Fractional Dynamic Caching: Minimizing the File Delivery Time under Limited Backhaul

Liumeng Wang, Sheng Zhou
Tsinghua National Laboratory for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
Email: wlm14@mails.tsinghua.edu.cn, sheng.zhou@tsinghua.edu.cn

***Abstract*—To reduce the average file delivery time under limited backhaul bandwidth, a fractional dynamic caching scheme, that coordinates the utilization of the backhaul and the cache storage, is proposed in this paper. In our scheme, only parts of the files are pre-fetched and stored in the static storage segment of small base stations (SBSs), while the remaining parts of the files will be fetched into the dynamic storage segment when requested by users. We aim to minimize the average file delivery time by investigating the optimal pre-fetching policy, including the static caching size of each file and the size of the dynamic storage segment. We first derive the average file delivery time given the size of the dynamic storage segment, and obtain the corresponding optimal pre-fetching policy. The optimal size of the dynamic storage segment is then derived in closed-form. Numerical results show that with our scheme, the average file delivery time can be substantially reduced.**

***Index Terms*—Caching, file delivery time, limited backhaul**

I. Introduction

To meet the ever-increasing need for the broadband wireless access [1], more and more small base stations (SBSs) are expected to be deployed [2]. However, it is expensive and sometimes difficult to deploy high-bandwidth backhaul links between SBSs and macro base stations (MBSs) or core-network servers, and thus low cost solutions such as wireless backhauling are promising candidates to support dense SBSs [3]. However, on the other hand, the limited backhaul bandwidth may degrade users' quality of service (QoS), such as increasing the file delivery time.

Exploiting the fact that many users in the network may have common interests on files/contents, and the cost of storages is kept getting lower [4], proactive caching is a promising way to reduce the file delivery time under limited backhaul [5]. Frequently requested files can be pre-fetched to the SBSs' storage with low transmission rate. Upon users' requests, the pre-fetched files can be delivered to the users directly from the SBSs without downloading from the core network, and thus the delivery rates of the files to the users can overcome the bandwidth limitation of the backhaul. Via proactive push and further exploiting the renewable energy harvesting, caching can also reduce the network energy consumption [6]. Recently, caching is also applied in the cloud radio access network (C-RAN) based on fog computing [7], which will potentially relax the fronthaul bandwidth requirements.

Given the storage size on SBSs, file allocation to the caches of SBSs determines the performance of caching, e.g., average file downloading time, file hitting ratio, and etc. If few files are frequently requested by users, caching the most popular files brings good performance [8]. Otherwise, if files have similar request frequencies, distinct files should be cached by different SBSs [9]. In these caching schemes, only a limited number of files can be cached, while other files should be delivered over the backhaul upon request with possibly large delivery time. Other schemes cache parts of the files, but the remaining parts will be fetched from the MBS or the servers after the cached parts are delivered [9]. This has low backhaul utilization efficiency and may have long file delivery time, because the backhaul is actually idle when the SBS is transmitting the cached parts.

In our scheme, we divide the storage of SBSs into two parts: the static storage segment and the dynamic storage segment. The static storage segment will be used to cache parts of the frequently requested files, and the remaining parts of the files will be fetched and cached into the dynamic storage segment while the files are requested and transmitting to the users. By doing so, our scheme not only increases the number of files cached at the SBS, but also enhance the bandwidth usage efficiency of the backhaul. As a result, the average file delivery time can be substantially reduced, which is confirmed by our analysis and simulation studies. We also explore the optimal caching policy to minimize the average file delivery time given the static caching portion of each file and the size of the dynamic storage segment. We then derive the closed-form optimal size of the dynamic storage segment, and the corresponding static caching portion of each file.

The paper is organized as follows. The system model is described in Section II. The average file delivery time is derived in Section III, and then the optimal pre-fetching policy is investigated in Section IV. The numerical results are presented in Section V. We conclude the paper in Section VI.

