Received January 17, 2020, accepted February 2, 2020, date of publication February 13, 2020, date of current version February 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973953

Dynamic Caching Content Replacement in Base Station Assisted Wireless D2D Caching Networks

MING-CHUN LEE[1], (Student Member, IEEE), HAO FENG[2], (Member, IEEE), AND ANDREAS F. MOLISCH[1], (Fellow, IEEE)

[1]Department of Electrical and Computer Engineering, University of Southern California (USC), Los Angeles, CA 90089, USA
[2]Intel Labs, Intel Corporation, Hillsboro, OR 97124, USA

Corresponding author: Ming-Chun Lee (mingchul@usc.edu)

This work was supported in part by the National Science Foundation (NSF) under Grant CNS-1816699 and Grant CCF-1423140.

ABSTRACT The concentrated popularity distribution of video files and the caching of popular files on users and their subsequent distribution via device-to-device (D2D) communications have dramatically increased the throughput of wireless video networks. However, since popularity distribution is not time-invariant, and the files available in the neighborhood can change when other users move into and out of the neighborhood, there is a need for replacement of cache content. In this work, we propose a practical and feasible replacement architecture for base station (BS) assisted wireless D2D caching networks by exploiting the broadcasting of the BS. Based on the proposed architecture, we formulate a caching content replacement problem, with the goal of maximizing the time-average service rate under the cost constraint and queue stability. We combine the reward-to-go concept and the drift-plus-penalty methodology to develop a solution framework for the problem at hand. To realize the solution framework, two algorithms are proposed. The first algorithm is simple, but exploits only the historical record. The second algorithm can exploit both the historical record and future information, but is complex. Our simulation results indicate that when dynamics exist, the systems exploiting the proposed designs can outperform the systems using a static policy.

INDEX TERMS Caching content replacement, device-to-device communication, wireless caching network.

I. INTRODUCTION

The demand of wireless traffic has increased dramatically in the past several years, and this demand is expected to continue to grow in the future [2]. Among numerous wireless applications, the delivery of video content accounts for majority of the data traffic; how to support this application is one of the challenges for 4G and 5G systems [3], [4]. Conventional throughput-enhancing approaches (e.g., massive antenna systems, network densification, and millimeter wave systems) all rely on obtaining more physical resources and/or increasing infrastructure investment, which are generally expensive [5]. In contrast, memory has become the cheapest hardware resource owing to the rapid development of the semiconductor industry. This then motivates researchers to exploit content (in particular video) caching in both wireline and wireless networks [3]–[6]. The idea is to trade memory for bandwidth by caching files during 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 to satisfy video traffic demand [3]–[5].

Recently, on-device video caching, combined with high performance device-to-device (D2D) communications [7] for video distribution, has shown (both in theory and in practice) its capability to improve performance significantly without needing to install additional infrastructure and without having to use complicated coding [8]–[10].[1] As a result, numerous papers have investigated such D2D-based caching using different strategies and from different aspects, including successful transmission (hit) rate [17], [18], throughput [19], [20], energy efficiency [20], [21], and delay [22], [23]. For example, [20]–[22] investigated designs that consider the interplay between caching policy and

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Shuja.

[1]Other approaches can be used to exploit caching at the wireless edge, e.g., femtocaching [3], [11], [12]; BS-caching [13], [14]; and coded multicasting [15], [16]. Howeever, they are beyond the scope of this paper.

cooperation distance as well as the trade-offs between different goals. Stochastic geometry techniques were exploited to analyze cache-aided D2D networks in [24], whereas [25] leveraged mobility to enhance networks. By using the graphical approach, [26] proposed a joint content caching and link scheduling design. In [27], caching and dynamic streaming strategies were investigated while considering the trade-off between video quality and diversity in the design.

The main challenge in caching networks sits on the network's decision on which file is cached by whom. This is, by and large, considered as the caching policy design problem. Although many researchers have already investigated the different aspects of this problem using different approaches, their main emphasis generally lies on the direction of static policies 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. Conversely, it can be beneficial to consider dynamics of the network and to proactively conduct dynamic caching content replacement/refreshing, in which content of caches is proactively replaced by some controllers according to the dynamics of the network. There are several motivations for it: (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); (ii) the caching realizations of the network can be inappropriate (e.g., users do not cache files according to the designated policy); and (iii) user mobility can change the locally available user density and cached files. Such network dynamics can degrade the performance of a network that uses a statically designed caching policy, whereas adaptations (cache replacement) can automatically compensate. Accordingly, adopting a dynamic design can be beneficial; however, to the best of our knowledge, proactive dynamic content caching and replacement in wireless D2D caching networks have still yet to be investigated fully (as will be further discussed below). Thus, this paper aims to improve this situation.

A. RELATED LITERATURE REVIEW

Several papers have investigated the dynamic cache replacement in femtocaching and BS-caching cases [28]–[34]. In [28], the authors proposed adopting a distributed caching replacement approach via Q-learning, albeit their focus was on caching at the BS and on fixed network topology. In [29], the caches of BSs were refreshed periodically to stabilize the queues of two request types and to satisfy the quality of service requirements. In [30], the authors aimed to offload the traffic to infostations, and thus used a multi-armed bandit optimization to refresh the caches of BSs. Meanwhile, [31] proposed an algorithm exploiting the multi-armed bandit optimization to learn the popularity of the cached content and update the cache to increase the cache-hit rate. In [32], a reinforcement learning framework was proposed while considering popularity dynamics into the analysis in order to refresh the caches in BSs and to minimize delivery cost. In [33], the loss due to outdated caching policy was analyzed for a small cell BS and an updating algorithm to minimize the offloading loss was proposed. Based on real-data observations, [34] established a workload model and then developed simple caching content replacement policies for edge-caching networks. However, these caching replacement policies for femtocaching do not carry over to D2D caching networks due to the following: (i) the use of more constrained wireless channels demands a specific architecture for conducting replacement; (ii) the distributed file-caching structure and intertwined D2D cooperations and communications between users result in a more complicated and constrained conditions for making replacement decisions; and (iii) the locally available cached files can change with time due to user mobility, e.g., users carrying critical files could vanish right after the replacement actions.

Cache replacement in users has its history in the Computer Science community [35], [36], which generally consider individual replacement and/or networks with special properties without considering D2D cooperation. Although [37] implicitly used content caching replacement, the study mainly focused on joint content delivery and caching design at a given time slot at a given user demand. This is obviously different from our goal. In [38], user cache refreshment was investigated using a Markov decision process (MDP); however, the study focused on efficient buffering for a single user and ignored the important multiuser situation and D2D network communications. In [39], the problem of how users can ‘‘reactively’’ update their caching content was investigated. This is different from our aim of ‘‘proactively’’ updating the caching content.

B. CONTRIBUTIONS OF THIS PAPER

In this work, we consider BS-assisted wireless D2D caching networks and focus on dealing with different dynamics, including user mobility and the change of popularity distribution. We first propose a network architecture for content replacement. Since dynamics exist when conducting content replacement, we then devise approaches to help decide which files should be cached and what files should be removed from users' caches. To provide a general design for the network, we describe the network using several random processes, i.e., service process that describes the services for video file requests, arrival process that describes the arrivals of requests, and outage process that describes the dropping of requests. Thus, any network whose behavior can be described by those processes can use our design. To conduct replacement, we propose using the broadcasting nature of the BS. To observe the network behavior and make decisions, we use a queueing system to individually queue up requests of different files. Hence the BS can make decisions by observing the network state and queueing record. Since the replacement action (via broadcasting) generates cost, we thus aim to maximize the time-average service rate, defined as the average number of requests served per time slot, subject to a time-average cost constraint and queue stability.

The replacement problem includes three parts: (i) deciding when to conduct a replacement; (ii) deciding which files to newly cache on users; and (iii) deciding what files should be replaced, i.e., deleted from the caches. The joint design of these three problems is extremely challenging. Thus, we propose a heuristic but effective procedure for the final part of the problem. Most of the work in the paper concentrates on the first two parts, i.e., deciding when and which files to push into the user caches. For this, we develop a sequential decision-making optimization problem, and propose a solution framework by combining the "reward-to-go" concept and the drift-plus-penalty methodology from Lyapunov optimization [40]. We also provide analytical results to show insights and benefits of this framework. Directly solving the optimization problem in the framework might not be feasible; thus, we propose two algorithms for practical implementation, as the algorithms can satisfy the time-average constraint and stabilize the queues. The first algorithm makes myopic decisions to minimize the upper bound of the drift-plus-penalty term. This approach is fairly simple; however, it uses historical record and present system states without considering future information. On the other hand, we can leverage on potential future information in the second algorithm, as it employs Monte-Carlo sampling [41], [42] to incorporate future information into the decision-making process. To enhance the second approach, two complexity-reduction approaches for Monte-Carlo sampling are proposed. We use simulations to demonstrate the efficacy of the proposed replacement designs and to gain insights into these approaches. The results show that when dynamics exist and our approaches are used, the network is significantly improved as compared to that when the static approaches are used. Our main contributions are summarized as following:

- We discuss the replacement problem in wireless D2D caching networks and propose a network architecture for the replacement procedure. To the best of our knowledge, this is the first work to focus on dynamic replacement in wireless D2D caching networks.
- We formulate the replacement problem in the form of a sequential decision-making problem with time-average cost constraint and queue stability. We propose a solution framework that incorporates the reward-to-go concept into the drift-plus-penalty methodology and then discuss the insights and benefits gained from adopting this framework.
- To put the proposed framework in practice, we develop and propose using two replacement algorithms that can satisfy the time-average constraints and stabilize the queueing system. The first algorithm is fairly simple to implement, but uses only the current system state and historical records for the content replacement. The second algorithm, on the other hand, can effectively leverage on both historical record and future predictions to make decisions.
- Our simulations, which adopt the practical network configurations for cache replacement, validate the effectiveness of our proposed designs. The results show that the dynamic cache replacement can significantly improve network performance. Likewise, the simulation results provide insights into the dynamic replacement process performed in this paper.

II. SYSTEM MODEL

In this work, we consider a BS-assisted wireless D2D caching network, in which users in the network can cache files and communicate with one another. We consider a centrally controlled scheduling for D2D networks; the BS serves as the central controller that collects requests and caching information from users, schedules D2D communications, and decides on the replacement actions. We also assume that the BS can broadcast files to users, thereby enabling the cache content replacement for users. To focus on the performance of the on-device caching, we assume that users can be served only through self-caching, D2D caching, and broadcast without using user-specific BS links. Thus, user requests can only be satisfied in three ways: files in their own caches, files accessible via D2D communication, and broadcast files. When a user generates a request, it first checks whether this request can be satisfied by the files in its own cache, i.e., by self-caching. If yes, then the request is satisfied; otherwise, i.e., a request cannot be satisfied by self-caching, this request is sent to the BS for possible services via the D2D communication or via broadcast.

We assume in the file replacement process that the central controller can observe all requests sent to the BS and knows the information on which files are cached by which users. These assumptions consequently lead to some additional signaling cost. Moreover, broadcasting files from the BS to the users also induces cost. Since the amount of signaling bits is typically much smaller than the number of bits in a video delivery, the cost of the signaling overhead could be included as part of the cost of conducting a file replacement (which is mainly dominated by the cost of the file broadcast). As will be shown later in Sec. III, our problem formulation considers this cost by having a time-average cost constraint.

