k}_{\ell+1}\in\{k_1,k_2,\ldots,k_\ell\}} c_{\bar{k}_{\ell+1}}+\sum_{k_{\ell+1}\in\mathcal{K}-\{k_1,k_2,\ldots,k_\ell\}} u_{k_{\ell+1}}}, \tag{47}$$

Next, we formulate the problem of minimizing the version age for a single-cache system with $N$ files as follows

$$\begin{aligned} \min_{c_i,u_i} \quad & \sum_{i=1}^{N} \frac{\lambda_i}{u_i} + \frac{\lambda_i}{c_i} \\ \text{s.t.} \quad & \sum_{i=1}^{N} c_i \leq C, \\ & \sum_{i=1}^{N} u_i \leq U, \\ & c_i \geq 0, \quad u_i \geq 0, \ i = 1, \ldots, N, \end{aligned} \tag{62}$$

where $u_i$ and $c_i$ are the user and cache update rates, respectively, for file $i$ in a single-cache system. This formulation highlights some key differences between binary freshness and version age. Not only is the optimization problem in (62) convex, but it is also separable as the optimal $u_i$s and $c_i$s can be found independently of each other. This is not the case with the binary freshness metric where the objective function is non-convex and the update rates found via alternating maximization based method might not be globally optimal [44]. We separate the problem in (62), and write the problem for finding the optimal cache rates as follows

$$\begin{aligned} \min_{c_i} \quad & \sum_{i=1}^{N} \frac{\lambda_i}{c_i} \\ \text{s.t.} \quad & \sum_{i=1}^{N} c_i \leq C, \\ & c_i \geq 0, \quad i = 1, \ldots, N. \end{aligned} \tag{63}$$

Similarly, we can write an optimization problem to determine the optimal user update rates in (62). The solution to the optimization problem in (63) is (see also [34]),

$$c_j = \frac{C\sqrt{\lambda_j}}{\sum_{i=1}^{N}\sqrt{\lambda_i}}, \qquad u_j = \frac{U\sqrt{\lambda_j}}{\sum_{i=1}^{N}\sqrt{\lambda_i}}. \tag{64}$$

Thus, the optimal policy for the version age in a single-cache system is a square-root policy, which means that every file gets updated proportional to the square-root of the respective file update rate at the source.[5] In contrast, the policy for the binary freshness in [44] is a threshold based policy, where the files updated frequently at the source may not get updated by the cache and by the users.

### *B. Optimization of Version Age in Parallel Connected Cache Systems*

In this section, we provide an approximate solution for the problem of minimizing the version age in a parallel connected cache system shown in Fig. 1(b). In the following lemma, we show that the version age of a file is smaller in the single-cache system compared to the multi-cache system when the total update rates are kept the same.

[5]Even for arbitrarily connected networks, whenever a cache is serially connected to a preceding cache, the version age becomes additive, and thus, the optimization problem becomes separable. Therefore, for these caches, independent of file update rates, or the connection types in the preceding cache layers, the optimal updating policy for the serially connected caches is to update the files that are updated in the preceding cache based on the square-root policy obtained in this work.

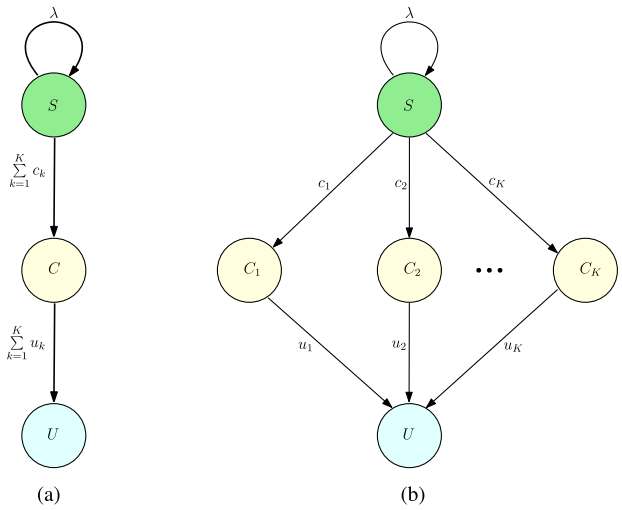


Fig. 5. Update rates (a) for a single-cache system, and (b) for a parallel multi-cache system considered in Lemma 5.

