Freshness Based Cache Updating in Parallel Relay Networks

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

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 the 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 policy. Numerical results show that the sub-optimal policy closely approximates the optimal policy.

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 obsoleteness of data. AoI has been considered in a wide range of contexts, such as queueing networks, energy harvesting systems, web crawling, scheduling problems, remote estimation, UAV systems and so on [1]–[49].

The works that are most closely related to our work here are [40]–[49]. In [40], 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 [40] assumes constant file update durations, which is extended in [41] by considering file update durations to be dependent on the size and the age of the files. While [40], [41] use the AoI metric, reference [42] uses a binary freshness metric in a caching system, and determines the optimum update rates at the user and the cache. [42] also extends the approach to a cascade sequence of cache nodes, and [43] generalizes it to the case of nodes with limited cache capacity. Here, we further generalize [42] to a more complex network which is composed of parallel caches.

Other related work that use caching and relaying techniques for freshness include: [44] where a tradeoff between content freshness and service latency from the aspect of mobile edge caching is studied; [45] which considers caching policies in opportunistic networks; [46] 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; [47] where an optimal policy is derived when the current rate of requests for a file is dependent on both history of requests and the freshness of the file; [48] which considers a two-hop status update system where an optimal scheduling policy is determined by a constrained Markov decision process approach; and [49] where a two-hop system with energy harvesting at source and relay nodes is considered.

In this paper, we consider a parallel network with multiple cache routes between a source and a user (Fig. 1). We first derive a closed-form expression for freshness at the user. 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, 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. Numerical results show that the proposed sub-optimal policy closely approximates the optimal policy.

Fig. 1. System model for a parallel multi-cache system.
II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a system with a source, \( K \) parallel relays and a user, as shown in Fig. 1. The source has the most up-to-date versions of a library of \( N \) files. File \( i \) is updated at the source with exponential inter-update times with rate \( \lambda_i \). The source updates file \( i \) at cache \( k \) with exponential inter-update times with rate \( c_{ki} \). Cache \( k \) updates file \( i \) at the user with exponential inter-update times with rate \( u_{ki} \). There is no delay or information loss in any source-cache links or cache-user links. However, the source is subject to a total update rate constraint \( \sum_{k=1}^{K} \sum_{j=1}^{N} c_{ki} \leq C \), and cache \( k \) is subject to a 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 when the source updates the file which is given by

\[
\mathbb{E}[W_u] = \mathbb{E}[\mathbb{E}[W_u|T_u]].
\]

For a given update cycle duration \( T_u = t \) at the source, \( W_c \) is exponentially distributed with rate \( c \), either takes a value in between \( 0 \) and \( T_u \), or takes value \( T_u \), i.e., \( W_c = T_u \), and in this case, the cache is not updated in that cycle. Thus, we have

\[
\mathbb{E}[W_c|T_u = t] = \int_0^t x e^{-ct} dx + \int_t^\infty t e^{-ct} dx = \frac{1 - e^{-ct}}{c}. \quad (4)
\]

Then, we obtain \( \mathbb{E}[W_c] \) as

\[
\mathbb{E}[W_c] = \int_0^\infty \left( 1 - \frac{e^{-ct}}{c} \right) \lambda e^{-\lambda t} dt = \frac{1}{\lambda + c}. \quad (5)
\]

By using (5), we obtain \( \mathbb{E}[T_c] = \mathbb{E}[T_u] - \mathbb{E}[W_c] \) as

\[
\mathbb{E}[T_c] = \frac{c}{\lambda(\lambda + c)}. \quad (6)
\]

Finally, by substituting (6) into (3), we obtain \( F_c(i) \) as

\[
F_c(i) = \frac{c}{\lambda + c}. \quad (7)
\]

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

\[
f_u(i, t) = \begin{cases} 1, & \text{if file } i \text{ at the user is fresh at time } t, \\ 0, & \text{otherwise.} \end{cases} \quad (8)
\]

Once file \( i \) is updated at the cache after \( W_c(j) \), and the same file is updated at the user after \( W_u(j) \), 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 \( T_u = W_c + W_u \). We denote \( F_u(i) \) as the long term average freshness of the file \( i \) at the user which is given by

\[
F_u(i) = \frac{\mathbb{E}[T_u]}{\mathbb{E}[T_c]}, \quad (9)
\]

where \( T_u \) denotes the typical random variable for \( T_u(j) \).