We consider a library consisting of M files and assume that all files have equal sizes for simplicity. We assume users can cache only a single file of the library, i.e., $S = 1$, in most of the paper (Secs. III-VI) for simplicity, and extend to networks where users can cache multiple files, i.e., $S > 1$, in Sec. VII. We consider a homogeneous request probability model which uses a_m to describe the probability of a user to request file m, with $\sum_{m=1}^{M} a_m = 1$.[2] To describe the realization of the files cached in the network at time t, we denote the caching

[2]Nevertheless, it will be evident that our proposed replacement framework and designs can also be applied to networks that consider individual user preference [43], [44], although the information on individual preference is not fully leveraged. The design that fully exploits such information is an important direction for future studies

probability of file m as $b_m(t)$:

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

where $N_m(t)$ is the number of users caching file m in the D2D network at time t known by the BS via signaling. By definition, $0 \leq b_m(t) \leq 1$ then indicates the probability of file m being cached by a user of the network. We consider both active and inactive users. The active users are defined as the users who generate requests, whereas the inactive users are those who do not, albeit both types of users participate in the D2D communications. Note that an inactive user can also choose not to participate in the D2D communications depending on the scenario assumptions. However, such inactive user is then independent of the D2D network, and can thus be ignored without restriction of generality. Moreover, as will become clear in the succeeding discussions, our replacement approach does not use the specific information regarding the number of active and inactive users for making decisions. Instead, we use the queuing dynamics of requests to implicitly convey the information on the number of active users waiting for the services. Hence, we do not need to specify the distributions (or numbers) of active and inactive users in the model.

We adopt a queueing system at the BS with M queues, where queue m stores requests for file m, to help identify the historical record and make replacement decisions. We denote $Q_m(t)$ as the number of requests in queue m at time t. The update of queue m is described as

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

where $r_m(t) \geq 0$ is the number of requests of file m arriving at time t, $s_m(t) \geq 0$ is the number of requests of file m satisfied by the network at time t, and $s_m^{\text{out}}(t) \geq 0$ is the outage of requests of file m at time t. Here, an outage is defined as a user dropping the request before being served by the network. It should be noted that $r_m(t)$ and $s_m(t)$ of (2) do not include the requests and services directly satisfied by and provided through self-caching. This is because those requests that self-caching can satisfy would be directly handled by the corresponding services, and they cancel each other. Such result is in line with our model, which posits that a BS can only observe those requests that self-caching cannot satisfy. On the contrary, the impact of self-caching is implicitly considered in the process, as the requests satisfied by self-caching are resolved without having to add requests to the queuing system. Note that when evaluating the overall network performance in simulations, the requests satisfied by self-caching are still considered in the evaluation. Our results in this paper can be used by any file request and content delivery model described in (2). As a result, we do not assume a specific file request and content delivery model.

Observe 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 users and the number of users in the network; $s_m^{\text{out}}(t)$ depends on the user's willingness to wait for the service. Obviously, for files that are not stored by any of the users, if there is no other sources for accessing them (e.g., file broadcasting due to replacement actions), then an outage occurs no matter how long the user is willing to wait. With these interpretations, we identify conditions, i.e., time scale decomposition and monotonicity below, that can significantly benefit the replacement scheme. Note that these conditions are not assumptions that will be used in our analysis later on. Instead, they describe the conditions that would give large replacement benefits in practice. However, violating these conditions can gradually decrease the performance gain. For example, when user mobility becomes faster, the performance gradually degrades (see Fig. 5). Despite this, violating these conditions does not prevent us from using the analytical results and replacement designs provided in this paper.

Time scale decomposition:

1) Popularity distribution varies slowly with respect to the replacement, i.e., $E\left[T_{\text{pop}}\right] > E\left[T_{\text{rep}}\right]$, where $E\left[T_{\text{pop}}\right]$ is the average time period that the popularity distribution stays invariant and $E\left[T_{\text{rep}}\right]$ is the average time between two replacement actions.
2) User mobility is slow with respect to the replacement, i.e., $E\left[T_{\text{cell}}\right] > E\left[T_{\text{rep}}\right]$, where $E\left[T_{\text{cell}}\right]$ is the average time period that a user stays in the effective service area of the same BS.
3) The user mode switches from active to inactive at a frequency similar to or slower than the frequency of the replacement, i.e., $E\left[T_{\text{mode}}\right] > E\left[T_{\text{rep}}\right]$, where $E\left[T_{\text{mode}}\right]$ is the average minimal time period that each user switches from active to inactive. This condition guarantees that a user request stays in the queue for a reasonably long period.

Monotonicity:

1) When the number of requests is sufficient, the number of services, $E\left[s_m(t)\right]$, is monotonically increasing as a function of $b_m(t)$. However, $s_m(t)$ can also be a function of other parameters, such as queue size $Q_m(t)$, user locations, user modes, etc. Usually, the more that the network caches a file, the higher the service rate for the network would be for that file.
2) The expected number of outages, $E\left[s_m^{\text{out}}(t)\right]$, is a monotonically increasing function of the queue size $Q_m(t)$. This is also commonly observed since a larger queue size indicates longer latency of delivery, and thus higher probability that users would cancel a request.

The overall procedure in time slot t is as follows: the users first check whether their requests can be satisfied by the files in their own caches. If yes, then the requests are satisfied. Otherwise, users send requests to the BS. The BS then collects the requests and observes $r_m(t)$ ($Q_m(t), \forall m$, are already known), and then decides what action to take. If the BS decides to conduct a file replacement, then the replacement procedure is consequently conducted according to the decision. After the action, the network serves the users by a pre-determined content delivery mechanism and decides $s_m(t)$. Finally, the transition of user modes is conducted leading to $s_m^{\text{out}}(t)$. We then finish time slot t, and the network transitions to time slot $t+1$. The following summarizes the assumptions and feasibility of our model:

1) The BS can centrally control the D2D scheduling and conduct replacement action, collect requests that cannot be satisfied by self-caching, and collect information on what files are cached by users.
2) To focus on the effects of on-device caching, users are served only by self-caching, D2D-caching, and broadcast from the BS.
3) Users can be either active or inactive. Since our replacement will use the queuing dynamics of requests to make decisions, we do not need to specify the statistics on numbers of active and inactive users in the network (of course, we need to specify their statistics for obtaining the numerical results in Sec. VIII).
4) Our model is very general such that any file request and any content delivery model that (2) describes can use our design. We thus do not specify a file request and content delivery model here (again, we need a specific file request and content delivery model for obtaining the numerical results in Sec. VIII).
5) Although our design could be feasibly used in general situations, this does not mean it would perform well under extreme scenarios, e.g., high-mobility scenarios. We thus discussed the conditions, i.e., *time scale decomposition* and *monotonicity*, that would give large benefits.

III. DYNAMIC CACHING CONTENT REPLACEMENT

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

We assume that $S = 1$ and that the BS can broadcast a single file at a time.[3] Suppose we want to increase $b_m(t)$ by $d_m(t)$, where $0 < d_m(t) \leq 1 - b_m(t)$ is the replacement step-size, i.e., we want to replace other files by file m with a targeted fraction $d_m(t)$. To do this, the BS broadcasts file m to users and decides which files should be replaced or deleted from the cache. Here, our policy is to first replace those files that have the lowest "pressure", i.e., smallest queue size, on the queue. To be specific, we first construct a file replacement order by assigning a smaller index to the file with the smaller queue size. Thereafter, we select and replace the files that have the lowest index, and then follow the order of the indices to drop files until we achieve the desired ratio of files, i.e., $d_m(t)$. Note that the user that should drop the file is selected randomly. For example, when deciding to drop file 3 and cache file 1 from the broadcasting, the users that should perform this operation are selected randomly from the set of users caching file 3 in the network. 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 $d_1(t) = 0.05$. 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. Consider another example that we want to increase file 1 by $d_1(t) = 0.5$. Then we again broadcast file 1 and replace files, leading to $b_1(t) = 0.8$, $b_2(t) = 0.2$, $b_3(t) = 0$ after the replacement. The intuition of this replacement procedure is that the file with a lower pressure likely is cached on users more frequently than is necessary to serve the user requests. We note that since the number of files cached in the network is integer in practice, we cannot realize arbitrary step-size. Thus in practice, we use N_{rep} to decide how many users should conduct the replacement, where $N_{\text{rep}} = \text{round}(U \cdot d_m(t))$ is the integer that can provide the closest approximation to the desired step-size and U is the number of users in the network. It is obvious that the considered replacement procedure can be further optimized by considering more flexible strategies. For example, instead of dropping all the 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. Since we focus on deciding when and which file should be broadcast and what step-size to take, investigating this flexible assignment is left for future work. Note this suboptimal replacement procedure is effective enough if we choose carefully both the file to be broadcast and the step-size.

For most of this work, we focus on deciding when and which files should be broadcast and newly cached by users and what step-size to take in the replacement procedure. The goal of the decisions is to maximize the time-average number of requests satisfied by the D2D network subject to the cost constraint and queue stability. We define a broadcasting action at time t as a two tuple: $(m, d_m(t))$, where $m = 1, 2, \ldots, M$ is the file being broadcast and $0 < d_m(t) \leq 1 - b_m(t)$ is the replacement step-size of broadcasting file m. We also define the silent action without broadcasting as $A^{\text{slt}} = (0, 0)$. Consequently, denoting $\mathcal{D}_m(t)$ as the set involving all possible step-sizes of broadcasting file m at time t, the action space at time t is $\mathcal{A}(t) = \mathcal{A}^{\text{br}}(t) \cup \{A^{\text{slt}}\}$, where

$$\mathcal{A}^{\text{br}}(t) = \{(m, d_m(t)) \mid m = 1, \ldots, M; d_m(t) \in \mathcal{D}_m(t)\}. \tag{3}$$

The cardinality of $\mathcal{D}_m(t)$ can be infinitely large since $d_m(t)$ is generally a real number. However, in practice, $\mathcal{D}_m(t)$ is finite because we only have finite number of users and because we can implement quantization. With the definition of the action space, our replacement problem is mathematically formulated as

$$\max_{P} \liminf_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{M} E\left[s_m^{A(t)}(t)\right] \tag{4a}$$

$$\text{s.t.} \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} E\left[c_{\text{inst}}^{A(t)}(t)\right] \leq C \tag{4b}$$

[3]The issue of extending the broadcast of multiple files at a time is still a subject for future research. As such, broadcasting multiple files within a short period is not too different from broadcasting only one single file at a time.

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E\left[Q_m^{A(t)}(t)\right] < \infty, \forall m, \tag{4c}$$

where $A(t) \in \mathcal{A}(t)$ is the action we take at time t according to some policy P; $c_{\text{inst}}^{A(t)}(t)$ is the cost of action $A(t)$; C is the cost constraint; and $Q_m^{A(t)}(t)$ is the size of queue m under a sequence of decisions $A(0), A(1), \ldots, A(t-1)$. Note that we use $Q_m(t)$ for the general purpose, whereas we use $Q_m^{A(t)}(t)$ to stress that the result is under the sequence $\{A(0), A(1), \ldots, A(t-1)\}$. Besides, the superscript $\cdot^{A(t)}$ is to explicitly indicate that decision sequence $A(t)$ influences the random processes. This concept applies to all notations in the remainder of this paper.