*Lemma 5:* Version age of a file in a $K$-cache system with update rates at the cache $(c_1, c_2, \ldots, c_K)$, and at the user $(u_1, u_2, \ldots, u_K)$ is higher than the freshness in a single-cache system with update rates at the cache $\sum_{k=1}^{K} c_k$, and at the user $\sum_{k=1}^{K} u_k$.

*Proof:* We demonstrate the caching systems considered in this lemma in Fig. 5, where we drop subscript $i$ for file $i$ in the remainder of the section for ease of exposition. We denote $\Delta_S^{single}$ and $\Delta_S^{multi}$ as the version age of a set $S$ in a single-cache system shown in Fig. 5(a) and in a multi-cache system shown in Fig. 5(b), correspondingly. As all the rates for a file are consolidated in a single route, such that the user rate in the route is $\sum_{k=1}^{K} u_k$ and the cache rate is $\sum_{k=1}^{K} c_k$, as in Fig. 5(a), the version age expression at the cache is

$$\Delta_c^{single} = \frac{\lambda}{\sum_{k=1}^{K} c_k}, \tag{65}$$

and the version age at the user is

$$\Delta_u^{single} = \frac{\lambda}{\sum_{k=1}^{K} u_k} + \frac{\lambda}{\sum_{k=1}^{K} c_k}. \tag{66}$$

Next, let us consider the $K$-cache system where route $k$ has user rate $u_k$ and cache rate $c_k$. Then, by using the SHS method developed in Section V-B employing [50, Thm. 1] recursively, we have

$$\Delta_u^{multi} = \frac{\lambda}{\sum_{k=1}^{K} u_k} + \frac{\sum_{k=1}^{K} u_k \Delta_{\{u,c_k\}}^{multi}}{\sum_{k=1}^{K} u_k}. \tag{67}$$

Note that for all $k$, $\Delta_{\{u,c_k\}}^{multi} = \min\{\Delta_{c_k}^{multi}, \Delta_u^{multi}\} \geq \min\{\Delta_{c_1}^{multi}, \Delta_{c_2}^{multi}, \ldots, \Delta_{c_K}^{multi}, \Delta_u^{multi}\} = \Delta_{\{c_1,\ldots,c_K,u\}}^{multi} = \Delta_c^{single}$ Thus, from (67), we have

$$\begin{aligned} \Delta_u^{multi} &\geq \frac{\lambda}{\sum_{k=1}^{K} u_k} + \Delta_{\{c_1,\ldots,c_K,u\}}^{multi} \\ &= \frac{\lambda}{\sum_{k=1}^{K} u_k} + \Delta_c^{single} = \Delta_u^{single}. \end{aligned} \tag{68}$$

Thus, the version age at the user is smaller when all rates are consolidated in a single route. ■

Specifically, comparing the version age for the two-cache system in (60) with the corresponding single-cache case having version age $\frac{\lambda}{u_1+u_2}+\frac{\lambda}{c_1+c_2}$ from (66), we observe that the two-cache case has an extra positive term, and hence, a higher version age.

Similar to the analysis applied to (20) in Section III, we rewrite (60) in terms of $u_1=\bar{u}-b$, $u_2=\bar{u}+b$, $c_1=\bar{c}-a$, and $c_2=\bar{c}+a$. Then, for given user rates $u_1$ and $u_2$, and the total cache rate $2\bar{c}$, we obtain $a_v^*$ that minimizes the version age as follows

$$a_v^*=\min\left\{b+\frac{\bar{c}+\bar{u}}{2b\bar{c}}\left(\bar{u}(2\bar{c}+\bar{u})-b^2 - \sqrt{(\bar{u}^2-b^2)((2\bar{c}+\bar{u})^2-b^2)}\right),\bar{c}\right\}. \quad (69)$$

Therefore, for given user rates $u_1$ and $u_2$, the optimal cache rates are $c_1^*=\bar{c}-a_v^*$ and $c_2^*=\bar{c}+a_v^*$. We use $a_v^*$ in (69) in the simulation results provided in Section VII. The expression obtained for $a_v^*$ in (69) is intuitive as the portion of third term dependent on $a$ and $b$ variables in (20) is similar to (60), with $\lambda$ replaced with 0. Note that there is no negative sign in the third term in (60), since our goal is to minimize version age, as opposed to maximizing the binary freshness in (20).

