Design of Caching Content Replacement in Base Station Assisted Wireless D2D Caching Networks

Ming-Chun Lee, *Student Member, IEEE*, Hao Feng, and Andreas F. Molisch, *Fellow, IEEE*

Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, CA, USA
Email: mingchul@usc.edu; haofeng@usc.edu; molisch@usc.edu

***Abstract*—Due to the concentrated popularity distribution of video files, caching of popular files on devices, and distributing them via device-to-device (D2D) communications allows a dramatic increase in the throughput of wireless video networks. However, since the popularity distribution is not static and the caching policy might be outdated, there is a need for replacement of cache content. In this work, by exploiting the broadcasting of the base station (BS), we model the caching content replacement in BS assisted wireless D2D caching networks and propose a practically realizable replacement procedure. Subsequently, by introducing a queuing system, the replacement problem is formulated as a sequential decision making problem, in which the long term average service rate is optimized under average cost constraint and queue stability. We propose a replacement design using Lyapunov optimization, which effectively solves the problem and makes decisions. Using simulations, we evaluate the proposed design. The results clearly indicate that, when dynamics exist, the systems exploiting replacement can significantly outperform the systems using merely the static policy.**

I. Introduction

Demand of wireless traffic has increased dramatically during the past several years and will continue its growth. Among numerous wireless applications, delivery of video content accounts for the majority of data traffic; how to support this application is one of the challenges for 4G and 5G systems [1]. Conventional throughput-enhancing approaches, such as massive antenna systems, network densification, and millimeter wave systems, all rely on obtaining more physical resources and/or increasing investment into infrastructure, which are generally expensive. In contrast, with the rapid development of semiconductor technology, memory has become the cheapest hardware resource. This motivates the exploitation of content caching in both wireline and wireless networks [1], [2]. The idea is to trade memory for bandwidth by caching files in the off-peak hours and then using the cached files during the peak hour. This idea combined with the asynchronous content reuse and concentrated popularity distribution of video content renders caching in wireless networks a promising solution for satisfying video traffic demand [1], [3].

Video caching at the wireless edge for video distribution was first proposed in [3]. That paper exploits the caching in femtocell base stations (BSs) to enhance the network throughput. The idea of femtocaching was then generalized to different types of BSs and to heterogeneous network environments with multiple antennas [4], [5]. To benefit from the memory inherently included in user devices and the recent progress of high performance device-to-device (D2D) communications, use of content caching in wireless D2D networks was first proposed in [1], which also suggested to augment the control (and supply rare video files) through assistance by the BS. Subsequently, different caching strategies and different aspects of the networks were then proposed and discussed in the literature, e.g., [6]–[12]. Results showed that by combining caching with D2D communications, performance can be improved significantly when compared with conventional unicasting.

The main challenge in caching networks is deciding which file should be cached by whom. This is by and large considered as caching policy design problem. Although there are many papers investigating different aspects of this problem using different approaches, the main emphasis generally lies in the determination of a static policy based on network statistics, i.e., the same caching policy is used throughout the whole network and over the whole time horizon without considering specific dynamics. In contrast to this, benefits can be obtained by conducting dynamic caching content replacement/refreshment. The motivations are: (i) the popularity distribution can change with time (e.g., emergence of a new viral video) and space (e.g., recording of different sports teams are popular in different cities); and (ii) the caching realizations of the network could be inappropriate (e.g., users do not cache files according to the designated policy).[1] Such dynamics of the network can degrade the performance of the network that uses a static caching policy, whereas adaptations (cache replacement) can automatically compensate.

Cache replacement in user devices has its history in the Computer Science community [13], [14], which generally considers individual replacement and/or networks with special properties without considering D2D cooperation. Although the content caching replacement was implicitly used in [15], its focus was on joint content delivery and caching design in a given time slot, which is obviously different from our goal which is to dynamically refresh the caching content. In [16], the device cache refreshment is investigated using a Markov

[1]A further source is that mobility of users could result in changes of the locally available user density and cached files, e.g., due to movement of the devices, though this is beyond the scope of this paper.

decision process (MDP). However, its focus was on efficient buffering for a single user, and it ignored the important multi-user situation and D2D communications of the networks. In [17], the problem that how users can "reactively" update their caching content is investigated. This is different from our aim that we want to "proactively" update the caching content. To our best knowledge, dynamic content caching and replacement in wireless D2D caching networks has not been well investigated yet. This paper aims to close this gap.

A. Contributions of this paper

In this work, we consider a network with dynamic popularity distribution. Our goal is to develop an approach to decide whether to conduct a replacement action and which files in users' caches to be replaced by what files. We first propose a network structure to realize the content replacement action via introducing the file broadcasting and a queuing system. Since the replacement action incurs a cost and the stability of the system is necessary, our replacement problem then aims to maximize the time-average service rate subject to a time-average cost constraint and queue stability.