In the formulation, (4b) indicates that we need to follow a time-average cost constraint. Besides, (4c) indicates that we need to stabilize every queue such that all requests can be possibly served as long as they stay in the system [40].[4] Furthermore, note that $s_m^{A(t)}(t)$ in the objective function can be replaced by some other reward functions, such as number of bits. In this case, we need to use number of bits to represent our queue size. In addition, $s_m^{A(t)}(t)$ is indeed a function of the system parameter set $\mathcal{P}(t)$, which is subject to the actual file request and content delivery mechanism of the networks. However, to simplify the notation in the paper, we do not explicitly write dependence on $\mathcal{P}(t)$. Finally, although $c_{\text{inst}}^{A(t)}(t)$ can be different when we choose to conduct different actions, we simply assume here a constant cost when broadcasting different files and zero cost when being silent without broadcasting. Mathematically, we thus let $c_{\text{inst}}^{A(t)}(t) = c$ if $A(t) \in \mathcal{A}^{\text{br}}(t)$ and $c_{\text{inst}}^{A(t)}(t) = 0$ if $A(t) = A^{\text{slt}}$. Note that since conducting a broadcasting action should induce a much higher cost than being silent without broadcasting, a broadcasting action cannot always be performed. As a result, we generally set c to be larger than C.

IV. DRIFT-PLUS-PENALTY AIDED MINIMIZATION METHODOLOGY

Considering the replacement architecture proposed in Sec. III, our goal is to find a policy P that maximizes the time-average service rate while subject to queue stability and cost constraint as described in (4). However, solving (4) is a sequential decision-making problem, which is very challenging under general conditions and with large dimension. To solve this, we combine the drift-plus-penalty methodology in Lyapunov optimization [40] with the idea of "reward-to-go" [45], i.e., the reward in the future, to develop the policy design framework.

First, we define the reward-to-go for file m at time t in l time-slots as:

$$\tilde{R}_m(t, A(t)) = \frac{1}{l}\left(s_m^{A(t)}(t) + E\left[\sum_{\tau=1}^{l-1} s_m^{A(t+\tau)}(t+\tau)\right]\right), \tag{5}$$

where $A(t+\tau), \tau = 0, \ldots, l-1$ are actions determined by a policy P and $E\left[s_m^{A(t)}(t+\tau)\right]$ is the expected service rate in the τth time-slot after the considered time t. With this definition, we then formulate another optimization problem:

$$\max_{P} \liminf_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{M} E\left[\tilde{R}_m(t, A(t))\right]$$
$$\text{s.t. (4b), (4c),} \tag{6}$$

where $A(t), \forall t$, are determined by a policy P. We then provide the following Lemma:

Lemma 1: Suppose actions $A(t) \in \mathcal{A}(t), \forall t$, are determined by a policy P and the expected service rate is upper bounded by a finite number s_{max} as $E\left[s_m^{A(t)}(t)\right] \le s_{max}$. Accordingly, the following holds:

$$\lim_{T\to\infty} \frac{1}{T}\left(\sum_{t=0}^{T-1} E\left[s_m^{A(t)}(t)\right] - \sum_{t=0}^{T-1} E\left[\tilde{R}_m(t, A(t))\right]\right) = 0. \tag{7}$$

Proof: See Appendix A. □

Lemma 1 shows that the optimization problem in (4) is equivalent to that in (6). Besides, when $l = 1$, (6) automatically degenerates to (4). Thus, Lemma 1 explains the rationale of considering (6). To find the effective solution for (6), we consider using the drift-plus-penalty-minimization methodology. To define the drift, we first introduce a virtual cost queue:

$$Z(t+1) = \max\left(Z(t) + c_{\text{inst}}^{A(t)}(t) - C, 0\right), \tag{8}$$

where $0 \le Z(0) < \infty$ is the initial condition. We assume that the number of arrivals is bounded, i.e., $r_m(t) < \infty, \forall m$. Then by (2) and (8), we can obtain (9), as shown at the bottom of the next page, where

$$2B \ge \sum_{m=1}^{M}\left(r_m(t) - s_m^{A(t)}(t) - s_m^{\text{out}}(t)\right)^2 + \left(c_{\text{inst}}^{A(t)}(t) - C\right)^2 \ge 0$$

is a constant.

We define $L(t) = \frac{1}{2}\left[\sum_{m=1}^{M}(Q_m(t))^2 + (Z(t))^2\right]$ and define the drift as $\Delta(t) = L(t+1) - L(t)$. Consider a finite non-negative number V. The drift-plus-penalty is then bounded as in (10), shown at the bottom of the next page. A policy that selects actions by minimizing the drift-plus-penalty in (10b) leads to the following theorems:[5]

[4]Note that due to the physical constraints in practice, it is possible that queues $Q_m(t)$ are bounded inherently. However, in this case, it is still meaningful to derive an algorithm based on the notion that queues can become infinite. This is because any algorithm derived for a limiting case will work close to optimum for a finite, but sufficiently large, n. Consequently, we are devising an algorithm assuming a large number of requests, which requires queue stabilization, and then apply this algorithm to cases of inherently finite queue lengths.

[5]We note that by following the concept of the Bellman's principle of optimality, the necessary condition for an optimal policy P is that, while subject to the queuing stability and cost constraint, P can at each time slot maximize $\tilde{R}_m(t, A(t))$ with $l \to \infty$. Consequently, when l and V tend to infinity, the drift-plus-penalty-minimization can satisfy the optimality condition. However, since directly finding $\tilde{R}_m(t, A(t))$ with $l \to \infty$ might not be possible, we need to resort to minimizing (6) with finite l and V for finding a feasible solution.

Theorem 1: Suppose M, V, $Z(0)$, and $Q_m(0), \forall m$, are some finite numbers. Assume $r_m(t) \leq r_{\max}, \forall m$, are finite and bounded; $C > 0$ and $c_{\text{inst}}^{A(t)}(t), \forall A(t) \in \mathcal{A}(t)$, are also finite and bounded. If the adopted policy chooses the action $A(t) \in \mathcal{A}(t)$ such that (10b) is minimized for all t, then $Q_m^{A(t)}(t), \forall m, \forall t$, are upper bounded. Accordingly, constraints in (4c) are satisfied, i.e., every queue is stable. Moreover, the time-average cost constraint in (4b) is satisfied.

Proof: See Appendix B. □

Theorem 2: Assume $\sum_{m=1}^{M} E[Q_m(t)] \leq \epsilon V$, $E[Z(t)] \leq \delta V$, $r_m(t) \leq r_{\max}$, and $c_{\text{inst}}^{A(t)}(t) \leq C_{\max}$ for some finite positive ϵ, δ, $r_{\max}$, and $C_{\max}$. Assume that $\sum_{m=1}^{M} \tilde{R}_m(t, A(t))$ is finite and upper bounded. When the actions $A(t) \in \mathcal{A}(t), \forall t$, are determined by a policy P, there must exist a finite non-negative number y^* such that

$$\liminf_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{M} E\left[\tilde{R}_m(t, A(t))\right] \geq y^* - \frac{B}{V} - \epsilon r_{\max} - \delta C_{\max}. \tag{11}$$

Furthermore, y^* can be maximized when (10b) is minimized at all t.

Proof: See Appendix C. □

By observing the proof of Theorem 1, $Q_m^{A(t)}(t), \forall m$, and $Z(t)$ are upper bounded. Therefore, the prerequisite of Theorem 2 can be realized. Theorem 2 indicates that minimizing (10b) can effectively maximize y^*. In addition, V controls the trade-off between the performance of the reward-to-go and the real and cost queue lengths. When $V = 0$, Theorem 2 induces a trivial lower bound. However, this does not necessarily mean that the time-average service rate in this situation is very poor. This is because even if $V = 0$, we can still stabilize the queuing system, which implicitly provides good service rate. In this context, the inclusion of the penalty term can be interpreted as a means of controlling the optimization of the service rate. Finally, we show a lower bound performance of the proposed design using Theorem 3:

Theorem 3: Assume that there exists a randomized policy Θ that is i.i.d. with respective to time t and independent to $Q_m(t)$, $Z(t)$, and to $B_m(t), \forall m$, such that the following is satisfied:

$$E\left[c_{\text{inst}}^{\Theta}(t)\right] - C \leq \delta; \quad E\left[s_m^{\Theta}(t)\right] \geq y_m^{\Theta}, \forall m;$$
$$E\left[\sum_{m=1}^{M} \left(r_m(t) - s_m^{\Theta}(t)\right)\right] \leq \delta, \tag{12}$$

where δ could be arbitrary small. Suppose $A(t) \in \mathcal{A}(t), \forall t$, are determined by a policy P that minimizes (10b). Then, the following is satisfied:

$$\liminf_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{M} E\left[\tilde{R}_m(t, A(t))\right] \geq \sum_{m=1}^{M} y_m^{\Theta} - \frac{B}{V}. \tag{13}$$

Proof: See Appendix D. □

Theorem 3 indicates that the drift-plus-penalty minimization approach can be better than an arbitrary randomized design. This characterizes a lower bound performance of the drift-plus-penalty-minimization methodology.

The above results show the benefits of using the drift-plus-penalty methodology to design a policy. However, directly minimizing (10b) can be very difficult or even impossible due to the need of computing $\sum_{m=1}^{M} \tilde{R}_m(t, A(t))$. We thus propose in Secs. V and VI two alternative designs that can be practiced to help resolve this issue.

V. MYOPIC DRIFT-PLUS-PENALTY AIDED MINIMIZATION REPLACEMENT

In this section, we propose the first design which myopically minimizes the drift-plus-penalty, i.e., the drift-plus-penalty minimization is performed without considering the future payoff. Observe that when $l = 1$, the drift-plus-penalty can be bounded as in (14), as shown at the bottom of the next page, where $X \geq \sum_{m=1}^{M} Q_m(t) r_m(t) \geq 0$ is a constant-bound given that $Q_m(t), \forall m$, are upper bounded (see Theorem 4 later); (a) is because $Q_m(t) s_m^{A(t)}(t) \geq 0, \forall m$.