Lemma 5 shows that instead of updating a file over multiple routes, it is better to update it over a single route with the same total update rates at the cache and at the user. Parallel to the solution method proposed for the binary freshness in Section IV, here, we propose an approximate solution for the version age. First, for a single-cache system with the total update rates $C$ and $\sum_{k=1}^K U_k$, we find the optimal update rates at the cache and at the user based on the square-root policy obtained in Section VI-A. Then, we use the method employed in Section IV to distribute these optimal rates across the $K$ routes to get an approximate solution, with the objective of assigning each file to only a single route as much as possible.

Due to Lemma 5, the optimal solution obtained for a single-cache system lower bounds the optimal solution for the multi-cache system. Therefore, by looking at the difference between the optimum version age value for the single-cache system and the version age value as a result of the approximate solution proposed in this section, we can have an upper bound on the optimality gap for the multi-cache problem. In the following section, we provide numerical results for the binary freshness and version age metrics for multi-cache systems.

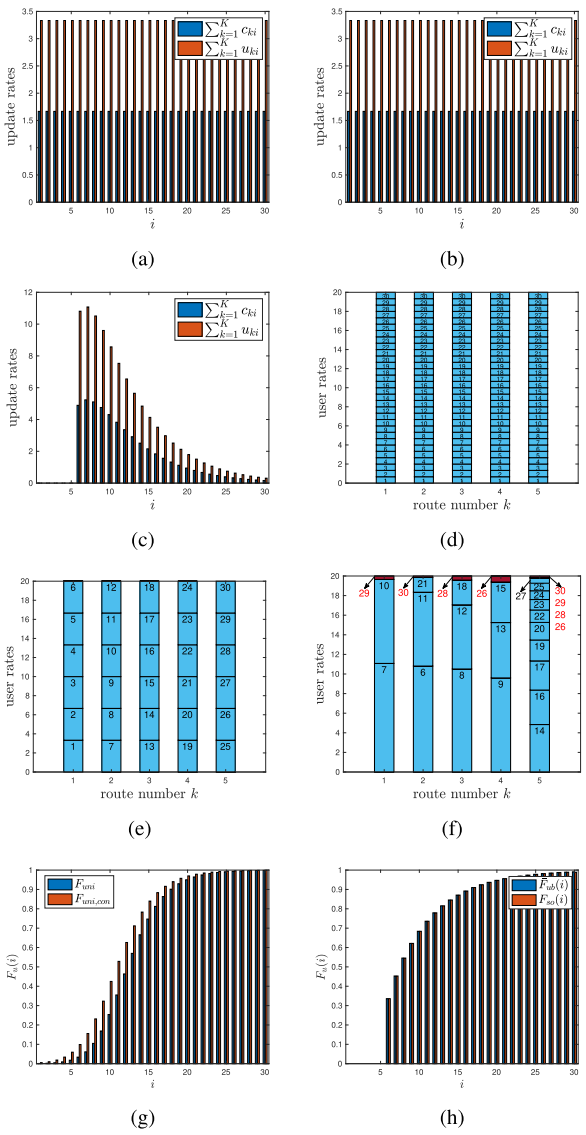


Fig. 6. Total user rates and total cache rates for the uniform update rate policies (a) without consolidation, (b) with consolidation, (c) total user rates and total cache rates obtained from the auxiliary solution. Route allocations for files (d) for the uniform update policy without consolidation, (e) for the uniform update policy with consolidation, and (f) for our proposed strategy (files with blue rates in single routes). Freshness obtained for the uniform update policy (g) with and without consolidating update rates into single cache, (h) for the single-cache and parallel cache systems.

## VII. Numerical Results

In the first numerical result, we choose number of routes $K=5$, number of files $N=30$, total update rate at source $C=50$ and at caches $U=100$ where each route has $U_k=20$. We use update arrival rates $\lambda_i=bq^i$ at the source for $i=1,\ldots,N$, where $b>0$, $q=0.7$, and $\sum_{i=1}^N\lambda_i=a$, with $a=100$. Note that since $q<1$, the update arrival rates at the source $\lambda_i$ decrease with the file index, i.e., $\lambda_i>\lambda_j$ for $i<j$.