First, we find \( \mathbb{E}[W_u] \) by using nested expectations. When the \( j \)th update arrives at the source, due to memoryless property of the exponential distribution, \( W_c \) and \( W_u \) are exponentially distributed with rates \( c \) and \( u \), respectively. Hence, the distribution of \( W_u \) denoted by \( f_{W_u}(x) \) is equal to the convolution of the distributions of \( W_c \) and \( W_u \) which is given by

\[
f_{W_u}(x) = \frac{cu}{c-u} (e^{-ux} - e^{-cx}), \quad 0 \leq x < \infty. \quad (10)
\]

For a given update cycle duration \( T_u = t \) at the source, the
total waiting time $W_u$ with pdf in (10) 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,

$$E[W_u | T_s = t] = \frac{cu}{c - u} \left( \frac{1}{u^2} - \frac{1}{c^2} \right).$$ (11)

By using $E[W_u] = E[E[W_u | T_s]]$, we obtain $E[W_u]$ as

$$E[W_u] = \frac{\lambda + c + u}{(\lambda + u)(\lambda + c)}. \quad (12)$$

Then, we obtain $E[T_u] = E[T_s] - E[W_u]$ as

$$E[T_u] = \frac{uc}{\lambda(\lambda + u)(\lambda + c)}. \quad (13)$$

Finally, by substituting (13) into (9), we obtain $F_u(i)$ as

$$F_u(i) = \frac{E[T_u]}{E[T_s]} = \frac{u}{\lambda + u} \frac{c}{\lambda + c}, \quad (14)$$

which is equal to the freshness expression in (42). Above, we have provided an alternative method (to (42)) 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. For simplicity, we drop file index $i$ from all variables. 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 of the caches. The file at the cache is updated after $W_{ck}$ duration. The cache $k$ updates the same file at the user after $W_{uk}$ duration. We denote the random variable $X_k = W_{ck} + W_{uk}$ as the total waiting time for cache $k$ to send a successful update to the user. As $W_{ck}$ and $W_{uk}$ are exponentially distributed with rates $c_k$ and $u_k$, respectively, similar to (10), we have $f_{X_k}(x) = \frac{c_k u_k}{c_k - u_k} (e^{-u_k x} - e^{-c_k x})$ for $x \geq 0$. For a given update cycle $T_s = t$, the user is updated after $W_u$, given by

$${W_u} = \min\{t, X_1, X_2, \ldots, X_K\}, \quad (15)$$

where $W_u = t$ denotes the case where the user is not updated in that update cycle. The cdf of $X_k$ is given by

$$P(X_k > x) = \begin{cases} \frac{c_k - u_k}{c_k - u_k} (e^{-u_k x} - e^{-c_k x}) / c_k, & x \geq 0, \\ 1, & x < 0. \end{cases} \quad (16)$$

Since $W_u$ takes only positive values, $E[W_u]$ can be found by integrating its cdf, i.e., $E[W_u | T_s = t] = \int_0^t P(W_u > x) dx$,

$$E[W_u | T_s = t] = \int_0^t P(X_1 > x) \cdots P(X_K > x) dx. \quad (17)$$

For ease of exposition, let $p_v = \{p_k\}_{k \in [K]}$ be the file at cache $k$. Then, $E[W_u | T_s = t]$ equals

$$\prod_k c_k \prod_k u_k \left( \sum_{p_v \in V_p} \frac{(-1)^{S_c} \left( 1 - e^{-t(S_c p_v)} \right)}{(\sum_k p_k) \prod_k p_k} \right). \quad (18)$$

Next, we find $E[W_u] = E[E[W_u | T_s]]$ as

$$E[W_u] = \prod_k c_k \prod_k u_k \left( \sum_{p_v \in V_p} \frac{(-1)^{S_c} \left( 1 - e^{-t(S_c p_v)} \right)}{(\sum_k p_k) \prod_k p_k} \right). \quad (19)$$