The original drift-plus-penalty methodology aims to minimize the first inequality in (14). However, when the D2D

$$\sum_{m=1}^{M} [Q_m(t+1)]^2 + [Z(t+1)]^2 \leq \sum_{m=1}^{M} \left[Q_m(t) + r_m(t) - s_m^{A(t)}(t) - s_m^{\text{out}}(t)\right]^2 + \left[Z(t) + c_{\text{inst}}^{A(t)}(t) - C\right]^2$$
$$\leq \sum_{m=1}^{M} [Q_m(t)]^2 + [Z(t)]^2 + 2\left[\sum_{m=1}^{M} Q_m(t)\left(r_m(t) - s_m^{A(t)}(t) - s_m^{\text{out}}(t)\right) + Z(t)\left(c_{\text{inst}}^{A(t)}(t) - C\right)\right] + 2B. \tag{9}$$

$$\Delta(t) - V\sum_{m=1}^{M} \tilde{R}_m(t, A(t)) \leq \sum_{m=1}^{M} Q_m(t)(r_m(t) - s_m^{A(t)}(t) - s_m^{\text{out}}(t)) - V\sum_{m=1}^{M} \tilde{R}_m(t, A(t)) + Z(t)(c_{\text{inst}}^{A(t)}(t) - C) + B \tag{10a}$$
$$\leq \sum_{m=1}^{M} Q_m(t)(r_m(t) - s_m^{A(t)}(t)) - V\sum_{m=1}^{M} \tilde{R}_m(t, A(t)) + Z(t)(c_{\text{inst}}^{A(t)}(t) - C) + B. \tag{10b}$$

scheduler is complicated, $s_m^{A(t)}(t)$ might not have an analytical expression that is easy to compute or estimate under different actions. Thus, we use the final inequality in (14) and develop a simplification that is based on the following observation: if we choose to broadcast file m at time t, then we can immediately know $s_m^{A(t)}(t) = Q_m(t) + r_m(t)$ since the broadcast can satisfy all requests for file m at time t. Besides, since we assume no cost for silence, we also know that a sufficient condition to choose to be silent is:

$$-(Q_m(t) + V)s_m^{A_m}(t) + Z(t)c_{\text{inst}}^{A_m}(t) > 0, \forall m, \tag{15}$$

where $A_m \in \mathcal{A}^{\text{br}}(t)$ denotes any action broadcasting file m. By previous observations, we solve the following optimization problem for making the decision:

$$A(t) = \arg\min_{A\in\mathcal{A}(t)} g_{A,V}(t), \tag{16}$$

where $g_{A,V}(t) = -\left[\sum_{m=1}^{M} \mathbf{1}_{\{A=A_m\}} \cdot (Q_m(t) + V)s_m^{A}(t)\right] + Z(t)c_{\text{inst}}^{A}(t)$ and $\mathbf{1}_{\{A=A_m\}}$ is an indicator function that has value of 1 only if the BS broadcasts file m. Note that when $A = A^{\text{slt}}$, $g_{A^{\text{slt}}}(t) = 0$ and when $A = A_m$, $g_{A_m,V}(t) = -(Q_m(t) + V)(Q_m(t) + r_m(t)) + Z(t)c_{\text{inst}}^{A_m}$. As a result, solving (16) is very simple. The intuition in the solution to (16) is that the system tends to broadcast the file with higher pressure on the queue provided that the pressure in the virtual (cost) queue is sufficiently low.

The complete replacement approach is to solve (16) and decide the action at every time slot. Since (16) can be easily solved, the complexity of the approach is low. Also, since the proposed approach here exploits only the history record (queue sizes) and the current system state without using any future information, this approach is named myopic drift-plus-penalty minimization (MyDPP) replacement. Note that MyDPP cannot distinguish the differences in step-sizes when we broadcast the same file; thus this approach cannot adaptively select step-sizes. Consequently, when implementing the MyDPP replacement, we consider a compressed broadcasting action space $\mathcal{A}^{\text{cp}}(t)$, in which a constant step-size d is adopted for any broadcasting action. Mathematically, this indicates $\mathcal{A}(t) = \mathcal{A}^{\text{cp}}(t) \cup \{A^{\text{slt}}\}$ for the MyDPP replacement, where

$$\mathcal{A}^{\text{cp}}(t) = \{(m, d_m(t)) \mid m = 1, \ldots, M; d_m(t) = d\} \tag{17}$$

Note that the constant step-size d of the replacement procedure should be carefully selected right at the beginning.

Algorithm 1 Proposed MyDPP Replacement Design

```
1:  Init: Start at $t = 0$, $Q_m(0) \geq 0, \forall m$, $Z(0) \geq 0$. Set step-size $d(t) = d$ and $V \geq 0$.
2:  for $t = 0, 1, \ldots$ do
3:      Evaluate $g_{A_m,V}(t) = -(Q_m(t)+V)(Q_m(t) + r_m(t)) + Z(t)c_{\text{inst}}^{A_m}, \forall m$
4:      if $\min_{m=1,\ldots,M} g_{A_m,V}(t) < 0$ then
5:          Broadcast the file $m$, where $m = \arg\min_{m=1,\ldots,M} g_{A_m,V}(t)$
6:          Conduct the replacement procedure provided in Sec. III with step-size $d$
7:      else
8:          Keep silent, i.e., $A(t) = A^{\text{slt}}$
9:      end if
10:     Update the real queues $Q_m(t), \forall m$, and the virtual queue $Z(t)$
11: end for
```

Finally, the overall algorithm of MyDPP replacement is summarized in Alg. 1. In addition, the proposed MyDPP approach can guarantee the time-average cost constraint and stabilize the queues according to the Theorem 4:

Theorem 4: Suppose $M, V, Z(0)$, and $Q_m(0), \forall m$, are some finite numbers. Consider $r_m(t) \leq r_{\max}, \forall m$, are finite and bounded; $C > 0$ and $c_{\text{inst}}^{A(t)}(t), \forall A(t) \in \mathcal{A}(t)$, are also finite and bounded. Considering using the MyDPP policy, then $Q_m^{A(t)}(t), \forall m, \forall t$, are upper bounded. Accordingly, constraints in (4c) are satisfied. Moreover, the time-average cost constraint in (4b) is satisfied.

Proof: The proof follows the similar approach in Theorem 1. We thus omit it for brevity. □

VI. DRIFT-PLUS-PENALTY AIDED MINIMIZATION REPLACEMENT EXPLOITING SAMPLING

Next, we derive a method that exploits the potential future information, i.e., the knowledge about future changes, in the popularity distribution and the corresponding payoff. Guessing the future popularity distribution (e.g., which videos will become "viral") is a widely investigated topic; thus we will not be discussed further. Instead, we simply assume here that such future information is available on the BS. Besides, we assume the operation of the network can be modeled and simulated via Monte-Carlo methods. Accordingly, we

$$\begin{aligned}
\Delta(t) - V\sum_{m=1}^{M}\tilde{R}_m(t, A(t)) &\leq \sum_{m=1}^{M} Q_m(t)(r_m(t) - s_m^{A(t)}(t)) - V\sum_{m=1}^{M} s_m^{A(t)}(t) + Z(t)(c_{\text{inst}}^{A(t)}(t) - C) + B \\
&\leq -\sum_{m=1}^{M} Q_m(t)s_m^{A(t)}(t) - V\sum_{m=1}^{M} s_m^{A(t)}(t) + Z(t)c_{\text{inst}}^{A(t)}(t) - Z(t)C + X + B \\
&\overset{(a)}{\leq} -(Q_m(t) + V)s_m^{A(t)}(t) + Z(t)c_{\text{inst}}^{A(t)}(t) - Z(t)C + X + B, m = 1, 2, \ldots, M.
\end{aligned} \tag{14}$$

propose the second design, which decides on the actions to take with the aid of future information (i.e., $l > 1$).

A. PROPOSED REPLACEMENT EXPLOITING SAMPLING AND ROLLING HORIZON

To introduce the future information and to satisfy the constraints in (4), we first need to minimize (10b) with finite V and $l > 1$. However, computing for $\tilde{R}(t, A(t))$ at each time might be impossible and/or can be very complex. Therefore, we propose an alternative approach for estimating $\tilde{R}(t, A(t))$. Besides, to reduce complexity, we aim to skip such estimation for some time-slots. As such, we observe that on one hand, we should not broadcast if the cost queue is already highly pressured. On the other hand, if we broadcast, then the algorithm shall select the action that provides the highest reward-to-go. This observation then breaks down the decision-making problem into two sub-problems: (i) whether to conduct a broadcast with replacement; and (ii) which file to broadcast with what step-size. We then solve the sub-problems sequentially by exploiting the drift-plus-penalty methodology.

When $V = 0$ and $l = 1$, the drift-plus-penalty approach leads to the most stable and cost effective network, which indicates that such approach is conservative. Thus, we solve the problem with $V = 0$ and $l = 1$ to decide whether to conduct a broadcast, i.e., we exploit the MyDPP approach with $V = 0$ in Sec. V to decide whether to broadcast a file.

When we decide to broadcast a file, we need to select the specific file and the step-size. In this case, we consider $V = \infty$[6] to optimize this decision, and then introduce the Monte-Carlo sampling along with a probabilistic candidate selection approach to estimate $\tilde{R}(t, A(t))$. Suppose we have decided to broadcast and conduct a replacement. We first construct the candidate set that includes all the possible broadcasting candidates. Recall that when we consider to broadcast at time t, we select an action from $\mathcal{A}^{\text{br}}(t)$ in (3). Accordingly, the candidate set $\Pi(t)$ is constructed by including all possible broadcasting actions, i.e., all possible combinations of the broadcasting files and step-sizes:

$$\Pi(t) = \{\pi = (m, d_m) | m = 1, 2, .., M; d_m \in \mathcal{D}_m(t)\}. \quad (18)$$

We then use the proposed Monte-Carlo based sampling to select the best action. Suppose we are in time slot t. A Monte-Carlo sample of a candidate $\pi = (m, d_m)$ is derived by using the following T_{stage} stage simulation procedure:

1) At the simulation time $k = t$,[7] we broadcast file m with a step-size d_m, and then simulate the system using Monte-Carlo method and record $\hat{R}(k, A(k), W(k))$, where $\hat{R}(k, A(k), W(k))$ is the sampling reward with randomness $W(k)$ at time k.
2) At simulation time $k = t + 1$ to $t + T_{\text{stage}} - 1$, we simulate the system with $A(k) = A^{\text{slt}}$, i.e., the system is silent, and record $\hat{R}(k, A(k), W(k))$.
3) Output the estimated reward-to-go of candidate π:
$$\tilde{R}_t(\pi) = \sum_{k=t}^{t+T_{\text{stage}}-1} \hat{R}(k, A(k), W(k)).$$

We note that we assume the operation of the system can be modeled and simulated effectively. Besides, T_{stage} needs to be carefully chosen to provide effective approximations. Since we conduct simulations considering only a single broadcast in T_{stage} time-slots, T_{stage} is suggested to be the average number of time slots between two replacement actions. This is because, by definition, the system should remain silent between two replacement actions. Note that cost constraint and the cost of each broadcast determines the average number of time slots between two replacement actions. For example, if the broadcasting cost is $c^A_{\text{inst}}(t) = c, \forall A \in \mathcal{A}^{\text{br}}(t)$, then we can on average broadcast only once every $\frac{c}{C}$ time slots.

We now describe the candidate selection following the idea in [41]. Suppose we acquire N samples for each candidate at time slot t. Denote the selection probability for a candidate π as $\Psi_n(\pi)$ when considering n samples, where $\sum_{\pi\in\Pi} \Psi_n(\pi) = 1$. For a candidate $\pi \in \Pi$, the update of the selection probability is:

$$\Psi_n(\pi) = \frac{(\beta_\pi)^{\tilde{R}_{t,n}(\pi)}}{\sum_{\pi\in\Pi} (\beta_\pi)^{\tilde{R}_{t,n}(\pi)}} \Psi_{n-1}(\pi), n = 1, \ldots, N, \quad (19)$$

where β_π is the annealing coefficient of candidate π and $\tilde{R}_{t,n}(\pi)$ is the sampling reward-to-go for sample n of candidate π at time t. We then use the selection probability $\Psi_N(\pi), \forall\pi \in \Pi$ to decide which file to broadcast and what its corresponding step-size should be; that is, we decide the final action according to the sample that the distribution $\Psi_N(\cdot)$ randomly generates. The initial selection probabilities can be any distribution such that $\sum_{\pi\in\Pi} \Psi_0(\pi) = 1$. However, we usually consider the uniform distribution for initialization. We stress that, according to Theorem 3.1 in [41], when N tends to infinity, this selection approach converges to the optimal distribution that offers the optimal reward based on the given sampling procedure and on the candidate set.