The replacement problem includes two parts: deciding which files to newly cache on devices, and deciding what files should be replaced, i.e., deleted from the caches. The joint design of these two problems is extremely challenging. We thus propose an effective but heuristic procedure for the second part of the problem, i.e., which files to delete from the caches. Most of the work concentrates on the first part, i.e., deciding which file to push into the caches for which we formulate a sequential decision-making optimization problem. To solve this problem, we propose to use Lyapunov optimization [18]. By introducing a virtual cost queue, the proposed approach makes decisions to myopically minimize the drifts of the queues. Such minimization can be implemented with extremely low complexity, and we can show that the replacement actions suggested by this approach can stabilize queues and satisfy the cost constraint. We use simulations to demonstrate the efficacy of the proposed replacement approaches. Simulations using our two-step procedure show that compared to algorithms based on the assumption of static popularity, significant improvement in dynamic environments can be observed. Our main contributions are:

- We elaborate the replacement problem in wireless D2D networks and propose to use the BS with broadcasting and a centrally controlled queueing system to realize the replacement procedure.
- We propose a fundamental replacement procedure. Based on this procedure, we formulate the replacement problem in the form of a sequential decision-making problem with the time-average cost constraint and queue stability.
- We solve the replacement problem by incorporating effective heuristics and concepts of Lyapunov optimization. This design can appropriately use system state and historical records to conduct replacement and is implemented with extremely low complexity. Besides, the proposed design can satisfy the time-average constraints asymptotically and stabilize the queuing system.
- We use computer simulations to validate the proposed designs and show that the dynamic cache replacement can significantly improve the networks. We use different example systems to demonstrate the proposed designs and obtain insights of the replacement actions.

B. Organization of this paper

The remainder of the paper is organized as follows. In Sec. II, the system model is elaborated. In Sec. III, we discuss the dynamic cache replacement and provide the basic replacement procedure. The replacement design is proposed in Sec. IV. We numerically evaluate the performance in Sec. V. Conclusions are provided in Sec. VI.

II. System Model

In this work, we consider a BS assisted wireless D2D caching network, in which the BS can broadcast files to user devices for caching content replacement. To focus on the performance of D2D caching, we consider users to be served only via D2D networks and broadcast without using user-specific BS links, i.e., user requests can only be satisfied in three ways: files accessible via D2D communications, files in their own caches, and broadcast files. We consider a centrally-controlled scheduling for D2D networks, in which the BS serves as the central controller to collect requests from users, schedule D2D communications, and make the decisions for replacement actions.[2] Besides, we assume the central controller has the ability to observe all requests. Thus, even if a request of the user can be immediately satisfied by self-caching, the controller in the BS can still observe this request.[3]

A library consisting of M files is considered. We assume user devices to be able to cache only a single file of the library[4]. We consider a homogeneous request probability model with $a_m(t)$ to describe the probability of a user to request file m at time t. To describe the caching result of the network at time t, we denote $b_m(t)$ as the caching popularity of file m, defined by

$$b_m(t) = \frac{N_m(t)}{\sum_{n=1}^{M} N_n(t)}, \tag{1}$$

where $N_m(t)$ is the number of times the file m is cached in the D2D network at time t. By definition, $0 \leq b_m(t) \leq 1$ then indicates the probability of file m being cached in a user device of the network. We consider both active and inactive users, in which the active users are defined as the users sending requests and participating in the D2D communications; the

[2]Some additional signaling might be necessary for realizing the replacement scheme. However, the amount of signaling bits is typically much smaller than the number of bits in a video file. In any case, signaling overhead could be included as the cost each time we conduct a replacement action (see. Sec. III).

[3]In [20], we present modifications that eliminate the need for this (slightly impractical) assumption.

[4]The extension to networks where each device can cache multiple files will be discussed in the Journal version [20].

inactive users are defined as the users that do not send any request but, depending on the scenario assumptions, may or may not participate in the D2D communications.

A queuing system at the BS with M queues, where queue m stores requests for file m, is adopted to help identifying a historical record and making replacement decisions. Denote $Q_m(t)$ as the number of requests in queue m at time t. The update of queue m is described as

$$Q_m(t+1) = \max\{Q_m(t) + r_m(t) - s_m(t) - s_m^{\text{out}}(t), 0\}, \quad (2)$$

where $r_m(t) \geq 0$ is the requests for file m arrived at time t, $s_m(t) \geq 0$ is the number of users served by the D2D network for file m at time t, and $s_m^{\text{out}}(t) \geq 0$ is the outage of requests of file m at time t. Note that here outage is defined as a user to drop the request before being served by the network. It can be observed that $Q_m(t)$ $r_m(t)$, $s_m(t)$, and $s_m^{\text{out}}(t)$ are random processes, where $r_m(t)$ is related to the popularity distribution and the number of users and their modes; $s_m(t)$ is related to the caching distribution of devices and the number of users in the network; $s_m^{\text{out}}(t)$ depends on how patient a user would be for waiting the service. Obviously, for files not stored by any of the users, an outage always occurs unless the users can wait for infinitely long time, which is not possible in practice. With these interpretations, we identify the following conditions that enable a large benefit of the replacement scheme, as well as mathematical tractability:[5]