In the first numerical result, we use the binary freshness metric. We apply alternating maximization approach described in [44] to solve the auxiliary single-cache problem to obtain total update rate for file $i$ at the user $\sum_{k=1}^K u_{ki}$ and at the caches $\sum_{k=1}^K c_{ki}$ as shown in Fig. 6(c). The total cache update rate constraint, i.e., $\sum_{k=1}^K\sum_{i=1}^N c_{ki}\leq C$, is already satisfied by both problems. In a parallel cache system, each route has its own total update rate constraint $\sum_{i=1}^N u_{ki}\leq U_k$, whereas the single-cache system has only one total update rate constraint for the user, i.e., $\sum_{k=1}^K\sum_{i=1}^N u_{ki}\leq U$ where $U=\sum_{k=1}^K U_k$. Thus, we need to choose the user rate allocation for all files in each route as described in Section IV, with corresponding cache rates found by (21) which are shown in Fig. 6(f). In order to compare the performance of our proposed algorithm, we consider two different cache updating schemes with uniform update rates. For both of these schemes, we choose the update rates as $\sum_{k=1}^K c_{ki}=\frac{C}{N}$, $\sum_{k=1}^K u_{ki}=\frac{U}{N}$ which is shown in Fig. 6(a)-(b). In the first updating scheme called uniform updating policy without consolidation, all the files are updated by all the caches as shown in Fig. 6(d), and

we denote the freshness obtained by this updating scheme by $F_{uni}$. In the second updating scheme called uniform updating policy with consolidation, we consolidate the file update rates into a single cache as shown in Fig. 6(e) and denote $F_{uni,con}$ as the freshness obtained by this scheme. We denote the freshness at the user for the single-cache system obtained by the method in [44] as $\bar{F}_{ub}$. We plot $F_{uni}$, $F_{uni,con}$ in Fig. 6(g) and $\bar{F}_{ub}$ and $F_{so}$ in Fig. 6(h). Among these three updating schemes, the uniform updating policy without consolidation performs the worst and the total freshness obtained by this scheme is $\sum_{i=1}^{N} F_{uni}(i) = 17.8004$. As we observed from Section III that consolidating update rates into a single cache increases the freshness. With the uniform updating policy with consolidation, the freshness obtained from all files is equal to $\sum_{i=1}^{N} F_{uni,con}(i) = 19.4495$. Lastly, if we further optimize the update rates among the files and use the proposed algorithm in Section IV, we obtain the total freshness as $\sum_{i=1}^{N} F_{so}(i) = 21.0692$ which significantly outperforms both of the uniform updating schemes. Even though we split the update rates among different routes for $K-1=4$ files that have some of the highest freshness (files 26, 28, 29, 30), their freshness loss is negligible as shown in Fig. 6(c). In this system, since $\bar{F}_{ub} = 21.0718$, the total freshness loss, i.e., $\bar{F}_{ub} - F_{so}$, is equal to $0.0026$, which is much smaller than the theoretical upper bound $0.5(K-1) = 2$.

To get more perspective on these freshness values, a more loose upper bound can be obtained by removing the relay-to-user update constraint rate $U$ in the single cache problem by setting $U = \infty$, which allows each $u_i = \infty$, thereby giving $\lim_{u_i \to \infty} F_u(i) = \lim_{u_i \to \infty} \frac{u_i}{\lambda_i + u_i} \frac{c_i}{\lambda_i + c_i} = \frac{c_i}{\lambda_i + c_i}$. Denoting freshness of this system model by $F_{u,d}(i)$, the corresponding freshness maximization problem for the system with no cache is a convex optimization problem stated as

$$\begin{aligned} \max_{\{c_i\}} \quad & \sum_{i=1}^{N} F_{u,d}(i) = \sum_{i=1}^{N} \frac{c_i}{\lambda_i + c_i} \\ \text{s.t.} \quad & \sum_{i=1}^{N} c_i \le C, \\ & c_i \ge 0, \quad i = 1, \ldots, N, \end{aligned} \tag{70}$$