We now summarize the proposed replacement approach in this sub-section as follows and in Alg. 2. At each time t, we first decide whether to broadcast using Alg. 1. If the result of Alg. 1 suggests broadcasting, then we enter the next phase, in which we decide the broadcasting file and the step-size; otherwise, the system remains silent. If we broadcast, then we need to construct the candidate set Π and use Monte-Carlo sampling to acquire reward-to-go samples. We then compute for the final selection distribution $\Psi_N(\cdot)$ using (19); the action, including both the broadcasting file and step-size, is determined by using a random sample of the selection distribution. The replacement approach proposed here is named sampling based drift-plus-penalty (SPDPP). Compared with MyDPP, SPDPP can adaptively adjust the step-size and exploit the future benefits to make decisions. Besides, the proposed SPDPP replacement can also satisfy the required constraints. This is

[6]In practice, different V could be considered for different tradeoffs. However, this does not change the essence of our design.

[7]Note that we conduct the system simulation starting at $k = t$

Algorithm 2 Proposed SPDPP Replacement Design

```
1:  Init: Set $Q_m(0) \geq 0, \forall m$, $Z(0) \geq 0$, and the number of samples $N$
2:  for $t = 0, 1, \ldots$ do
3:      Evaluate $g_{A_m,0}(t) = -Q_m(t)\,(Q_m(t) + r_m(t)) + Z(t)c_{\text{inst}}^{A_m}, \forall m$
4:      if $\min_{m=1,..,M} g_{A_m,0}(t) < 0$ then
5:          Construct the candidate set $\Pi$
6:          Compute $\Psi_N(\cdot)$ using (19) with the proposed sampling procedure
7:          Select the action: $(m, d_m) = \pi \sim \Psi_N(\cdot)$
8:          Broadcast file $m$ and conduct the replacement procedure with step-size $d_m$
9:      else
10:         Keep silent, i.e., $A(t) = A^{\text{slt}}$
11:     end if
12:     Update the real queues $Q_m(t), \forall m$, and the virtual queue $Z(t)$
13: end for
```

because we use the same approach as MyDPP to decide whether to broadcast or not. We thus omit the proofs for brevity.

B. COMPLEXITY REDUCTION APPROACH

Alg. 2 considers all possible broadcasting files and step-sizes as candidates and uses a pre-determined sample size N. However, it is sometimes unnecessary to go through all candidates and use up to N samples for every candidate. In this section, we discuss some approaches to make Alg. 2 less complex. Specifically, we aim to use the algorithm itself to decide the number of candidates and samples. We therefore propose two complexity reduction approaches that can be used simultaneously.

1) INITIAL CANDIDATE NUMBER REDUCTION

In some situations, some files are redundantly cached to the point that we even want to decrease their percentages in the network. Thus, we do not have to include them in the candidate set. To identify those files, we observe that we broadcast only if there exists a file m such that $g_{A_m,0}(t) < 0$. This indicates that it is more necessary to broadcast files with $g_{A_m,0}(t) < 0$. Thus, we can include only those files in our candidate set. Note that this approach might result in the drop of the optimal solution. However, the probability for this to occur can be reduced by setting some lower bound on the minimal number of files to be included in the candidate set, and then adding files with smaller $g_{A_m,0}(t)$ in an ascending order. In addition, we can also set up a hard constraint for the maximum number of files included in the candidate set. Although this might result in the loss of the optimal solution, it could also effectively bound complexity in practice.

2) SAMPLING WITH CANDIDATE PRUNING

We can adaptively prune the candidates to reduce the number of samples per candidate during the sampling process. Recall that the update of the selection distribution $\Psi_n(\cdot)$ is a sequential update. Thus, instead of completely generating $\tilde{R}_{k,n}(\pi), \forall \pi, n$, and then find the final selection distribution, we can gradually generate the samples and update the selection distribution; that is, we generate $\tilde{R}_{k,1}(\pi), \forall \pi$, and then compute for $\Psi_1(\cdot)$; generate $\tilde{R}_{k,2}(\pi), \forall \pi$, and then compute for $\Psi_2(\cdot)$; and so on. In updating the selection distribution, when there is a candidate π such that $\Psi_n(\pi) < \epsilon$, it is improbable that this candidate would be selected. Hence, we set $\Psi_n(\pi) = 0$ and normalize $\Psi_n(\cdot)$ such that their sum is still equal to one. When $\Psi_n(\pi) = 0$, we then know that it is never selected. Thus, π is pruned from the candidate set, and we no longer need to generate more samples for this candidate. This process continues until either there exists a candidate π such that $\Psi_n(\pi) = 1$ or until $n = N$ is reached. This approach can reduce the number of candidates during the sampling process, and can allow to terminate the process earlier. It is clear that when $\epsilon \to 0$, this approach tends to maintain optimality asymptotically.

VII. EXTENSION TO CACHING MULTIPLE FILES

For the convenience of elaborating the designs and fundamental concepts, we assumed in the previous sections that each user would cache only one file, i.e., $S = 1$. Here, we describe how we extend the proposed designs to the networks such that the users can cache multiple files, i.e., $S > 1$. As discussed previously, a caching content replacement is constituted by deciding which file should be newly cached by users and which file should be removed. To extend the proposed designs, we first extend the replacement procedure in Sec. III to determine what files to remove from the users when $S > 1$. Suppose we want to increase $b_m(t)$ by $d_m(t)$. We first find all users who do not cache file m. Among those users, we randomly select N_{rep} users such that $b_m(t)$ can increase $d_m(t)$ if those users newly cache file m. Recall that the N_{rep} defined in Sec. III is the integer that can provide the closest approximation to the desired step-size. When the selected users receive the broadcast file m, they need to decide which file to remove from their caches in order to cache file m. To make the decision, each user looks at the files in their own caches and removes the file that has the smallest corresponding queue size. Clearly, such decision follows the similar intuition as that discussed in Sec. III – we remove the file whose corresponding queue has the lowest pressure. With this extended replacement procedure, our designs, aiming to decide what file should be newly cached by users, can directly be applied to the networks. Thus, to conduct the replacement in networks with $S > 1$, we first decide when and which file to newly cache by using the same approaches as those proposed in Secs. V and VI, and then use the extended replacement procedure to decide which file should be removed by which user.

VIII. PERFORMANCE EVALUATIONS AND DISCUSSIONS

In this section, we use simulations to evaluate the proposed replacement designs and provide relevant discussions. Note that although we need to consider a specific file request and content delivery mechanism in the following simulations for the purpose of obtaining numerical results, this does not mean that our proposed framework and algorithms are restricted to it.[8]

A. SIMULATION ENVIRONMENT

In all simulations, we consider 4000 users located (and possibly moving) within a square-shaped area, with side length 1000 m. The BS is located at the center of this square and serves as the central controller. The service coverage of the BS, i.e., the cell, also covers a square-shaped area, with side length 500 m. The consideration of a simulation area that is larger than the serving area is to emulate the users' behavior, which moves in and out of the coverage region. D2D communication is implemented based on the clustering of the users in a cell, as has been widely adopted for D2D based video caching [9], [18], [20], [22]. In particular, the cell is split into several smaller and equal-sized sqaure clusters, where only users within the same cluster can communicate with each other. We denote the side length G of a cluster as the cluster size. To avoid interference, a spatial reuse scheme is employed, i.e., only clusters that are a minimum distance apart from one other may use the same time/frequency resources, similar to cellular frequency reuse. Thus, the size of a cluster, also interpreted as the cooperation distance, can greatly affect the throughput and outage performance. All communications within a cluster use the same data rate regardless of the distance between the users, corresponding to a fixed modulation and coding scheme. In all simulations, D2D links have a service rate of 200 Mbits/s. This service rate is feasible when we adopt mmWave communications or when we apply reuse factor one along with the advanced WiFi service. To be able to use either approach, we consider the cluster size G to be upper bounded by 100 m [9], [20]. All users generate requests according to a request distribution. In a cluster, users fulfill requests from files in the local cache whenever possible. Otherwise, the requests are sent to the BS. Among the requests (in the same cluster) that *can* be fulfilled via D2D communications, the BS randomly selects one such request to satisfy. The above D2D scheduling and delivery generally follow the priority-scheduling as that detailed in [20]. We consider here users cannot be served by user-specific BS links, but can be served by broadcasting of the BS. When the BS broadcasts file m (for both replacement and service), all user requests in the cell for file m are satisfied and the queue of file m is cleared. Control overhead is ignored in simulations for simplicity.

We model the service using a slotted structure and then evaluate the performance in terms of the number of requests satisfied per slot, which include the requests satisfied by self-caching, D2D communications, and BS broadcasting. We consider a slot length of 6 s and simulate $T = 14400$ time slots (complete 24 hours) to obtain one sample result. This setup allows the users to finish downloading a file whose size is 150 MB within each slot. Note that this file size is enough to provide around 30 minutes of video with fairly good quality. We adopt the mobility model in [25], which directly connects to the user velocity and the random waypoint model in [46] such that we can model the user movement. Each user u in the mobility model randomly selects a target point within the simulation area, i.e., within the 1 km^2 area, and moves toward the target point with a constant velocity. To decide the velocity of the movement, each user u randomly selects the velocity in $[0, 2V_u]$, where V_u is the average velocity of this user. V_u is randomly selected from $[0, 2V_{\text{net}}]$ at the beginning of the simulations, where $V_{\text{net}} = 1$ m/s (3.6 km/h) is the average velocity in the network, which corresponds to a fast walking speed. The general mobility pattern is as follows. Each user first picks a target point, selects the velocity for this trip, and then moves toward the target. Since we adopt the slotted structure, each user checks whether the moving distance is sufficient to reach the target point at the end of each time slot. If yes, then the user chooses another target point and velocity for a new trip; if not, then the user keeps moving toward the same target point until it arrives.

A user can either be in an active or inactive mode. When the request of an active user is satisfied, the user immediately transits to inactive. Each user can change its mode at the end of each time slot, and the probability of changing mode is 0.05. When a user changes from active to inactive, the request of the user is dropped from the queueing system, thereby causing outage. Conversely, a user's request is generated according to the request distribution at the time that a user changes from an inactive to an active. This request is accordingly sent to the BS at the beginning of the next time slot if the local cache cannot satisfy the request. A user can move in and out of the cell. When a user moves out of the cell at the end of the time slot, the request of the user is dropped from the network, and the BS loses the information of the user. On the other hand, when a user moves into the cell, the user can either be in an active or inactive mode with equal probability. If the user is in active mode, then the request is generated according to the request distribution at that time slot.