I. Time scale decomposition:

1) Popularity distribution varies slowly with respect to the replacement, i.e., $E[T_{\text{pop}}] > E[T_{\text{rep}}]$, where $E[T_{\text{pop}}]$ is the average time period that the popularity distribution stays invariant and $E[T_{\text{rep}}]$ is the average time between two replacement actions.
2) User mobility is slow with respect to the replacement, i.e., $E[T_{\text{cell}}] > E[T_{\text{rep}}]$, where $E[T_{\text{cell}}]$ is the average time period that a user stays in the cell served by the same BS.
3) Suppose a user has active and inactive modes. We assume the user mode changes at a frequency similar to or slower than the replacement, i.e., $E[T_{\text{mode}}] \gtrapprox E[T_{\text{rep}}]$, where $E[T_{\text{mode}}]$ is the average minimal time period between two different modes of users. Here $E[T_{\text{mode}}]$ indicates how long a user stays active. Thus, this assumption is to guarantee a user request to stay in the queue for a reasonably long period.

II. Monotonicity:

1) Although $s_m(t)$ can be a function of different parameters, such as queue size $Q_m(t)$, instant caching distribution $b_m(t)$, user locations, user modes, and etc., we assume that $s_m(t)$ is a monotonically increasing function with respect to the increase of $b_m(t)$. This is a common behavior of a caching network that the more the network caches a file, the higher the service rate for the network regarding that file could be.
2) We assume the expected outage rate, $E[s_m^{\text{out}}(t)]$, is a monotonically increasing function with respect to the queue size $Q_m(t)$. This is also commonly observed since a larger queue size indicates more users that can cause congestion in the queue.

The overall procedure in a time slot t is the following: the users first send requests to the BS. The BS then observes $r_m(t)$ (of course the $Q_m(t), \forall m$, are already known). Then the BS makes the decision on the replacement action and conducts the replacement procedure according to the decision. After the action, the network then serves the users through D2D communications and/or self-caching by the pre-determined scheduling mechanism and decides $s_m(t)$. Finally, the transition of user modes is conducted leading to $s_m^{\text{out}}(t)$, and then we finish time slot t and the network transitions to time slot $t+1$. We emphasize that our model is very general so that there is no need to specify a file request and content delivery model for the D2D links, i.e., any file request and content delivery model that can be described by (2) is feasible for using our design (of course, we need a specific file request and content delivery model for obtaining the numerical results as we will demonstrate in Sec. V).

III. Dynamic Caching Content Replacement

In this section, we describe the procedure of caching content replacement and introduce the replacement problem.

We consider the BS can broadcast a single file at a time.[6] Suppose we want to increase $b_m(t)$ by $d(t)$, where $0 \leq d(t) \leq 1 - b_m(t)$ is the replacement step-size, i.e., we want to replace other files by file m with a targeted number/ratio $d(t)$. To do this, in addition to broadcasting file m, the BS needs to decide which files should be replaced. Here our policy is to first replace those files with lower pressure on their queues. To be specific, we construct a file replacement order by assigning the file with the smaller queue size a smaller index and selecting files with lowest index to be replaced first, and then following the order of the indices to drop files until the desired number/ratio of files, i.e., $d(t)$, is achieved. Note that the user that should drop the file is selected randomly, e.g., when deciding to drop file 3 and cache file 1, the users that should perform this operation are selected randomly from the set of users having cached file 3. To provide a concrete example, suppose we have 3 files with $b_1(t) = 0.3$, $b_2(t) = 0.3$, $b_3(t) = 0.4$ and $Q_1(t) = 8$, $Q_2(t) = 4$, $Q_3(t) = 2$, and want to increase file 1 by $d(t) = 0.05$. The BS broadcasts file 1 and selects file 3 to be replaced by the ratio of 0.05, resulting in $b_1(t) = 0.35$, $b_2(t) = 0.3$, $b_3(t) = 0.35$ after the replacement. Thus, a randomly chosen $1/8$ of all users that have file 3 cached replace it with file 1. Consider another example that we want to increase file 1 by $d(t) = 0.5$. Then we again broadcast file 1 and replace files, leading to $b_1(t) = 0.8$, $b_2(t) = 0.2$,

[5]Violating these conditions leads to a *gradual* performance loss - e.g., as the mobility of the users becomes faster, the performance gain gets gradually reduced.

[6]The extension to broadcasting multiple files at a time is subject to future work. Having said that, broadcasting multiple files within a consecutive short period is not too different from broadcasting only a single file at a time.