II. System Model

Assume that each user is served by one SBS, only one file is requested by some user served by the SBS once a time, and there is no overlap between the service time of different users. The SBSs can cache the contents from a library with F files, and the size of each file is L bits. Labeling the files in the descending order of popularity, $f \in \{1, 2, ..., F\}$, where the popularity is represented by the probability P_f that file f is requested by a user, follows Zipf distribution [10], i.e.,

978-1-5090-3009-5/17/$31.00 ©2017 IEEE

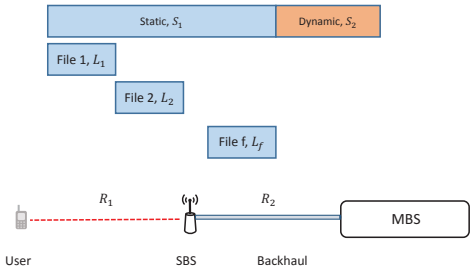

Fig. 1. The fractional dynamic caching scheme with static storage segment and dynamic storage segment.

$P_f = cf^{-\gamma}$, where $\gamma \geq 0$ is the given popularity exponent and c is a normalized factor.

As illustrated in Fig. 1, we divide the storage of the SBS into two segments. The first segment is the static storage with size of S_1 bits, into which $L_f \leq L$ bits of file f are pre-fetched. The second segment is dynamic storage with size of S_2 bits, into which the remaining part of file f is dynamically fetched when file f is requested by the user. The total storage is $S = S_1 + S_2$ bits. We have the storage limitation that $S \leq FL$, otherwise, all files can be statically cached at the SBS, which is a trivial case ignored in our analysis.

The transmission rate between the SBS and the user is R_1, while the transmission rate of the backhaul link is R_2. As SBSs are densely deployed and thus close to the users, while the backhaul is in general limited due to deployment costs and sometimes due to the wireless realization of the backhaul [3], we assume that $R_2 < R_1$. The cached parts of the requested file in both the static and dynamic storages are delivered to the user with rate R_1 over the wireless link from the SBS to the user. The uncached parts of the requested file will be prefetched into the dynamic storage with rate R_2 if the dynamic storage is not fully utilized. However, if all the cached part has already been delivered to the user, the fetched file will be forwarded to the user with R_2 instead because in this case the bottleneck is the backhaul.

III. Minimizing the Average File Delivery Time

Denote T_f as the delivery time of a file f. Considering the case that all the cached part of a file f is delivered to the user just when the dynamic storage is fully used with the remaining part (i.e., other than the static part L_f) of the requested file f, we have $(L_f + S_2)/R_1 = S_2/R_2$, i.e., $S_2 = R_2 L_f/(R_1 - R_2)$. Furthermore, if $L_f + S_2 = L$, the whole file f can be cached at the SBS when being requested, and then $L_f = (R_1 - R_2)L/R_1$. Given L_f and S_2, we can express the file delivery time T_f as

$$T_f = \begin{cases} L/R_1 & L_f > \frac{(R_1-R_2)L}{R_1}, S_2 \geq L - L_f \\ \frac{L_f+S_2}{R_1} + \frac{L-L_f-S_2}{R_2} & L_f > \frac{(R_1-R_2)L}{R_1}, S_2 < L - L_f \\ \frac{L-L_f}{R_2} & L_f \leq \frac{(R_1-R_2)L}{R_1}, S_2 \geq \frac{R_2 L_f}{R_1-R_2} \\ \frac{L_f+S_2}{R_1} + \frac{L-L_f-S_2}{R_2} & L_f \leq \frac{(R_1-R_2)L}{R_1}, S_2 < \frac{R_2 L_f}{R_1-R_2} \end{cases}, \quad (1)$$

where the first case is that the dynamic storage is large enough to cache all the remaining part of the file, while the static caching size for file f, i.e., L_f, is large enough so that there is enough time (since $R_2 < R_1$) to fetch the remaining part of the file to the dynamic storage, and thus the whole file can be cached at SBS and delivered to the user with rate R_1 continuously; the second case is that the dynamic storage is not large enough to cache all the remaining part of the file, while L_f is large enough to provide enough time to fulfill the dynamic storage, so that $L_f + S_2$ bits will be delivered to the user with rate R_1 while the remaining $L - L_f - S_2$ bits can only be delivered to the user with rate R_2; the third case is that only $R_2 L_f/(R_1 - R_2)$ bits of the dynamic storage will be used as L_f is not large enough, so that only $R_1 L_f/(R_1 - R_2)$ bits will be delivered to the user with rate R_1 while the remaining $L - R_1 L_f/(R_1 - R_2)$ bits can only be delivered to the user with rate R_2; the fourth case is that the dynamic storage can be fulfilled but the total storage is not enough to cache the whole file, so that $L_f + S_2$ bits will be delivered to the user with rate R_1 while the remaining $L - L_f - S_2$ bits will be delivered to the user with rate R_2.