We consider a single update of the request distribution per hour, i.e., a single update every 600 time slots. The request distribution update is always the last function to be conducted in a time slot. In each update, K new files are added into the library and become the most popular K files. Thus, the rank of all the original files should degrade by K. In addition, the originally least popular K files are dropped from the

[8]Simulation results under a different simulation environment can be found in the conference version [1]. Although we present only the results of the MyDPP approach in [1], the results still demonstrate the generality of our replacement framework. Moreover, although we cannot analytically characterize nor empirically demonstrate the optimality of the proposed designs in complex networks, we still numerically show that the proposed design is near-optimal in a very simplified scenario (see Fig. 2 in [1]).

library, indicating the users are no longer interested in those files. Aside from adding and dropping files, the concentration rate of the request distribution can change at each update. We model the request distribution by using a Zipf distribution [20] with Zipf parameter $\gamma = 0.2 + 0.5(k-1), k = 1, 2, \ldots, 25$. The change of index k, indicates the change of the concentration rate, and we model this using a Markov process with a transition probability matrix P, in which $P_{k,k} = 0.5, P_{k,k+1} = 0.25, P_{k,k-1} = 0.25$, where $2 \leq k \leq 24$; $P_{1,1} = 0.5, P_{1,2} = 0.5, P_{25,25} = 0.5, P_{25,24} = 0.5$; $P_{k,l} = 0$, otherwise.

Due to users' mobility, we also need to consider the outage caused by those users moving away from each other during the transmission. This condition is called "mobility outage". Notice that users are guaranteed to communicate with each other only if they are in the same cluster. Thus, mobility outage occurs when two users that have established D2D communications at the beginning of a time slot are not in the same cluster at the end of the time slot. Once an outage occurs, the request is not satisfied, and the user remains active with the same request. Note that when users are served by the broadcasting from the BS, such mobility outage does not happen.

To initialize a simulation, we adopt the following procedures: (i) all users are uniformly distributed within the square with side length 1000 m; (ii) users located within the BS service area are set to active mode, whereas the users located outside the BS service area are set to inactive mode; (iii) every user randomly selects their average velocities used during the simulation, and then initializes a new trip by using the described mobility model; and (iv) the initial request distribution is set at index $k = 13$, i.e., $\gamma = 0.8$.

In all the simulations below, MATLAB$^{\text{TM}}$ is used to build up our simulation environment. We run simulations on a server with 72 CPU cores. Each core has a rate of 2.1 GHz.

B. SIMULATION RESULTS

Now, we evaluate the proposed designs. We present our results by their sample means (specific points) and sample deviations (error bars). In all simulations, we consider $C = 1$ and $c^{A}_{\text{inst}} = 20, \forall A \in \mathcal{A}^{\text{br}}(t)$. This means that on the average, the broadcasting action happens once per 20 time slots. In the MyDPP approach, $V = 0$ is considered[9] and different step-sizes (indicated in the legends of the figures) are used. In the SPDPP approach, we consider $T_{\text{stage}} = 20$, $N = 10$, $\beta_{\pi} = 1.3, \forall \pi \in \Pi$, and $\mathcal{D}_m(t) = \{d_m \mid (1 - b_m(t))/2^k > d_{\min}, k = 0, 1, \ldots, \} \cup \{d_{\min}\}$, where $d_{\min} = 0.001$ is the minimal step-size. We use the complexity reduction approaches in Sec. V.C for SPDPP. The minimal and maximal number of candidate files are 2 and 4, respectively.[10] The threshold for pruning a candidate is $\epsilon = 10^{-6}$. In Figs. 1 and 2, to focus on evaluating the performance of the replacement designs, the mobility outage is temporarily excluded. Then in the remaining figures, the influence of such outage is included. All the proposed replacement designs can satisfy the cost constraint within $\delta < 0.005$ accuracy, i.e., $\frac{1}{T}\sum_{t=0}^{T-1} c^{A(t)}_{\text{inst}}(t) \leq C + \delta$ with high probability, and accordingly stabilize the queueing system in the simulations. This is not shown in the figures for brevity.

In the following dicussion, we demonstrate the performance of the proposed replacement designs and compare them with static approaches. In all figures, "Zipf-0.8" indicates a time-invariant caching policy based on a Zipf distribution with parameter 0.8 [22]; "Brod" indicates that the BS periodically broadcasts, i.e., the BS broadcasts the files in a round-robin manner every 20 time slots, but does not conduct replacement. The "Zipf-0.8" policy is also used as the initial caching policy for the replacement designs. Since we focus on demonstrating the performance of the replacement designs, we do not try to optimize the static policy. Besides, we adopt this policy because it is simple to use and performs well [22] as it matches the initial request distribution, which also has the Zipf parameter $\gamma = 0.8$.

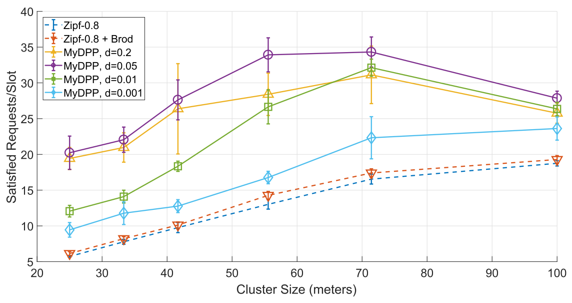


FIGURE 1. Throughput as a function of cluster size of the MyDPP replacement with different step-sizes.

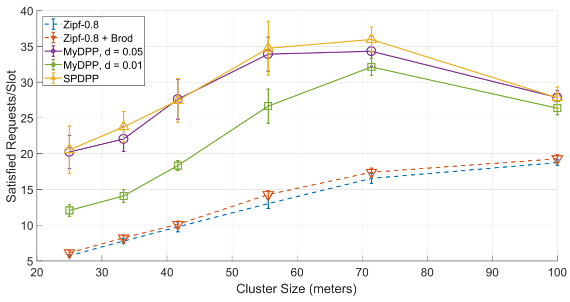
In Fig. 1, $S = 1$, $M = 100$, and $K = 3$ are considered. We observe that the choice of step-size indeed significantly influences the results, and the optimal step-size depends on the adopted parameters and network configurations. Clearly, the best step-size cannot be obtained before we actually run the simulations, thereby preventing the real-time optimization. Fortunately, we can still obtain a somewhat efficient step-size by looking at the concentration rate of the request distribution. From experience with our simulations, the step-size performs well when it is on the order of the popularity of the most popular files, e.g., $d = 0.05$ in the figure.[11] Besides, when the caching distribution is inappropriate, having a larger cluster size could improve performance.

[9]Although different V can entail different trade-off by theorems. However, the low-complexity implementation is merely the approximation of the exact drift-plus-penalty minimization; thus, the trade-off entailed by V in MyDPP is not very unclear. We thus choose the most cost-effective case ($V = 0$) for the demonstrations.

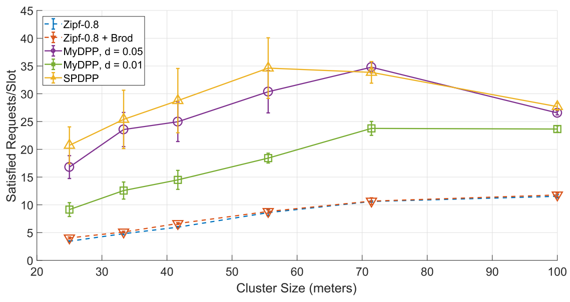
[10]Of course, a candidate file can have different step-sizes and recall that a final candidate is jointly determined by the candidate file and step-size.

[11]A step-size of $d = 0.05$ is optimum to use for the simulations discussed in this paper. In simulations that consider different network setups, however, a different d could be the optimum.

This is intuitive because when the caching distribution is inappropriate, we need to enlarge the cluster size to increase the probability that a user can find the desired file in the cluster. For example, in Fig. 1, the MyDPP with $d = 0.05$ has the best performance when cluster size is within the range of $60 - 70$ m, whereas the MyDPP with $d = 0.01$ performs best when it is around 71 m. This is because when the step-size is $d = 0.01$, the replacement might not be fast enough to adjust the user caches such that the new files can be accommodated within a short period after the request distribution is updated. Finally, we observe that all proposed designs perform better than the static approaches and outperform the MyDPP with extremely small step-size ($d = 0.001$). This validates the benefits of having appropriate replacement even when some type of broadcasting is used. Note that when d is very small, the MyDPP is very close to simply providing appropriate broadcasting without cache content replacement.



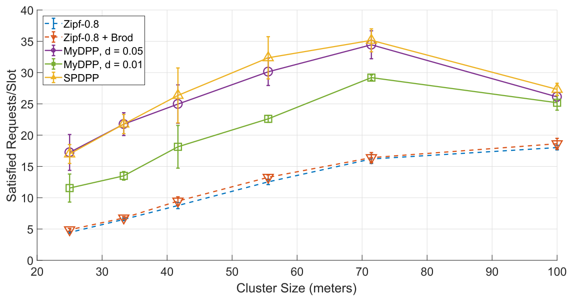
(a) $K = 3$.



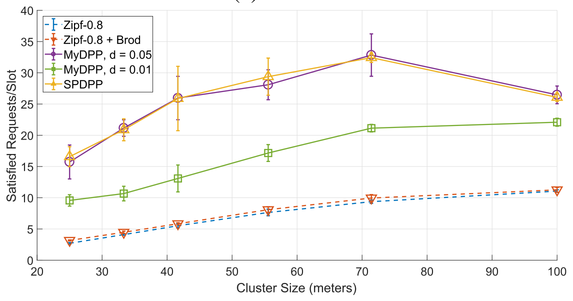
(b) $K = 6$.

FIGURE 2. Throughput as a function of cluster size under different replacement schemes.

We now compare MyDPP and SPDDP in Fig. 2. We assume $S = 1$ and $M = 100$ in the figures, and $K = 3$ and $K = 6$ in Fig. 2a and Fig. 2b, respectively. We observe that the proposed SPDPP replacement performs the best without needing to manually select the appropriate step-size. The proposed MyDPP design is comparable with the SPDPP design when we optimize the step-size. The benefit of MyDPP is that it is less complex and does not need predictive information, although a suitable step-size still needs to be selected for the replacement. All the proposed replacement designs demonstrate significant improvement when compared to the static policy. In Fig. 3, we compare the performance of the same network under the different replacement schemes, similar to that done in Fig. 2. This time, however, we consider



(a) $K = 3$.



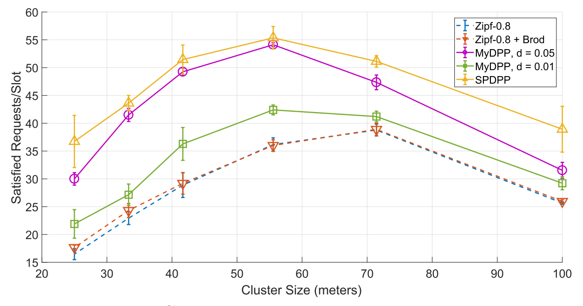
(b) $K = 6$.

FIGURE 3. Throughput as a function of cluster size under the different replacement schemes in networks with mobility outage.