$b_3(t) = 0$ after the replacement. Thus, all users that have cached file 3 replace it by file 1, and a randomly chosen 1/3 of users that have cached file 2 replace it with file 1. The intuition of this replacement procedure is that the file with a lower pressure, i.e., smallest queue size, likely are cached on devices more frequently than is necessary to serve the user requests. It is obvious the considered replacement procedure can be further optimized by considering more flexible strategies so that, instead of dropping all files with the smallest index first and then the second (see the second example), we can flexibly switch between dropping different files. However, this flexibility complicates the problem. Thus, in this work, we focus on deciding when and which file should be broadcast and what step-size to take, and leave the flexible assignment for future work. We note that by observations and empirical results, this sub-optimal replacement procedure is sufficiently effective if the file to be broadcast and the step-size are carefully decided.

In this work, our goal is to decide when and which file to be broadcast and what step-size to take using the provided replacement procedure. The goal of the decisions is to maximize the number of users served by the D2D network subject to some cost constraint due to replacement and to guarantee the queue stability. This is mathematically formulated as

$$\max_{A(t)\in\mathcal{A},\forall t} \quad \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\left[R(t,\mathcal{P}(t),A(t))\right] \tag{3a}$$

$$\text{s.t.} \quad \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} E\left[c_{\text{inst}}^{A(t)}(t)\right] \le C \tag{3b}$$

$$\lim_{T\to\infty}\frac{1}{T} E\left[Q_m^{A(t)}(T)\right] = 0, \forall m, \tag{3c}$$

where $A(t)$ is the action we take at time t and $A(t), t = 1, 2, ...$ constitute an action policy; $\mathcal{A}$ is the action space including broadcasting files $1, 2, ..., M$ and being silent without broadcasting; $\mathcal{P}(t)$ is the system parameter set at time t; $R(t, \mathcal{P}(t), A(t))$ is the reward function expressed as the number of users in the network served at time t;[7] $c_{\text{inst}}^{A(t)}(t)$ is the cost of action $A(t)$; C is the cost constraint; $Q_m^{A(t)}(t)$ is the size of queue m under a sequence of decisions $A(0), ..., A(t-1)$.[8] In the problem formulation, (3a) indicates our goal is to maximize the number of users to be served by the network; (3b) indicates there is a cost constraint we need to follow, and therefore we cannot always broadcast; (3c) indicates we need to mean-rate stabilize every queue [18] so that all requests can be possibly served as long as they stay in the system.[9]

IV. Myopic Drift Minimization Replacement

Solving (3) is a sequential decision-making problem, which is very challenging under general conditions and with large dimension. In this section, we will propose an approach that effectively maximizes the reward while subject to the required constraints via exploiting the concepts of Lyapunov optimization [18].

The idea of the Lyapunov optimization is to minimize the drift of queues [18]. To define the drift, we introduce a virtual cost queue:

$$Z(t+1) = \max\left(Z(t) + c_{\text{inst}}^{A(t)}(t) - C, 0\right), \tag{4}$$

where $0 \le Z(0) < \infty$ is the initial condition. Then by (2) and (4), we can have the inequality in (5) on top of the next page, where $2B \ge \sum_{m=1}^{M}\left(r_m(t) - s_m(t) - s_m^{\text{out}}(t)\right)^2 + \left(c_{\text{inst}}^{A(t)}(t) - C\right)^2 \ge 0$ is a constant. We define $L(t) = \frac{1}{2}\sum_{m=1}^{M}\left(Q_m(t)\right)^2 + \left(Z(t)\right)^2$ and assume that the arrival rate is bounded, i.e., $r_m(t) < \infty, \forall m$. The drifts of the queues $L(t+1) - L(t) = \Delta(t, \mathcal{P}(t), A(t))$ are then bounded as

$$\begin{aligned}
&\Delta(t,\mathcal{P}(t),A(t))\\
&\overset{(a)}{\le} -\sum_{m=1}^{M} Q_m(t)s_m(t) + Z(t)\left(c_{\text{inst}}^{A(t)}(t) - C\right) + X + B\\
&\overset{(b)}{\le} -Q_m(t)s_m(t) + Z(t)\left(c_{\text{inst}}^{A(t)}(t) - C\right) + X + B, \forall m,
\end{aligned} \tag{6}$$

where $X \ge \sum_{m=1}^{M} Q_m(t)r_m(t) \ge 0$ is a constant bound when $Q_m(t), \forall m$, are upper bounded, and we assume this bound exists. Note that (a) is because $Q_m(t)s_m^{\text{out}}(t) \ge 0, \forall m$, and (b) is because $Q_m(t)s_m(t) \ge 0, \forall m$.