for which the optimum update rates $c_i$s are given by $c_i = \lambda_i \left( \frac{1}{\sqrt{\theta \lambda_i}} - 1 \right)^+$, where $\theta > 0$ is the Lagrange multiplier satisfying total update rate constraint, i.e., $\sum_{i=1}^{N} c_i \le C$, and corresponding total optimum freshness is $F^*_{u,d} = 22.5096$, which is close to the total freshness obtained by our algorithm $F_{so}$. Note that from (7), for a set of 30 files, the maximum possible freshness value is 30, when $f_u(i,t) = 1$ for all $i$ and $t$, which is achieved when all constraints are relaxed. In that case $u_{ki} = c_{ki} = \infty$, and as a result files are instantly delivered from source through the caches to the user. In this example, we choose $\lambda_i$ as a monotonically decreasing function of $i$, which gives a diverse set of source update rates. Instead, if all files have the same update rate $\lambda_i = \frac{a}{N}$, $a = 100$, such that $\sum_i \lambda_i = a$ as before, we get $F_{uni,con} = F_{so} = F_{ub} = 5$, since the update rates $u_i$, $c_i$ are the same for all files and all these policies support consolidation. However, $F_{uni} = 2.0072$ is lower due to lack of rate consolidation across routes as a result of the uniform updating policy without consolidation. This further supports the key insights of our work.

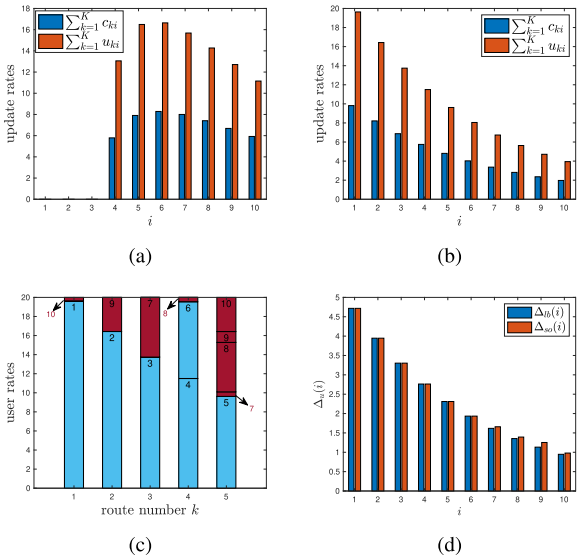


Fig. 7. Total user rates and total cache rates obtained from the auxiliary solution (a) with binary freshness, and (b) with version age. (c) Route allocations for files (files with blue rates in single routes) with version age, and (d) version age for the single-cache and parallel cache systems.

In the second numerical result, we consider the version age metric for a multi-cache system. We choose number of files $N = 10$, and keep the rest of the parameters the same as in the first numerical result. We repeat the process of finding the update rates of files in the first numerical result, with the difference that we use (64), which is the square-root policy found in Section VI-A, to solve the auxiliary single-cache problem. This difference is highlighted in Fig. 7(a) and Fig. 7(b). We see that in order to maximize the binary freshness, the cache and the user may not update the files that are frequently changing at the source as shown in Fig. 7(a), whereas in order to minimize the version age, each file is updated at the cache and at the user proportional to the square-root of file change rate at the source shown in Fig. 7(b). As a result, although both metrics measure the information freshness in different ways, we see that depending on the metric selection, the optimal rate allocation policy may differ significantly. We provide the user rate allocation in Fig. 7(c) where 4 files (files 7, 8, 9, 10) are split across two routes. For the files that are updated across two caches, we have derived the specific formula in (60) which we use in this numerical result to find the corresponding version age, with corresponding cache rates found by (69). The single-cache system gives the lower bound for the version age denoted by $\Delta_{lb} = \sum_{i=1}^{n} \Delta_{lb}(i)$, where $\Delta_{lb}(i)$ is the version age corresponding to file $i$. Let the approximate policy give the sub-optimal version age $\Delta_{so} = \sum_{i=1}^{n} \Delta_{so}(i)$. We plot $\Delta_{lb}$ and $\Delta_{so}$ in Fig. 7(d). In this numerical result, the total version age increase, i.e., $\Delta_{so} - \Delta_{lb}$, is equal to $0.2334$, which is small compared to the $\Delta_{lb} = 24.0257$ and $\Delta_{so} = 24.2591$.