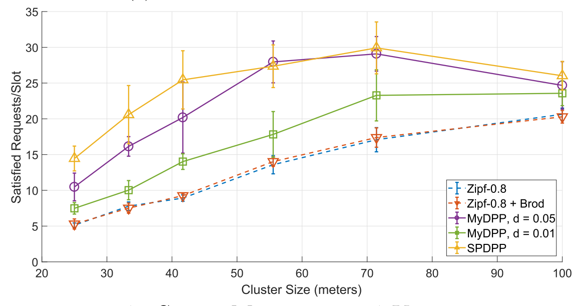
the influence of mobility outage in the analysis. From the figure, we gather the same observations as those in Fig. 2. Besides, by comparing Fig. 2 with Fig. 3, we observe that the performance in Fig. 3 slightly degrades due to the mobility outage, and such degradation is larger when the cluster size is smaller. This is intuitive because when the cluster size is small, it is more likely to suffer from mobility outages.

In Fig. 4, we evaluate the proposed designs in networks where the user can cache multiple files. The replacement design is implemented following the extension approach proposed in Sec. VII. We consider $S = 5$, $M = 100$, and $K = 3$ in Fig. 4a. The results are generally consistent with our previous observations. Besides, the performance is improved as compared with that $S = 1$ in Fig. 3. This is clearly because the total number of files that can be cached in a cluster increases. We also note that, in line with results from the literature, the optimum cluster size shrinks as more files can be cached per users. In Fig. 4b, we consider $S = 5$, $M = 1000$, and $K = 6$ and obtain the same observations as those in all previous figures. This indicates that our replacement designs are effective while considering a more practical library size. Due to page limitation and for simplicity, we do not show here that the same observations and improvements are likewise obtained in networks with other parameters, e.g., $M = 200$ and $T_{\text{stage}} = 30$.

Finally, we demonstrate the effects of violating the conditions provided at the end of Sec. II. In Fig. 5, we consider $S = 1$, $K = 3$, and $M = 100$ and evaluate the MyDPP design in networks with different average network velocities, i.e., $V_{\text{net}} = 1, 5, 13, 21$ m/s. We observe that the performance gain of the MyDPP design gradually decreases as V_{net} increases, yet it still outperforms the static policies



(a) $S = 5$, $M = 100$, and $K = 3$.



(b) $S = 5$, $M = 1000$, and $K = 6$.

FIGURE 4. Throughput as a function of cluster size under the different replacement schemes in networks including mobility outage and with caching of multiple files per user.

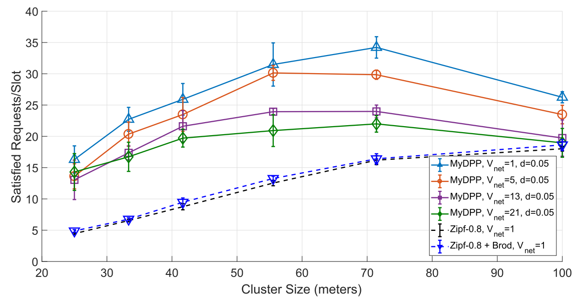


FIGURE 5. Throughput as a function of cluster size for MyDPP replacement with different average network velocities.

even at high mobility conditions, e.g., $V_{\text{net}} = 21$ m/s (75.6 km/h). This result demonstrates that the performance gain of a replacement design gradually decreases as the conditions are violated. However, even if the conditions are violated, the proposed replacement still gives more benefits than the static policies.

IX. CONCLUSION

In this paper, we investigated dynamic caching content replacement in BS-assisted wireless D2D caching networks as a response to the issue of time-varying dynamics of networks, e.g., time-varying popularity distribution and mobility of users. Our goal is to refresh the caching content in users such that it can match the demand of the network. We proposed a network architecture for caching content replacement by exploiting the broadcasting nature of the BS and by using a queueing system to track the history record. We formulated the replacement problem as a sequential decision-making problem that maximizes the service rate while being subject to the cost constraint and queue stability. By combining the concept of rewards-to-go and the drift-plus-penalty methodology, a solution framework was proposed. Two algorithms that approximate the solution were proposed: the first algorithm used only the historical record, whereas the second used both historical record and near-future information. We showed, both analytically and empirically, that our proposed designs can significantly improve the performance while still satisfying the constraints. We also observed that dynamic caching content replacement is necessary to realize the potential performance gain of D2D caching when dynamics exist.

APPENDIXES

APPENDIX A

PROOF OF LEMMA 1

Observe that

$$\sum_{t=0}^{T-1} \tilde{R}(t, A(t)) = \sum_{t=0}^{T-1} \frac{1}{l} \left(s_m^{A(t)}(t) + E\left[\sum_{\tau=1}^{l-1} s_m^{A(t+\tau)}(t+\tau) \right] \right) \tag{20}$$

Suppose that actions $A(t), \forall t$, are determined by policy P. By taking the expectations on both sides of (20) and then divided by T, we can obtain:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1} E\left[\tilde{R}(t, A(t))\right] \\
&= \frac{1}{T}\sum_{t=0}^{T-1}\left(\frac{1}{l}E\left[s_m^{A(t)}(t)\right] + E\left[\sum_{\tau=1}^{l-1} s_m^{A(t+\tau)}(t+\tau)\right]\right) \\
&= \frac{1}{T}\sum_{t=0}^{l-2}\frac{t+1}{l}E\left[s_m^{A(t)}(t)\right] + \frac{1}{T}\sum_{t=l-1}^{T} E\left[s_m^{A(t)}(t)\right] \\
&\quad + \frac{1}{T}\sum_{t=T+1}^{T+l-1}\frac{l+T-t}{l}E\left[s_m^{A(t)}(t)\right].
\end{aligned} \tag{21}$$

Eq. (21) then leads to:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1} E\left[s_m^{A(t)}(t)\right] - \frac{1}{T}\sum_{t=0}^{T-1} E\left[\tilde{R}(t, A(t))\right] \\
&= \frac{1}{T}\sum_{t=0}^{l-2}\frac{l-t-1}{l}E\left[s_m^{A(t)}(t)\right] - \frac{1}{T}\sum_{t=T+1}^{T+l-1}\frac{t-T}{l}E\left[s_m^{A(t)}(t)\right]
\end{aligned} \tag{22}$$

It follows that when $T \to \infty$, we obtain

$$\lim_{T\to\infty}\left(\frac{1}{T}\sum_{t=0}^{T-1} E\left[s_m^{A(t)}(t)\right] - \frac{1}{T}\sum_{t=0}^{T-1} E\left[\tilde{R}(t, A(t))\right]\right) = 0. \tag{23}$$

APPENDIX B

PROOF OF THEOREM 1

Proof of Queue Stability: Suppose M, $Z(0)$, and $Q_m(0), \forall m$, are finite numbers. Also, assume that $C > 0$,

$c^A_{\text{inst}} > 0, \forall A \in \mathcal{A}(t)$, and $r_m(t) \leq r_{\max}, \forall m, t$, are finite and bounded. Suppose that the system ultimately becomes unstable, then a sequence $\{A(t)\}$ generated by minimizing (10b) and an m such that $\lim_{t\to\infty} Q_m^{A(t)}(t) = \infty$ must exist. Thus, for some t_0, the network must not broadcast file m when $t > t_0$. However, this is impossible: as long as there is no broadcast, $Z(t)$ reduces to 0 as $t \to \infty$. When $Z(t) = 0$, however, we can choose freely to broadcast file m and empty queue m to minimize (10b). This leads to contradiction. Therefore, we conclude that $Q_m(t), \forall m, \forall t$, cannot go to infinity, and thus are upper bounded. This leads the queue stability in (4c) to be satisfied. □

Proof of Satisfaction of Cost Constraint: To prove this, we need to prove the rate stability of $Z(t)$. Recall that the decision whether to broadcast a file is determined by minimizing (10b). Assume that $Z(t)$ is infinite when $t \to \infty$. Then we know that for some t_0, $Z(t) > \frac{M\nu^2+Vy_{max}}{c^A_{\text{inst}}(t)}$ must hold when $t > t_0$, where ν and y_{max} are some positive numbers such that $s^A_m(t) \leq \nu$, $Q_m(t) \leq \nu$, and $\sum_{m=1}^M \tilde{R}_m(t, A(t)) \leq y_{max}$. Note that ν and y_{max} must exist since $Q_m(t), \forall m$, are upper bounded. This indicates that when $t > t_0$, the solution of minimizing (10b) must be $A^{\text{slt}}(t)$ since

$$-\sum_{m=1}^M Q_m(t)s_m^{A(t)}(t) - V\sum_{m=1}^M \tilde{R}_m(t, A(t)) + Z(t)c^A_{\text{inst}}(t) > 0 = Z(t)c^{A^{\text{slt}}}_{\text{inst}}. \quad (24)$$

This means, in this case, we will never choose to broadcast the file. Consequently, $Z(t)$ can never increase to infinity when $Z(0)$ is finite. This contradicts the assumption; thus, $Z(t)$ must be finite and upper bounded by $\frac{M\nu^2+Vy_{max}}{c}$ when $t \to \infty$. This indicates $Z(t)$ is rate stable according to Definition 2.2 in [40]. Then, by the Rate Stability Theorem (Theorem 2.5 in [40]), we know the cost constraint in (4b) is satisfied. □

APPENDIX C
PROOF OF THEOREM 2

Since $r_m(t) \leq r_{\max}$ and $c^{A(t)}_{\text{inst}} \leq C_{\max}$, it follows that there must exist some finite non-negative number y^* for a policy P such that (10) can be bounded as in (25), as shown at the bottom of this page. Then by summing the inequality for t from 0 to $T-1$, we can obtain

$$\sum_{t=0}^{T-1}\left[\Delta(t) - V\sum_{m=1}^M \tilde{R}_m(t, A(t))\right] \leq -TVy^* + TB + r_{\max}\sum_{t=0}^{T-1}\sum_{m=1}^M Q_m(t) + C_{\max}\sum_{t=0}^{T-1} Z(t). \quad (26)$$

After some algebraical manipulations, it follows that

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^M \tilde{R}_m(t, A(t)) \geq y^* - \frac{B}{V} - \frac{r_{\max}}{TV}\sum_{t=0}^{T-1}\sum_{m=1}^M Q_m(t) - \frac{C_{\max}}{TV}\sum_{t=0}^{T-1} Z(t) + \frac{L(T)-L(0)}{VT}. \quad (27)$$

We suppose that $\sum_{m=1}^M E[Q_m(t)] \leq \epsilon V$, and $E[Z(t)] \leq \delta V$. By letting $T \to \infty$ and taking the expectation, we then obtain

$$\liminf_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E\left[\sum_{m=1}^M \tilde{R}_m(t, A(t))\right] \geq y^* - \frac{B}{V} - \epsilon r_{\max} - \delta C_{\max}. \quad (28)$$

Finally, according to (25), we can understand that the minimization of (10b) can maximize y^*.