To minimize the drift, we should minimize the upper bounds in the first inequality in (6). However, when the D2D scheduler is complicated, $s_m(t)$ might not have an analytical expression that is easy to compute or estimate under different actions. We thus use the second inequality in (6) and develop a simplification that is based on the following observation: if we choose to broadcast file m at time t, we can immediately know $s_m(t) = Q_m(t) + r_m(t)$ since all requests for file m at time t can be satisfied by the broadcast. Besides, assuming no cost when silence, we also know that a sufficient condition to chose to be silent is:

$$-Q_m(t)s_m(t) + Z(t)c_{\text{inst}}^{A(t)}(t) > 0, \forall m. \tag{7}$$

We denote $A_m \in \mathcal{A}^{\text{br}}$ as the action to broadcast file m. By previous observations, we thus solve the following optimization problem for making the decision:

$$A(t) = \arg\min_{A\in\mathcal{A}} g_A(t), \tag{8}$$

where $g_A(t) = -\sum_{m=1}^{M}\mathbb{1}_{\{A=A_m\}}\cdot Q_m(t)s_m^A(t) + Z(t)c_{\text{inst}}^A(t)$ and $\mathbb{1}_{\{A=A_m\}}$ is an indicator function whose value is 1 only if the system broadcasts file m. We note that when $A = A^{\text{slt}}$, $g_{A^{\text{slt}}}(t) = 0$ and when $A = A_m$, $g_{A_m}(t) = -Q_m(t)\left(Q_m(t) + r_m(t)\right) + Z(t)c_{\text{inst}}^{A_m}(t)$. Solving (8) is simple. Since $g_A(t) = 0$ when $A = A^{\text{slt}}$, we only need to evaluate $g_A(t), \forall A = A_m \in \mathcal{A}^{\text{br}}$ and compare the resulting values with zero. If there is no value lower than zero, the system remains silent; otherwise, the system broadcasts the file providing the

[7]The formulation can also be extend to use other units, such as bit.

[8]We use $Q_m(t)$ for the general purpose, while use $Q_m^{A(t)}(t)$ to stress out that the results is under the policy $\{A(0), A(1), ..., A(t-1)\}$

[9]We actually can show a stronger result that the queue size is upper bounded for our design (see Theorem 1).

$$\sum_{m=1}^{M}\left[Q_m(t+1)\right]^2+\left[Z(t+1)\right]^2\leq\sum_{m=1}^{M}\left[Q_m(t)+r_m(t)-s_m(t)-s_m^{\text{out}}(t)\right]^2+\left[Z(t)+c_{\text{inst}}^{A(t)}(t)-C\right]^2$$
$$\leq\sum_{m=1}^{M}\left[Q_m(t)\right]^2+\left[Z(t)\right]^2+2\left[\sum_{m=1}^{M}Q_m(t)\left(r_m(t)-s_m(t)-s_m^{\text{out}}(t)\right)+Z(t)\left(c_{\text{inst}}^{A(t)}(t)-C\right)\right]+2B \tag{5}$$

minimal objective value. The intuition of (8) and the resulting approach is basically to broadcast the file with highest pressure on the queue when there exists a real (file) queue whose pressure is higher than the pressure of the virtual (cost) queue.

The complete replacement approach is to solve (8) at every time slot and decide the actions. The overall algorithm is summarized in Alg. 1. Since (8) can be easily solved, the complexity of the approach is extremely low. Also since the proposed approach exploits the history record (queue sizes) and the current system state without using any potential future information, this approach is named myopic drift minimization (MyDM) replacement. Finally, since MyDM uses a constant step-size for conducting the replacement procedure, the step-size should be carefully selected. The proposed MyDM approach can guarantee the time-average cost constraint and stabilize the queues asymptotically, according to the following theorems:

Theorem 1: Suppose $M<\infty$ is upper bounded, $Z(0)$ is finite, and $r_m\leq B$ is upper bounded. Consider $Q_m(0),\forall m$, is finite and bounded; $C>0$ and $c_{\text{inst}}^{A},\forall A\in\mathcal{A}$ is finite and bounded. Under MyDM, $Q_m^{A(t)}(t),\forall m$, are mean-rate stable, finite, and upper bounded.

Theorem 2: Consider the same conditions as in Theorem 1, MyDM can satisfy the time-average cost constraint, i.e., the proposed algorithm satisfies almost surely

$$\lim_{t\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E\left[c_{\text{inst}}^{A(t)}(t)\right]\leq C. \tag{9}$$

Proof. Refer the proofs to the Journal version [20]. □

Algorithm 1 Myopic Drift Minimization Replacement

Init: Start at $t=0$, $Q_m(0)\geq 0,\forall m$, $Z(0)\geq 0$. Set step-size $d(t)=d$.
for $t=0,1,...$ **do**
  Evaluate $g_{A_m}(t)=-Q_m(t)\left(Q_m(t)+r_m(t)\right)+Z(t)c_{\text{inst}}^{A_m}(t),\forall m$
  if $\min_{m=1,..,M}g_{A_m}(t)<0$ **then**
    Broadcast the file m, where $m=\arg\min_{m=1,..,M}g_{A_m}(t)$
    Conduct the replacement procedure in Sec. III
  else
    Keep silence, i.e., $A(t)=A^{\text{slt}}$
  end if
  Update the real queues $Q_m(t),\forall m$, and the virtual queue $Z(t)$
end for