The average file delivery time $\overline{T}$ is

$$\overline{T} = \sum_{f=1}^{F} P_f T_f. \quad (2)$$

Our objective is to minimize the average file delivery time $\overline{T}$ given the total storage capacity S at the SBS. The optimization problem is thus formulated as

$$\min_{L_f, S_2} \overline{T} \quad (3)$$

$$\text{s.t. } S_2 + \sum_{f=1}^{F} L_f \leq S, \quad (4)$$

where (4) is the constraint of total storage S. The objective function is convex but not differentiable due to the piecewise nature of (1). Subgradient method can be adopted to solve the optimization problem, but the convergence is slow [11], and does not bring much insights. In what follows, we will analyze the structure of the optimal solution.

IV. Optimal Pre-fetching Policy

Utilizing some unique properties of the delivery time T_f and the file popular probability P_f, the optimization problem can be simplified and the optimal pre-fetching policy can be found more efficiently.

First, according to (1), for any given L_f, when $S_2 > \frac{R_2 L}{R_1}$, the delivery time T_f will not decrease with S_2. For any scheme with $S_2 > \frac{R_2 L}{R_1}$, there exists a new scheme with $\hat{S}_2 = \frac{R_2 L}{R_1} < S_2$, and $\hat{L}_f = L_f$, where $\hat{T}_f = T_f$, i.e., the new scheme has the same average file delivery time with the previous one. We can thus reduce the feasible region to be:

$$S_2 \leq \frac{R_2 L}{R_1}. \quad (5)$$

Second, given S_2, T_f will not decrease with L_f. If $L_i > L_j$ for any $i > j$, $T_i \leq T_j$. A new scheme can be designed as $\hat{S}_2 = S_2$, $\hat{L}_i = L_j$, $\hat{L}_j = L_i$, $\hat{L}_f = L_f$ for $f \neq i$ and $f \neq j$, i.e., we only exchange the static caching size of file i and file j. We can thus get that $\hat{T}_i = T_j$, $\hat{T}_j = T_i$, while $\hat{T}_f = T_f$ for $f \neq i$ and $f \neq j$. Comparing the average file delivery time of the two schemes, we have

$$\begin{aligned}&\sum_{f=1}^{F} P_f \hat{T}_f - \sum_{f=1}^{F} P_f T_f \\ &= P_j T_i + P_i T_j - P_j T_j - P_i T_i \\ &= (P_i - P_j)(T_j - T_i) \leq 0, \end{aligned} \tag{6}$$

i.e., the average delivery time of the newly designed scheme is no larger than the previous one. Therefore, we can further reduce the feasible region to satisfy:

$$L_{f+1} \leq L_f, \tag{7}$$

which is similar to the most popular caching scheme, the more popular file will be allocated with larger static caching size.

Third, for any given $S_2 \leq R_2 L / R_1$, when $L_f = L - S_2$, $T_f = L/R_1$, it is not necessary to allocate more than $L - S_2$ bits static caching size for file f. For any scheme, if the static caching size of file $\hat{f}$ satisfies $L_{\hat{f}} > L - S_2$, we can design a new scheme with $\hat{L}_{\hat{f}} = L - S_2$, $\hat{S}_2 = S_2$, while $\hat{L}_f = L_f$ ($f \neq \hat{f}$). In the new scheme, $\hat{T}_f = T_f$, this new scheme has the same average file delivery time with the previous one. Thus the feasible region can be further reduced to satisfy

$$L_f + S_2 \leq L. \tag{8}$$

Lemma 1. *The optimal pre-fetching solution satisfies*

$$S_2 = \frac{R_2}{R_1 - R_2} L_1, \tag{9}$$

where L_1 is the static storage allocated for the most popular file.