## VIII. Conclusion

In this work, we considered timeliness of files in a parallel cache network via the binary freshness and version age metrics. First, for the binary freshness metric, we derived a closed form expression for the system and observed that

freshness for a file is maximum when it does not have to split its rates across multiple routes. We further analyzed that when number of files is large, most files are restricted to single routes in the optimal update policy. We used this observation to find an approximate policy for assigning rates to all files, by solving the alternate problem of a single-cache system. We further found an upper bound for the gap between the optimal policy and the approximate policy. Next, with the help of the SHS approach, we derived a closed form expression for the version age of information of the system, and determined the counterpart properties in a parallel relay network. Numerical results showed that the proposed sub-optimal policy closely approximates the optimal policy for both timeliness metrics.

## REFERENCES


[1] E. Najm, R. Yates, and E. Soljanin, "Status updates through M/G/1/1 queues with HARQ," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 131–135.

[2] A. Soysal and S. Ulukus, "Age of information in G/G/1/1 systems," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Nov. 2019, pp. 2022–2027.

[3] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1807–1827, Mar. 2019.

[4] P. Zou, O. Ozel, and S. Subramaniam, "Optimizing information freshness through computation–transmission tradeoff and queue management in edge computing," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 949–963, Apr. 2021.

[5] S. Farazi, A. G. Klein, and D. R. Brown, "Average age of information for status update systems with an energy harvesting server," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops*, Apr. 2018, pp. 112–117.

[6] X. Wu, J. Yang, and J. Wu, "Optimal status update for age of information minimization with an energy harvesting source," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 1, pp. 193–204, Mar. 2018.

[7] A. Baknina, O. Ozel, J. Yang, S. Ulukus, and A. Yener, "Sending information through status updates," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2271–2275.

[8] S. Leng and A. Yener, "Age of information minimization for an energy harvesting cognitive radio," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 2, pp. 427–439, Jun. 2019.

[9] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor, "Age-minimal transmission for energy harvesting sensors with finite batteries: Online policies," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 534–556, Jan. 2020.

[10] M. A. Abd-Elmagid, H. S. Dhillon, and N. Pappas, "A reinforcement learning framework for optimizing age of information in RF-powered communication systems," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4747–4760, Aug. 2020.

[11] N. Pappas, Z. Chen, and M. Hatami, "Average AoI of cached status updates for a process monitored by an energy harvesting sensor," in *Proc. 54th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2020, pp. 1–5.

[12] A. Arafa and S. Ulukus, "Timely updates in energy harvesting two-hop networks: Offline and online policies," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 4017–4030, Jun. 2019.

[13] J. Cho and H. Garcia-Molina, "Effective page refresh policies for web crawlers," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 390–426, Dec. 2003.

[14] A. Kolobov, Y. Peres, E. Lubetzky, and E. Horvitz, "Optimal freshness crawl under politeness constraints," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2019, pp. 495–504.

[15] E. T. Ceran, D. Gunduz, and A. Gyorgy, "A reinforcement learning approach to age of information in multi-user networks," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2018, pp. 1967–1971.

[16] M. Bastopcu and S. Ulukus, "Minimizing age of information with soft updates," *J. Commun. Netw.*, vol. 21, no. 3, pp. 233–243, Jun. 2019.

[17] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling policies for minimizing age of information in broadcast wireless networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2637–2650, Dec. 2018.

[18] Y.-P. Hsu, "Age of information: Whittle index for scheduling stochastic arrivals," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 2634–2638.

[19] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff, "Age-optimal sampling and transmission scheduling in multi-source systems," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 121–130.

[20] E. Ozfatura, B. Buyukates, D. Gunduz, and S. Ulukus, "Age-based coded computation for bias reduction in distributed learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[21] H. H. Yang, A. Arafa, T. Q. S. Quek, and H. V. Poor, "Age-based scheduling policy for federated learning in mobile edge networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8743–8747.

[22] N. Rajaraman, R. Vaze, and G. Reddy, "Not just age but age and quality of information," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1325–1338, May 2021.