V. Performance Evaluations and Discussions

Here we evaluate the proposed design using computer simulations. We will first evaluate the performance and discuss the insights using an idealized model to point out insights, and then provide evaluations using a more practical network configuration. In all the simulations, unless otherwise indicated, we consider $M=100$, $C=1$, and step-size $d=0.05$, and evaluate the average number of requests to be satisfied. The number of time slots T used for each simulations might be different, but they are large enough for the proposed design to converge to satisfy the average cost constraint.

A. Evaluations with Ideal Network Configurations

We first evaluate the proposed MyDM replacement in the idealized environment in which $r_m(t)=f_m(t)$, $s_m(t)=b_m(t)$, $s_m^{\text{out}}(t)=0,\forall m$, and $c_{\text{inst}}^{A(t)}=10,\forall A(t)\in\mathcal{A}$. We note that $f_m(t)$ is the request probability of file m at time t, in which $\sum_{m=1}^{M}f_m(t)=1$ and $\{f_m(t)\}$ varies from time to time according to some model. The intuition of this environment is that the request and service rates are linearly proportional to the request and caching distributions, respectively. Also, we can clearly see that the requests generated in each time slot can be perfectly served if the caching distribution is perfectly aligned with the request distribution. We first consider the request distribution to vary between different Zipf distributions with Zipf parameters $\gamma=0.2+0.5(k-1)$, where $k\in\{1,2...,25\}$, according to a Markov process whose transition probability matrix is $P_{k,k}=0.95,P_{k,k+1}=0.025,P_{k,k-1}=0.025$, where $2\leq k\leq 24$; $P_{1,1}=0.95,P_{1,2}=0.05,P_{25,25}=0.95,P_{25,24}=0.05$; $P_{k,l}=0$, otherwise. We then compare the proposed design with the non-replacement designs. To obtain more insights, we also compare to the non-replacement design with periodic broadcasting and the proposed design excluding the benefit of broadcasting, i.e., the queues for file m are not flushed out when file m is broadcast.

The results are shown in table I, in which "Self" indicates the selfish approach, i.e., $b_1(t)=1$ and $b_m(t)=0,\forall m\geq 2$; "Zipf-0.8" indicates the Zipf distribution based time-invariant caching policy [6] whose parameter is 0.8 (the value 0.8 is an optimized selection);[10] and "brod" indicates the involvement of broadcasting. Although not shown here, the MyDM replacement can offer almost identical performance with respect to different initial caching policies, showing the ability to correct inappropriate caching policies. Besides, the cost and queuing

[10]The optimization is conducted by first quantizing the Zipf parameter and then selecting the one providing the best reward in Monte-Carlo simulations.

TABLE I: Evaluations with Zipf Request Distribution

Scheme	Self	Zipf-0.8	MyDM no Brod
Service Rate	0.101	0.892	0.990
Scheme	Self + brod	Zipf-0.8 + brod	MyDM
Service Rate	0.989	0.994	0.994

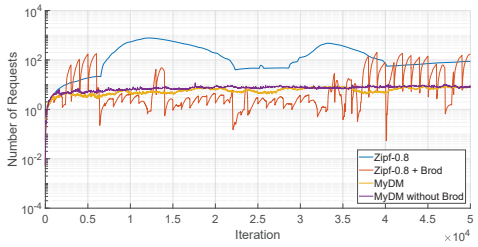

Fig. 1: Maximum queue size comparisons.

constraints are satisfied in all cases. We also observe that the selfish design provides poor performance while the Zipf-based scheme can provide good performance. However, such good performance needs the optimized Zipf parameter, based on (i) knowledge of the request distribution in the future; and (ii) the effectiveness of optimization. In contrast, the proposed design can adjust the caching results dynamically without knowing the future. In addition, we can observe that all schemes with broadcasting can offer performance close to perfect request-caching alignment because the content in the queues are never dropped. However, we will later see that different schemes can have different behaviors regarding the maximum queue sizes. Finally, the proposed design can be effective even if we exclude the benefit of broadcasting, indicating that it can appropriately adjust the caching realizations. In Fig. 1, we compare between the maximum queue sizes of different designs. Note that a large queue size might be inferior due to large delay and/or potential outage after the expirations of requests. From the figure, we can see that the proposed design can maintain a small maximum queue size while the other schemes cannot even when their average performance is close to the proposed design.