Proof: For any optimal scheme, if $S_2 > R_2 L_1/(R_1 - R_2)$, i.e., the dynamic storage is large enough compared with the static caching size for file 1, $T_f = (L - L_f)/R_2$. we can design a new scheme where $\hat{S}_2 = R_2 L_1/(R_1 - R_2) < S_2$, and $\hat{L}_f = L_f$ for any f. As $L_1 \geq L_f$, we have $\hat{S}_2 \geq R_2 \hat{L}_f/(R_1 - R_2)$, and we can thus have $\hat{T}_f = (L - \hat{L}_f)/R_2 = T_f$, i.e., the two schemes have the same average file delivery time.

On the other hand, if $S_2 < R_2 L_1/(R_1 - R_2)$, i.e., the dynamic storage is not enough given the static caching size for file 1,

$$T_1 = (L_1 + S_2)/R_1 + (L - L_1 - S_2)/R_2. \tag{10}$$

We design a new scheme where $\hat{S}_2 = R_2(L_1 + S_2)/R_1$, $\hat{L}_1 = (R_1 - R_2)(L_1 + S_2)/R_1$, and $\hat{L}_f = L_f$ for $f \geq 2$. According to (1), the delivery time of file 1 in the new scheme is

$$\hat{T}_1 = (\hat{L}_1 + \hat{S}_2)/R_1 + (L - \hat{L}_1 - \hat{S}_2)/R_2 = T_1, \tag{11}$$

and as $\hat{S}_2 \geq S_2$ while $\hat{L}_f = L_f$, $\hat{T}_f \leq T_f$ for $f \geq 2$, i.e., the average delivery time of the new scheme is no larger than the original one. Thus we can draw the conclusion that for any scheme that $S_2 \neq \frac{R_2}{R_1 - R_2} L_1$, we can find a scheme which satisfies (9) and has no worse performance. ■

With the refined constraints (5), (7), (8) and (9), and they are correspond to case 3 in (1), the expression of the delivery time of file f can thus be simplified as

$$T_f = \frac{L - L_f}{R_2}. \tag{12}$$

According to (5), (7) and (9), we have

$$S_2 + \sum_{f=1}^{F} L_f \leq S_2 + F L_1 \leq \frac{(F(R_1 - R_2) + R_2)L}{R_1}, \tag{13}$$

where $\frac{(F(R_1 - R_2) + R_2)L}{R_1}$ is minimum total storage to achieve the minimum average delivery time $\frac{L}{R_1}$. We will focus on the scenario that $S \leq \frac{(F(R_1 - R_2) + R_2)L}{R_1}$ in the following analysis.

The optimization problem is then simplified as

$$\max_{L_f} \sum_{f=1}^{F} P_f L_f \tag{14}$$

$$\text{s.t. } \frac{R_1}{R_1 - R_2} L_1 + \sum_{f=2}^{F} L_f \leq S \tag{15}$$

$$\frac{R_1}{R_1 - R_2} L_1 \leq L \tag{16}$$

$$L_{f+1} \leq L_f. \tag{17}$$

Note that according to Lemma 1, $S_2 + L_1 = R_1 L_1/(R_1 - R_2)$, and thus (15) is the constraint of the total storage at the SBS, (16) is an equivalent to (8). This is a linear programming problem, which can be efficiently solved with computational complexity $\mathcal{O}(F^3)$.

Furthermore, utilizing the popularity property that $P_{f+1} \leq P_f$, we can find an optimal pre-fetching policy that determines L_f given S_2.

Proposition 2. *Given the size of the dynamic storage segment size S_2, the optimal L_f satisfies*

$$L_f^* = \begin{cases} L_1^* & 1 \leq f \leq F_{th} \\ S - S_2 - F_{th} L_1^* & f = F_{th} + 1 \\ 0 & F_{th} < f \leq F \end{cases} \tag{18}$$

where $F_{th} = \lfloor (S - S_2)/L_1^* \rfloor$, and according to Lemma 1, $L_1^* = (R_1 - R_2) S_2 / R_2$.