[23] M. Bastopcu and S. Ulukus, "Age of information for updates with distortion: Constant and age-dependent distortion constraints," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2425–2438, Dec. 2021.

[24] O. Ayan, M. Vilgelm, M. Klugel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proc. 10th ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, Apr. 2019, pp. 109–117.

[25] S. Banerjee, R. Bhattacharjee, and A. Sinha, "Fundamental limits of age-of-information in stationary and non-stationary environments," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 1741–1746.

[26] R. D. Yates, P. Ciblat, A. Yener, and M. Wigger, "Age-optimal constrained cache updating," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 141–145.

[27] 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," in *Proc. IEEE WiOpt*, Jun. 2020, pp. 1–6.

[28] J. Liu, X. Wang, B. Bai, and H. Dai, "Age-optimal trajectory planning for UAV-assisted data collection," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 553–558.

[29] M. A. Abd-Elmagid and H. S. Dhillon, "Average peak age-of-information minimization in UAV-assisted IoT networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 2003–2008, Feb. 2019.

[30] M. Bastopcu and S. Ulukus, "Timely group updating," in *Proc. CISS*, Mar. 2021, pp. 1–6.

[31] B. Buyukates, A. Soysal, and S. Ulukus, "Age of information scaling in large networks with hierarchical cooperation," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[32] B. Buyukates, A. Soysal, and S. Ulukus, "Age of information in multihop multicast networks," *J. Commun. Netw.*, vol. 21, no. 3, pp. 256–267, 2019.

[33] M. Wang, W. Chen, and A. Ephremides, "Reconstruction of counting process in real-time: The freshness of information through queues," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[34] M. Bastopcu and S. Ulukus, "Who should Google scholar update more often?" in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 696–701.

[35] Y. Sun, Y. Polyanskiy, and E. Uysal-Biyikoglu, "Remote estimation of the Wiener process over a channel with random delay," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 321–325.

[36] M. Bastopcu and S. Ulukus, "Timely tracking of infection status of individuals in a population," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–7.

[37] J. Yun, C. Joo, and A. Eryilmaz, "Optimal real-time monitoring of an information source under communication costs," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 4767–4772.

[38] C. Kam, S. Kompella, and A. Ephremides, "Age of incorrect information for remote estimation of a binary Markov source," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 1–6.

[39] J. Chakravorty and A. Mahajan, "Remote estimation over a packet-drop channel with Markovian state," *IEEE Trans. Autom. Control*, vol. 65, no. 5, pp. 2016–2031, Jul. 2020.

[40] P. Mayekar, P. Parag, and H. Tyagi, "Optimal source codes for timely updates," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3714–3731, Jun. 2020.

[41] M. Bastopcu, B. Buyukates, and S. Ulukus, "Selective encoding policies for maximizing information freshness," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5714–5726, Sep. 2021.