Since $E[T_u] = E[T_s] - E[W_u]$, we find $F_u(i)$ in (9) as

$$F_u(i) = 1 - \prod_k c_k \prod_k u_k \left( \sum_{p_v \in V_p} \frac{(-1)^{S_c} \left( 1 - e^{-t(S_c p_v)} \right)}{(\sum_k p_k) \prod_k p_k} \right). \quad (20)$$

We note that when $K = 1$, i.e., single-cache system, the user freshness in (20) reduces to the expression in (14). When $K = 2$, i.e., two-cache system, the user freshness in (20) reduces to the expression in (21) at the top of the next page. Interestingly, comparing (14) and (21), 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 (21).

**III. STRUCTURE OF THE OPTIMAL POLICY**

In this section, we find the optimum update rate allocation structure for general $K$ and $N$. 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 $i$, 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. We define the average variables as $\bar{u} = u_1 + u_2$ and $\bar{c} = c_1 + c_2$, and deviation from the average as $b = u_1 - u_2$ and $a = c_1 - c_2$. 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 and $N = 2$ files, for given user rates $u_1$ and $u_2$, and the total cache rate $2\bar{c}$, the optimal cache rates are equal to $c_1^* = \bar{c} - a^*$ and $c_2^* = \bar{c} + a^*$ where

$$a^* = \min \left\{ \frac{b + (\bar{c} + \lambda + \bar{u})}{b(2\lambda + 2\bar{c})}, \frac{b + (\bar{c} + \lambda + \bar{u})}{b(2\lambda + 2\bar{c})} - \sqrt{\left(\frac{b}{b(2\lambda + 2\bar{c})}\right)^2 - \frac{b^2}{(2\lambda + 2\bar{u})^2}} \right\}. \quad (23)$$

**Proof:** We prove the lemma by writing (21) equivalently as (22) after inserting $\bar{u}, \bar{c}, a$ and $b$. 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 the term inside the parentheses with respect to $a$ yields the first part of the min in (23). As this critical point yields $\frac{\partial^2 F_u(i)}{\partial a^2} < 0$, we conclude that $a^*$ in (23) maximizes the freshness at the user. We note that $\frac{\partial^2 F_u(i)}{\partial a^2} \geq 0$, and thus, $a^*$ increases monotonically with $b$, till it reaches $\bar{c}$, after which $a^*$ is equal to $\bar{c}$, yielding (23).
As an aside, we remark that in (23) we have \(a^* \geq b\) as long as \(a^* < \tilde{c}\), that is, for a deviation \(b\) of \(u_1\), \(u_2\) from their average \(\overline{u}\), the optimal \(a\) yields a bigger deviation for \(c_1^*, c_2^*\) from their average \(\tilde{c}\).

Next, we define \(F_u(i)\) 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 (21). Note that \(F_u(i)\) is a function of \(b, \tilde{u}\) and \(c\). 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 \(F_u(i)\) increases.

**Lemma 2** \(F_u(i)\) is an increasing function of \(b\).

We prove Lemma 2 by showing \(\frac{dF_u(i)}{db} > 0\). 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 \(\tilde{c}_i = \frac{c_i + u_i}{2}\) and \(\tilde{u}_i = \frac{u_i + u_{i+1}}{2}\) for \(i = 1, 2\), which fixes total user rates and total cache rates. Similarly, we have \(u_{1i} = \tilde{u}_i - b_i\) and \(u_{2i} = \tilde{u}_i + b_i\) 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 \(u'_{11} = \tilde{u}_1 - b_1 - \delta_1, u'_{21} = \tilde{u}_1 + b_1 + \delta_1, u'_{12} = \tilde{u}_2 - b_2 + \delta_2\) and \(u'_{22} = \tilde{u}_2 + b_2 - \delta_2\) 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. 3.