Proof: For any scheme satisfies $F_{th} \leq F$, i.e., all of the static storage is used to cache the files,

$$\begin{aligned}&\sum_{f=1}^{F} P_f L_f^* - \sum_{f=1}^{F} P_f L_f \\ &\geq P_{F_{\text{th}}+1} \sum_{f=1}^{F_{\text{th}}} (L_f^* - L_f) + \sum_{f=F_{\text{th}}+1}^{F} P_f (L_f^* - L_f) \\ &\geq P_{F_{\text{th}}+1} \sum_{f=1}^{F} (L_f^* - L_f) \geq 0. \end{aligned} \tag{19}$$

On the other hand, if $F_{\text{th}} > F$,

$$\sum_{f=1}^{F} P_f L_f^* - \sum_{f=1}^{F} P_f L_f = \sum_{f=1}^{F} P_f (L_f^* - L_f) \geq 0,$$

combining with (19), we can draw the conclusion that the average file delivery time of the pre-fetching policy in (18) is no larger than any other policy. ∎

With Proposition 2, the key of the optimization problem (14) is to find the optimal size of the dynamic storage S_2^*, which is expressed in the following theorem.

Theorem 3. *Define a function of file popular probability as*

$$g(m) = \sum_{f=1}^{m-1} (P_f - P_m) - \frac{R_2}{R_1 - R_2} P_m, \tag{20}$$

where $1 \leq m \leq F$ is an integer. If $g(F) > 0$, there exists a threshold $M < F$ satisfies that $g(M) \leq 0$ and $g(M+1) > 0$, the optimal size of dynamic storage segment S_2^* satisfies

$$S_2^* = \begin{cases} \frac{R_2 S}{M(R_1 - R_2) + R_2} & S < \frac{(M(R_1 - R_2) + R_2)L}{R_1} \\ \frac{R_2 L}{R_1} & S \geq \frac{(M(R_1 - R_2) + R_2)L}{R_1} \end{cases}. \tag{21}$$

Otherwise, if $g(F) \leq 0$,

$$S_2^* = \frac{R_2 S}{F(R_1 - R_2) + R_2}. \tag{22}$$

Proof: First, we will prove the scenario that $g(F) > 0$. When $S < (M(R_1 - R_2) + R_2)L/R_1$, for any scheme with $S_2 < \frac{R_2 S}{M(R_1 - R_2) + R_2} < \frac{R_2 L}{R_1}$, there should be $L_{M+1} > 0$. We can thus design a new scheme, where $\hat{S}_2 = S_2 + \frac{R_2 L_{M+1}}{R_2 + M(R_1 - R_2)}$, $\hat{L}_f = L_f + \frac{(R_1 - R_2)L_{M+1}}{R_2 + M(R_1 - R_2)}$ for $1 \leq f \leq M$, $\hat{L}_{M+1} = 0$, and $\hat{L}_f = L_f$ for $f > M + 1$. Comparing the average file delivery time of the two schemes, we have

$$\begin{aligned} &\sum_{f=1}^{F} P_f \hat{L}_f - \sum_{f=1}^{F} P_f L_f \\ &= \sum_{f=1}^{M} P_f (\hat{L}_f - L_f) - P_{M+1} L_{M+1} \\ &= \sum_{f=1}^{M} P_f \frac{(R_1 - R_2)L_{M+1}}{R_2 + M(R_1 - R_2)} - P_{M+1} L_{M+1} \\ &= \frac{(R_1 - R_2)L_{M+1}}{R_2 + M(R_1 - R_2)} g(M+1) > 0, \end{aligned} \tag{23}$$

i.e., the average file delivery time of the newly designed scheme is smaller than the previous one, and thus there should be $S_2^* \geq \frac{R_2 S}{M(R_1 - R_2) + R_2}$.

On the other hand, for any scheme that satisfies $S_2 > \frac{R_2 S}{M(R_1 - R_2) + R_2}$, we have $F_{th} < M$. We can then design a new scheme by decreasing S_2, where $\hat{S}_2 = S_2 - \frac{l R_2}{R_2 + F_{\text{th}}(R_1 - R_2)}$, $\hat{L}_f = L_f - \frac{l(R_1 - R_2)}{R_2 + F_{\text{th}}(R_1 - R_2)}$ for $1 \leq f \leq F_{\text{th}}$, $\hat{L}_{F_{\text{th}}+1} = L_{F_{\text{th}}+1} + l$, and $\hat{L}_f = 0$ for $f > F_{\text{th}} + 1$, where $l = \frac{(R_1 - R_2)S}{(F_{th}+1)(R_1 - R_2) + R_2} - L_{F_{\text{th}}+1}$. Comparing the average file delivery time of the two schemes, we have