[42] D. Ramirez, E. Erkip, and H. V. Poor, "Age of information with finite horizon and partial updates," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 4965–4969.
[43] B. Buyukates and S. Ulukus, "Timely distributed computation with stragglers," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5273–5282, Jun. 2020.
[44] M. Bastopcu and S. Ulukus, "Information freshness in cache updating systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1861–1874, Mar. 2021.
[45] M. Bastopcu and S. Ulukus, "Maximizing information freshness in caching systems with limited cache storage capacity," in *Proc. 54th Asilomar Conf. Signals, Syst., Comput.*, Nov. 2020, pp. 423–427.
[46] 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 *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2018, pp. 1–6.
[47] W. Gao, G. Cao, M. Srivatsa, and A. Iyengar, "Distributed maintenance of cache freshness in opportunistic mobile networks," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, Jun. 2012, pp. 132–141.
[48] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Information freshness and popularity in mobile caching," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 136–140.
[49] Y. Gu, Q. Wang, H. Chen, Y. Li, and B. Vucetic, "Optimizing information freshness in two-hop status update systems under a resource constraint," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1380–1392, May 2021.
[50] R. D. Yates, "The age of gossip in networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 2984–2989.
[51] B. Abolhassani, J. Tadrous, A. Eryilmaz, and E. Yeh, "Fresh caching for dynamic content," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.
[52] R. D. Yates, "The age of information in networks: Moments, distributions, and sampling," *IEEE Trans. Inf. Theory*, vol. 66, no. 9, pp. 5712–5728, Sep. 2020.
[53] B. Buyukates, M. Bastopcu, and S. Ulukus, "Age of gossip in networks with community structure," in *Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Sep. 2021, pp. 326–330.
[54] M. Bastopcu, B. Buyukates, and S. Ulukus, "Gossiping with binary freshness metric," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–6.
[55] A. Kosta, N. Pappas, and V. Angelakis, "Age of information: A new concept, metric, and tool," *Found. Trends Netw.*, vol. 12, no. 3, pp. 162–259, Nov. 2017.
[56] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," *Synth. Lectures Commun. Netw.*, vol. 12, no. 2, pp. 1–224, Dec. 2019.
[57] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.
[58] S. Zhang, N. Zhang, X. Fang, P. Yang, and X. S. Shen, "Self-sustaining caching stations: Toward cost-effective 5G-enabled vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 202–208, Nov. 2017.
[59] Z. Gu, H. Lu, M. Zhang, H. Sun, and C. W. Chen, "Association and caching in relay-assisted mmWave networks: A stochastic geometry perspective," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 8316–8332, Dec. 2021.
[60] B. T. Bacinoglu, E. T. Ceran, and E. Uysal-Biyikoglu, "Age of information under energy replenishment constraints," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Feb. 2015, pp. 25–31.

**Priyanka Kaswan** (Student Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, in 2017. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Maryland, College Park. Her research interests include information freshness, gossip algorithms, resource allocation, and low latency wireless communication systems.

**Melih Bastopcu** (Member, IEEE) received the B.S. degree in electrical and electronics engineering from Bilkent University, Turkey, in 2016, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, USA, in 2020 and 2021, respectively. He is currently a Post-Doctoral Research Associate with the Coordinated Science Laboratory, University of Illinois Urbana–Champaign. His research interests include age of information, resource allocation, low latency systems, communication network design, dissemination of misinformation, and social networks.

**Sennur Ulukus** (Fellow, IEEE) received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University and the Ph.D. degree in electrical and computer engineering from the Wireless Information Network Laboratory (WINLAB), Rutgers University. She is currently the Anthony Ephremides Professor in information sciences and systems with the Department of Electrical and Computer Engineering, University of Maryland, College Park, where she also holds a joint appointment with the Institute for Systems Research (ISR). Prior to joining UMD, she was a Senior Technical Staff Member at the AT&T Laboratories-Research. Her research interests are in information theory, wireless communications, machine learning, and signal processing and networks; with recent focus on private information retrieval, age of information, machine learning for wireless, distributed coded computing, group testing, physical layer security, energy harvesting communications, and wireless energy and information transfer.

She is also a Distinguished Scholar-Teacher of the University of Maryland. She received the 2003 IEEE Marconi Prize Paper Award in Wireless Communications, the 2019 IEEE Communications Society Best Tutorial Paper Award, the 2020 IEEE Communications Society Women in Communications Engineering (WICE) Outstanding Achievement Award, the 2020 IEEE Communications Society Technical Committee on Green Communications and Computing (TCGCC) Distinguished Technical Achievement Recognition Award, the 2005 NSF Career Award, the 2011 ISR Outstanding Systems Engineering Faculty Award, and the 2012 ECE George Corcoran Outstanding Teaching Award. She was a Distinguished Lecturer of the IEEE Information Theory Society (2018–2019). She has been an Area Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS since 2019 and a Senior Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING since 2020. She was an Area Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING (2016–2020), an Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS-SERIES ON GREEN COMMUNICATIONS AND NETWORKING (2015–2016), an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY (2007–2010), and an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS (2003–2007). She was a Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (2008, 2015, and 2021), *Journal of Communications and Networks* (2012), and the IEEE TRANSACTIONS ON INFORMATION THEORY (2011). She is also the TPC Chair of 2021 IEEE GLOBECOM and was the TPC Co-Chair of 2019 IEEE ITW, 2017 IEEE ISIT, 2016 IEEE GLOBECOM, 2014 IEEE PIMRC, and 2011 IEEE CTW.