Now we provide the evaluations adopting a time-varying request distribution varing between 100 different request distributions, in which each request distribution is constructed by averaging 100 different preference probabilities, generated by using the practical generator in [19], of different users.[11] The model for the transition of the request distribution at each time slot is: the probability to be invariant is 0.95; otherwise transit to other distributions with equal probability. The results are shown in Table II. It can be observed that the proposed design is again very effective. Besides, we can see that the performance of the Zipf-based scheme is less effective (even with the optimized parameter 0.6) owing to less effectiveness when considering the practically generated distributions.

[11]The motivation of using this model is to evaluate the proposed design using distributions based on real data. Details of the generating procedure are omitted here for space reasons.

TABLE II: Evaluations with Practical Request Distribution

Scheme	Self	Zipf-0.6	MyDM no Brod
Service Rate	0.026	0.686	0.998
Scheme	Self + brod	Zipf-0.6 + brod	MyDM
Service Rate	0.985	0.995	0.999

B. Evaluations with Practical Network Configurations

We now evaluate our design considering a more practical network configuration. We consider the prioritized-push network in [12] without the use of BS links and consider 1000 users to move within a square cell whose side length is $D = 500$m. The prioritized-push network adopts clustering, where only devices in the same cluster can communicate with each other, and employ a spatial reuse scheme to reuse resources and avoid interference. Thus, the size of a cluster, denoted as cooperation distance, can greatly affect the throughput and outage performance (for more details see [12].). A BS covering the whole cell serves as the central controller and conducts replacement. A user can be active or inactive. When an active user is served by the network, the user immediately transits to inactive after the service. Each user could change their modes at the end of each time slot. When a user changes from active to inactive, the request of the user drops off from the queuing system, causing outage; when a user changes from inactive to active, a request of the user is generated according to the request distribution at that time, and then sent to the BS at the beginning of the next time slot. After the transitions of user modes, the transition of the request distribution is conducted and the locations of users are changed. We consider each user to randomly move to any location in the cell with equal probability. In the following simulations, the probability of changing mode is 0.05. Besides, we consider $c_{\text{inst}}^{A(t)} = 10, \forall A(t)$. We note that although locations of users might change at each time slot, this network is still considered slowly-varying because users remain in the same cell.[12]

In our first evaluation, we consider the request probability to vary according to a special pattern: all users request the same popular file, and such preference changes every 100 time-slots in a round-robin manner, e.g., at the first 100 slots, every user requests file 1, and starting from slot 101, users request file 2. This scenario is to represent the sudden change of the popularity distribution due to the happening of some big events. With this pattern, the optimal hindsight replacement is simply to replace all cached files by the one that would start to be requested in the next time slot. We compare the proposed design to other schemes in Fig. 2. From the figure, we can observe that the proposed design outperforms the Zipf-based scheme and is near-optimal. We then provide the evaluation in the network adopting the same setup as in Table II except that now $d = 0.01$, and show the results in Fig. 3. From the figure,

[12]Note that there are still some impractical considerations in the adopted network. Evaluations using even more practical network configurations and request distributions will be provided in the journal version. That being said, evaluations in this paper still provide strong support for the benefits of the proposed replacement design.

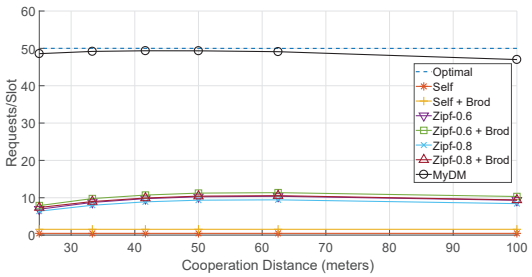

Fig. 2: Performance comparisons between different schemes.

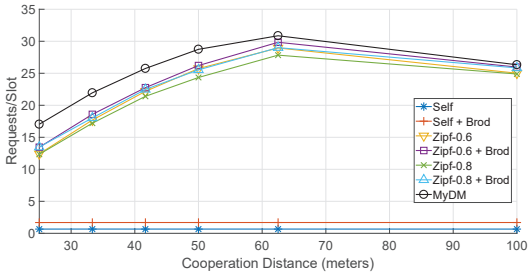

Fig. 3: Performance comparisons between different schemes.

we can see that the proposed design can again outperform other schemes, especially when the non-replacement design adopts a bad caching policy. This indicates that the proposed replacement design can correct a bad initial caching policy and significantly improve the performance. We stress that the performance of the proposed design is basically independent of the initial caching policy, and cost and queuing constraints are satisfied in all simulations. We also note that although the Zipf-based scheme performs well in this case, it requires perfect knowledge of the request distributions in the future and is subject to the necessity of an effective optimization. As a counter-example, we can see from Fig. 2 that the Zipf-based scheme works poorly while our design still performs well.