$$\begin{aligned} &\sum_{f=1}^{F} P_f \hat{L}_f - \sum_{f=1}^{F} P_f L_f \\ &= -\sum_{f=1}^{F_{th}} P_f \frac{(R_1 - R_2)l}{R_2 + F_{th}(R_1 - R_2)} + P_{F_{th}+1} l \\ &= -\frac{(R_1 - R_2)l}{R_2 + F_{th}(R_1 - R_2)} g(F_{th} + 1) \geq 0, \end{aligned} \tag{24}$$

i.e., the average file delivery time of the newly designed scheme is no larger than the previous one, and thus there should be $S_2^* \leq \frac{R_2 S}{M(R_1 - R_2) + R_2}$. We can thus get that when $S < \frac{(M(R_1 - R_2) + R_2)L}{R_1}$, an optimal pre-fetching policy leads to $S_2^* = \frac{R_2 S}{M(R_1 - R_2) + R_2}$.

When $S > \frac{(M(R_1 - R_2) + R_2)L}{R_1}$, if $S_2 < \frac{R_2 L}{R_1}$,

$$\begin{aligned} &\sum_{f=1}^{F} P_f L_f^* - \sum_{f=1}^{F} P_f L_f \\ &\geq \sum_{f=1}^{M} P_f (L_1^* - L_1) - P_{M+1}(M(L_1^* - L_1) + S_2^* - S_2) \\ &= (L_1^* - L_1) \sum_{f=1}^{M} (P_f - P_{M+1}) - P_{M+1}(S_2^* - S_2) \\ &= (L_1^* - L_1) g(M+1) > 0, \end{aligned} \tag{25}$$

we can thus draw the conclusion that $S_2^* = \frac{R_2 L}{R_1}$.

Second, if $g(F) \leq 0$, we can assume that there is an additional file $F + 1$ with popularity probability $P_{F+1} = 0$, then we have $g(F+1) > 0$, i.e., $M = F$, the proof is similar to the first case. ∎

Note that the first part in the expression of $g(M)$ is the gain brought by increasing the the static caching size of the $M - 1$ most popular files while decreasing the the corresponding static caching size of the M-th popular file, the second part is the cost of increasing the dynamic storage to satisfy the constraint of (9). If $g(M) < 0$, we should guarantee that L_f should be same as the static caching size of any more popular file. On the other hand, if $g(M) > 0$, no static storage should be allocated to the M-th popular file when $S_2 < \frac{R_2 L}{R_1}$. As $g(M)$ is an increasing function, with a given popularity distribution, it is easy to find the particular M that satisfies $g(M) \leq 0$ and $g(M+1) > 0$ with methods like binary search, and the computation complexity is $\mathcal{O}(F \log F)$.

V. Numerical Results

We set that there are totally $F = 1000$ files in the library, and the size of each file is $L = 100$Mb. The transmission rate of the backhaul is assumed to be $R_2 = 100$Mbps, while the transmission rate between the user and the SBS is assumed to be $R_1 = 1.5R_2$, $R_1 = 2R_2$ and $R_1 = 3R_2$ to investigate the effect of the ratio between R_1 and R_2. The popularity exponent γ is assumed to be 0.2, 0.5 and 1 to explore the

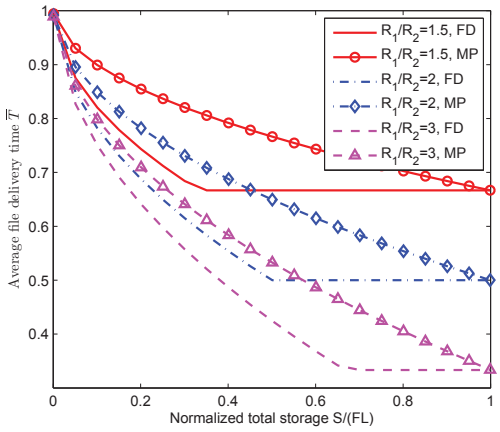

Fig. 2. Average file delivery time versus the normalized total storage with different R_1/R_2, the popularity exponent $\gamma = 0.5$.