In the first case, increasing \(b_1\) for one file leads to increasing \(b_1\) value for the other file as shown in Fig. 3(a). As distributions of user rates for both files become lopsided simultaneously, it is a win-win situation for both files. For this case, we increase \(b_1\) values of files till one file is completely in a single route. For example, in Fig. 3(a), the user rates for the second file (shown in yellow) are \(\tilde{u}_2 - b_2\) and \(\tilde{u}_2 + b_2\) in route 1 and route 2, respectively. Then, we increase \(b_1\) till \(\tilde{u}_2 - b_2 = 0\) 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 \(b_1\) value of one file decreases \(b_1\) value of the other file. This case is shown in Fig. 3(b) where both files have larger user update rates in route 2. In order to determine which file to prioritize, we compare \(\frac{dF_u(i)}{db}\) for both files. If \(\frac{dF_u(1)}{db} > \frac{dF_u(2)}{db}\), then we prioritize improving freshness of file 1. One can show that \(\frac{dF_u(i)}{db} > 0\). Thus, the increase in freshness of file 1 is always larger than the decrease in freshness of file 2. Similarly, if \(\frac{dF_u(2)}{db} > \frac{dF_u(1)}{db}\), 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 \(\tilde{u}_1, \tilde{u}_2, \tilde{c}_1, \tilde{c}_2\) 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 \((u_1 + u_2, u_3, \ldots, u_K)\).

**Proof:** With notation of Section 11 since freshness \(F_u(i) = 1 - \lambda E[W_s], \) where \(E[W_u] = \int_0^\infty E[W_s|T_s = t]e^{-M}dt\), we prove the lemma by showing \(E[W_u|T_s = t] - E[W_u|T_s = t] \geq 0\), where \(K\) in \(W_u^K\) denotes \(K\)-cache system. Thus, given total update rates \(\sum_{k=1}^K u_{ki}\) and \(\sum_{k=1}^K c_{ki}\) for a 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. APPROXIMATE SOLUTION**

The freshness maximization problem for our system is,

\[
\max_{c_{ki}, u_{ki}} \sum_{i=1}^{N} F_u(i) \quad \text{s.t.} \sum_{k=1}^{K} c_{ki} \leq C
\]

This parallel-cache problem is significantly more complex than the cascade-cache problem in (22). A Lagrangian approach as in (22) seems prohibitive as it results in highly nonlinear KKT
conditions. We pursue an approximate solution approach utilizing the properties of the optimal solution found in Section 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 2 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 (24) as \( F^* \), we have

\[
F_{so} < F^* < F_{ub}
\]  

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 \( b_i = \bar{a}_i \) (\( a^* = \bar{c} \)). Hence, using (22), we find an upper bound on maximum freshness loss ratio \( \rho \) possible for a file due to splitting,

\[
\rho = \frac{F_u(i) \mid (b_i, a^*_i) = (0, \bar{c}) - F_u(i) \mid (b_i, a^*_i) = (0, 0)}{F_u(i) \mid (b_i, a^*_i) = (0, \bar{c})},
\]  

which is equal to \( \rho = \frac{\lambda_i}{2(\lambda_i + c_i)} < 0.5 \). Since \( F_{ub}(i) < 1 \), the optimality gap is \( F^* - F_{so} \leq \rho \) \( K = 1 \) files have low \( u_i \), owing to very high or very low \( \lambda_i \), as observed in (22). In the former case, \( F_{ub}(i) \) is low, while in the latter case, \( \rho \) is very low.

V. Numerical Results

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 \( \lambda_i = b q^i \) 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 the update arrival rates at the source \( \lambda_i \) decrease with the file index since \( q < 1 \).

We apply alternating maximization approach described in (22) 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 \( \sum_{k=1}^{K} c_k \), as shown in Fig. 4(a). The total cache update rate constraint, i.e., \( \sum_{k=1}^{K} \sum_{i=1}^{N} c_k \leq C \), 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 4 with corresponding cache rates found by (23) which are shown in Fig. 4(b). We denote the freshness at the user for the single-cache system obtained by the method in (22) as \( \bar{F}_u \). We plot \( \bar{F}_u \) and \( F_{so} \) in Fig. 4(c). Even though we split the update rates among different routes for \( K = 4 \) files that have some of the highest freshness (files 26, 28, 29, 30), their freshness loss is negligible as shown in Fig. 4(c). In this system, the total freshness loss, i.e., \( F_{ub} - F_{so} \), is equal to 0.0026, which is much smaller than the theoretical upper bound 0.5(\( K - 1 \)) = 2.