VI. Conclusions

To deal with issues caused by time-varying popularity distribution and inappropriate static caching policy, a caching content replacement problem is formulated and investigated. We propose a network structure able to conduct replacement along with a replacement procedure. We propose the replacement design and offer theorems to show that the proposed design can generate feasible solutions to the problem. By using simulations, we validate that the system can be improved by the proposed design when compared with static policies, and the improvement is especially significant when the static policy performs poorly under varying popularity distribution. Besides, the robust performance of the proposed design with respect to different initial caching policies indicate that our design can correct inappropriate caching policies effectively.

The proposed replacement approach here does not exploit the potential future information, i.e., knowledge about future changes in the popularity distribution and the corresponding payoff. The extension to including such knowledge will be discussed in our Journal version [20].

Acknowledgment

The authors would like to thank Professor Rahul Jain for helpful discussions. This work was supported in part by the National Science Foundation (NSF) under CNS-1816699 and CCF-1423140.

References

[1] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commmun. Mag.*, vol. 51, no. 4, pp. 142-149, Apr. 2013.

[2] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, and A. Mauthe, "A survey of mobility in information-centric networks," *ACM Commun.,* vol. 56, no. 12, pp. 90-98, Dec. 2013.

[3] K. Shanmugam, N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402-8413, Dec. 2013.

[4] C. Yang, Y. Yao, Z. Chen, and B. Xia, "Analysis on cache-enabled wireless heterogeneous networks," *IEEE Trans. Wireless Commun.,* vol. 15, no. 1, pp. 131-145, Jan. 2016.

[5] Y. Cao, M. Tao, F. Xu, and K. Liu, "Fundamental storage-latency tradeoff in cache-aided MIMO interference networks," *IEEE Trans. Wireless Commun.,* vol. 16, no. 8, pp. 5061-5076, Aug. 2017.

[6] N. Golrezaei, P. Mansourifard, A. F. Molisch, and A. G. Dimakis, "Base-Station assisted device-to-device communications for high-throughput wireless video networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3665-3676, Jul. 2014.

[7] M. Ji, G. Caire, and A. F. Molisch "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. Sel. Area Commun.*, vol. 34, no. 1, pp. 176-189, Jan. 2016.

[8] B. Chen, C. Yang, G. Wang, "High-Throughput opportunistic cooperative device-to-device communications with caching," *IEEE Trans. Veh. Technol.,* vol. 66, no. 8, pp. 7527-7539, Aug. 2017.

[9] R. Wang, J. Zhang, S. H. Song, and K. B. Letaief, "Mobility-Aware Caching in D2D Networks," *IEEE Trans. Wireless Commun.,* vol. 16, no. 8, pp. 5001-5015, Aug. 2017.

[10] M. Naslcheraghi, M. Afshang, H. S. Dhillon, "Modeling and performance analysis of full-duplex communications in cache-enabled D2D networks," *in Proc. IEEE ICC,* May 2018.

[11] M.-C. Lee, M. Ji, A. F. Molisch, and N. Sastry, "Performance of caching-based D2D video distribution with measured popularity distributions," *arXiv:1806.05380,* Jun. 2018.

[12] M.-C. Lee and A. F. Molisch, "Caching policy and cooperation distance design for base station assisted wireless D2D caching networks: Throughput and energy efficiency optimization and trade-off," *IEEE Trans. Wireless Commun.,* vol. 17, no. 11, pp. 7500-7514, Nov. 2018.

[13] P. T. Joy and K. P. Jacob, "A comparative study of cache replacement policies in wireless mobile networks," *Advances in Comput. and Information Technol.,* pp. 609-619, 2012.

[14] M. Veeresha and M. Sugumaran, "Optimal hybrid broadcast scheduling and adaptive cooperative caching for spatial queries in road networks," *J. Ambient Intell. Human Comput.,* vol. 8, no. 4, pp. 607624, Aug. 2017.

[15] M. Gregori, J. Gómez-Vilardebó, J. Matamoros, and D. Gündüz, "Wireless content caching for small cell and D2D networks," *IEEE J. Sel. Area Commun.,* vol. 34, no. 5, pp. 1222-1234, May 2016.

[16] Z. Chen, H. Mohammed, and W. Chen, "Proactive caching for energy-efficiency in wireless networks: A Markov decision process approach," *IEEE Trans. ICC,* May. 2018.

[17] D. Wu, L. Zhou, Y. Cai, and Y. Qian, "Collaborative caching and matching for D2D content sharing," *IEEE Wireless Commun. Mag.,* vol. 25, no. 3, pp. 43-49, Jul. 2018

[18] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems.* Morgan & Claypool 2010.

[19] M.-C. Lee, A. F. Molisch, N. Sastry, and A. Raman, "Individual preference probability modeling for video content in wireless caching networks," *IEEE GLOBECOM*, Dec. 2017.

[20] M.-C. Lee, H. Feng, and A. F. Molisch, "Dynamic caching content replacement in base station assisted wireless D2D caching networks," *IEEE Trans. Commun.*, submitted to.