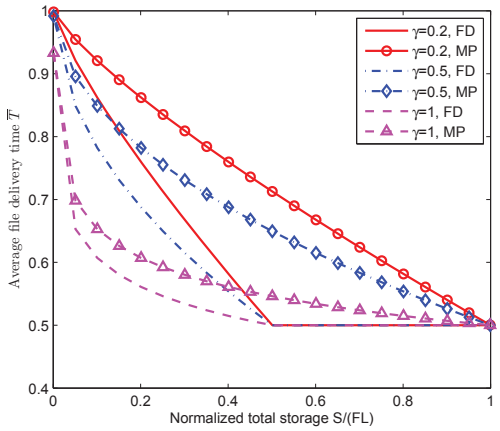

Fig. 3. Average file delivery time versus the normalized total storage with different popularity exponent γ, $R_1/R_2 = 2$.

effect of the popularity distributions. The most popular (MP) caching scheme is adopted as the baseline to compare with our proposed fractional dynamic (FD) caching scheme.

We normalize the total storage S as $S/(FL)$, where FL is the least required storage by the most popular caching scheme to achieve the minimum average delivery time, which is L/R_1. Average file delivery time w.r.t. the normalized total storage is shown in Fig. 2 when $R_1/R_2 = 1.5, 2, 3$ and $\gamma = 0.5$. We can see that with our scheme, the average file delivery time can be substantially reduced, about 23% average file delivery time is reduced when $R_1/R_2 = 2$ and $S/(FL) = 0.5$. We also find that the smaller R_1/R_2 is, the smaller is the least required total storage to achieve the minimum file average delivery time in our scheme, and more benefits can be brought. The reason is that according to (21), with smaller R_1/R_2, more dynamic storage should be allocated, and thus more storage can be reused when users request different files.

We further analyze the effect of the popularity exponent γ on the average file delivery time. As shown in Fig. 3, with smaller γ, i.e., the popularity is more dispersed, in which case our scheme can bring more benefits. The reason is that more files can be cached in our scheme, and with larger popularity exponent, the additional cached files have larger requested probability, and thus more performance gain can be obtained.

VI. Conclusions

In this paper, we have proposed a fractional dynamic caching scheme that effectively coordinate the backhaul and cache storage at the SBS. To minimize the average file delivery time by optimizing the allocation of the static caching size of each file and the size of dynamic storage segment, we first derive the expression of the delivery time of each file. We then find the optimal static caching sizes for all files, given the size of the dynamic storage segment. Finally, the closed-form expression of the optimal size of the dynamic storage segment is derived. Numerical results show that our scheme can greatly reduce the average file delivery time, especially when the radio transmission rate R_1 and the backhaul transmission rate R_2 are close, or when the popularity of the files is more dispersed.

Acknowledgment

This work is sponsored in part by the Nature Science Foundation of China (No. 61461136004, No. 61571265, No. 91638204), and Intel Collaborative Research Institute for Mobile Networking and Computing.

References

[1] "Cisco Visual Networking Index: Global mobile data traffic forecast update, 2015-2020," 2016.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[3] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: challenges and research advances," *IEEE Network*, vol. 28, no. 6, pp. 6–11, 2014.

[4] N. Golrezae, A. F. Molisch, and A. G. Dimakis, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, 2013.

[5] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[6] S. Zhou, J. Gong, Z. Zhou, W. Chen, and Z. Niu, "GreenDelivery: proactive content caching and push with energy-harvesting-based small cells," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 142–149, 2015.

[7] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog computing based radio access networks: Issues and challenges," *IEEE Network*, vol. 30, no. 4, pp. 46–53, 2016.

[8] M. Tao, E. Chen, H. Zhou, and W. Yu, "Content-centric sparse multicast beamforming for cache-enabled cloud RAN," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6118–6131, 2016.

[9] S.-H. Park, O. Simeone, and S. Shamai (Shitz), "Joint optimization of cloud and edge processing for fog radio access networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7621–7632, 2016.

[10] X. Peng, J.-C. Shen, and J. Zhang, "Backhaul-aware caching placement for wireless networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[11] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *notes for EE392o Stanford University Autumn*, 2013-2014.