APPENDIX D
PROOF OF THEOREM 3

Consider the drift-plus-penalty minimization policy P, in which $A(t)$ is determined by P. Using (10) and summing from $t = 0$ to $t = T-1$, we can obtain

$$\begin{aligned} L(T) - L(0) - V\sum_{t=0}^{T-1}\sum_{m=1}^M \tilde{R}_m(t, A(t)) \leq & \sum_{t=0}^{T-1}\sum_{m=1}^M Q_m^{A(t)}(t)(r_m(t) - s_m^{A(t)}) - V\sum_{t=0}^{T-1}\sum_{m=1}^M \tilde{R}_m(t, A(t)) \\ & + \sum_{t=0}^{T-1} Z^{A(t)}(t)\left(c^{A(t)}_{\text{inst}}(t) - C\right) + \sum_{t=0}^{T-1} B \end{aligned} \quad (29)$$

$$\begin{aligned} \Delta(t) - V\sum_{m=1}^M \tilde{R}_m(t, A(t)) &\leq \sum_{m=1}^M Q_m(t)(r_m(t) - s_m^{A(t)}(t)) - V\sum_{m=1}^M \tilde{R}_m(t, A(t)) + Z(t)(c^{A(t)}_{\text{inst}}(t) - C) + B \\ &\leq \sum_{m=1}^M Q_m(t)r_m(t) - V\sum_{m=1}^M \tilde{R}_m(t, A(t)) + Z(t)c^{A(t)}_{\text{inst}}(t) + B \\ &\leq r_{\max}\sum_{m=1}^M Q_m(t) - Vy^* + C_{\max}Z(t) + B. \end{aligned} \quad (25)$$

Since $A(t)$ minimizes (10b), we obtain

$$\begin{aligned}&\sum_{t=0}^{T-1}\sum_{m=1}^{M} Q_m^{A(t)}(t)(r_m(t) - s_m^{A(t)}) - V\sum_{t=0}^{T-1}\sum_{m=1}^{M}\tilde{R}_m(t, A(t))\\ &\quad + \sum_{t=0}^{T-1} Z^{A(t)}(t)\left(c_{\text{inst}}^{A(t)}(t) - C\right) + \sum_{t=0}^{T-1} B\\ &\leq \sum_{t=0}^{T-1}\sum_{m=1}^{M} Q_m^{A(t)}(t)(r_m(t) - s_m^{\Theta}) - V\sum_{t=0}^{T-1}\sum_{m=1}^{M}\tilde{R}_m(t, \Theta)\\ &\quad + \sum_{t=0}^{T-1} Z^{A(t)}(t)\left(c_{\text{inst}}^{\Theta}(t) - C\right) + \sum_{t=0}^{T-1} B. \qquad (30)\end{aligned}$$

Then by taking expectations on (29) and using (30) and the assumption in (12), we can obtain

$$\begin{aligned}&L(T) - L(0) - V\sum_{t=0}^{T-1}\sum_{m=1}^{M} E\left[\tilde{R}_m(t, A(t))\right]\\ &\leq \sum_{t=0}^{T-1}\sum_{m=1}^{M} E\left[Q_m^{A(t)}(t)\right]\delta - V\sum_{t=0}^{T-1}\sum_{m=1}^{M} E\left[\tilde{R}_m(t, \Theta)\right]\\ &\quad + \sum_{t=0}^{T-1} E\left[Z^{A(t)}(t)\right]\delta + \sum_{t=0}^{T-1} B\\ &\overset{(a)}{\leq} \sum_{t=0}^{T-1}\sum_{m=1}^{M} E\left[Q_m^{A(t)}(t)\right]\delta - V\sum_{t=0}^{T-1}\sum_{m=1}^{M} y_m^{\Theta}\\ &\quad + \sum_{t=0}^{T-1} E\left[Z^{A(t)}(t)\right]\delta + \sum_{t=0}^{T-1} B, \qquad (31)\end{aligned}$$

where (a) is because policy Θ is i.i.d.. Since δ can be arbitrarily close to zero, it follows that

$$\begin{aligned}&\liminf_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{M} E\left[\tilde{R}_m(t, A(t))\right]\\ &\geq \lim_{T\to\infty}\frac{1}{VT}\sum_{t=0}^{T-1}\sum_{m=1}^{M} V y_m^{\Theta} - \frac{B}{V} + \frac{L(T) - L(0)}{VT}\\ &\geq \sum_{m=1}^{M} y_m^{\Theta} - \frac{B}{V}.\end{aligned}$$

ACKNOWLEDGMENT

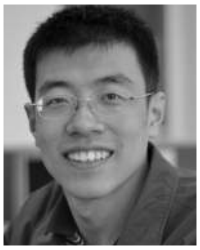
Hao Feng was at the Department of Electrical and Computer Engineering, University of Southern California (USC), Los Angeles, CA 90089, USA. This article was presented in part at the IEEE International Conference on Communications, May 2019.

REFERENCES

[1] M.-C. Lee, H. Feng, and A. F. Molisch, "Design of caching content replacement in base station assisted wireless D2D caching networks," in *Proc. IEEE ICC*, May 2019, pp. 1–7.
[2] "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022," San Jose, CA, USA, White Paper.
[3] 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 Commun. Mag.*, vol. 51, no. 4, pp. 142–149, Apr. 2013.
[4] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: Design aspects, challenges, and future directions," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 22–28, Sep. 2016.
[5] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. De Silva, F. Tufvesson, A. Benjebbour, and G. Wunder, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.
[6] G. Tyson, N. Sastry, R. Cuevas, I. Rimac, and A. Mauthe, "A survey of mobility in information-centric networks," *Commun. ACM*, vol. 56, no. 12, pp. 90–98, Dec. 2013.
[7] K. Doppler, M. Rinne, C. Wijting, C. Ribeiro, and K. Hugl, "Device-to-device communication as an underlay to LTE-advanced networks," *IEEE Commun. Mag.*, vol. 47, no. 12, pp. 42–49, Dec. 2009.
[8] M. Ji, G. Caire, and A. F. Molisch, "The throughput-outage tradeoff of wireless one-hop caching networks," *IEEE Trans. Inf. Theory*, vol. 61, no. 12, pp. 6833–6859, Dec. 2015.
[9] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 176–189, Jan. 2016.
[10] M.-C. Lee, M. Ji, A. F. Molisch, and N. Sastry, "Throughput–outage analysis and evaluation of cache-aided D2D networks with measured popularity distributions," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5316–5332, Nov. 2019.
[11] N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Wireless video content delivery through distributed caching and peer-to-peer gossiping," in *Proc. Conf. Rec. 45th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2011, pp. 1177–1180.
[12] S. H. Chae and W. Choi, "Caching placement in stochastic wireless caching helper networks: Channel selection diversity via caching," *IEEE Trans. Wireless Commun.*, vol. 15, no. 10, pp. 6626–6637, Oct. 2016.
[13] K. Li, C. Yang, Z. Chen, and M. Tao, "Optimization and analysis of probabilistic caching in N-tier heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1283–1297, Feb. 2018.
[14] 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.
[15] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[16] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1146–1158, Feb. 2017.
[17] D. Malak, M. Shalash, and J. G. Andrews, "Optimizing content caching to maximize the density of successful receptions in device-to-device networking," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4365–4380, Oct. 2016.
[18] B. Chen, C. Yang, and 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.
[19] Z. Chen, N. Pappas, and M. Kountouris, "Probabilistic caching in wireless D2D networks: Cache hit optimal versus throughput optimal," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 584–587, Mar. 2017.
[20] 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 tradeoff," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7500–7514, Nov. 2018.
[21] B. Chen, C. Yang, and A. F. Molisch, "Cache-enabled device-to-device communications: Offloading gain and energy cost," *IEEE Trans. Wireless Commun.*, vol. 16, no. 7, pp. 4519–4536, Jul. 2017.
[22] 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.
[23] N. Pappas, Z. Chen, and I. Dimitriou, "Throughput and delay analysis of wireless caching helper systems with random availability," *IEEE Access*, vol. 6, pp. 9667–9678, Feb. 2018.
[24] M. Naslcheraghi, M. Afshang, and H. S. Dhillon, "Modeling and performance analysis of full-duplex communications in cache-enabled D2D networks," in *Proc. IEEE ICC*, May 2018, pp. 1–6.
[25] 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.

[26] J. Song, M. Sheng, X. Wang, and C. Xu, "Content caching and sharing in D2D Networks based on content topology," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.
[27] M. Choi, J. Kim, and J. Moon, "Wireless video caching and dynamic streaming under differentiated quality requirements," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1245–1257, Jun. 2018.
[28] J. Gu, W. Wang, A. Huang, H. Shan, and Z. Zhang, "Distributed cache replacement for caching-enable base stations in cellular networks," in *Proc. IEEE ICC*, May 2014, pp. 2648–2653.
[29] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless networks with elastic and inelastic traffic," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 864–874, Jun. 2014.
[30] P. Blasco and D. Gündüz, "Content-level selective offloading in heterogeneous networks: multi-armed bandit optimization and regret bounds," 2014, *arXiv:1407.6154*. [Online]. Available: http://arxiv.org/abs/1407.6154
[31] S. Muller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.
[32] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb. 2018.
[33] B. N. Bharath, K. G. Nagananda, D. Gündüz, and H. V. Poor, "Caching with time-varying popularity profiles: A learning-theoretic perspective," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3837–3847, Sep. 2018.
[34] N. Carlsson and D. Eager, "Ephemeral content popularity at the edge and implications for on-demand caching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1621–1634, Jun. 2017.
[35] P. T. Joy and K. P. Jacob, "A comparative study of cache replacement policies in wireless mobile networks," in *Advances in Computing and Information Technology*. Berlin, Germany: Springer, 2012, pp. 609–619.
[36] M. Veeresha and M. Sugumaran, "Optimal hybrid broadcast scheduling and adaptive cooperative caching for spatial queries in road networks," *J. Ambient Intell. Humanized Comput.*, vol. 8, no. 4, pp. 607–624, Aug. 2017.
[37] M. Gregori, J. Gomez-Vilardebo, J. Matamoros, and D. Gunduz, "Wireless content caching for small cell and D2D networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1222–1234, May 2016.
[38] Z. Chen, H. Mohammed, and W. Chen, "Proactive caching for energy-efficiency in wireless networks: A Markov decision process approach," in *Proc. IEEE ICC*, May 2018, pp. 1–6.
[39] D. Wu, L. Zhou, Y. Cai, and Y. Qian, "Collaborative caching and matching for d2d content sharing," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 43–49, Jun. 2018.
[40] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.
[41] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, "An asymptotically efficient simulation-based algorithm for finite horizon stochastic dynamic programming," *IEEE Trans. Autom. Control*, vol. 52, no. 1, pp. 89–94, Jan. 2007.
[42] T. Homem-de-Mello and G. Bayraksan, "Monte Carlo sampling-based methods for stochastic optimization," *Surv. Oper. Res. Manage. Sci.*, vol. 19, no. 1, pp. 56–85, Jan. 2014.
[43] M.-C. Lee and A. F. Molisch, "Individual preference aware caching policy design for energy-efficient wireless D2D communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–7.
[44] M.-C. Lee, A. F. Molisch, N. Sastry, and A. Raman, "Individual preference probability modeling and parameterization for video content in wireless caching networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 676–690, Apr. 2019.
[45] D. Bertsekas, *Dynamic Programming and Optimal Control*, vols. 1–2, 4th ed. Belmont, MA, USA: Athena Scientific, 2012.
[46] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 2, no. 3, pp. 257–269, Jul. 2003.

MING-CHUN LEE (Student Member, IEEE) received the B.S. and M.S. degrees in electrical and computer engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Southern California. From 2014 to 2016, he was a Research Assistant with the Wireless Communications Lab, Research Center for Information Technology Innovation, Academia Sinica, Taiwan. His research interest includes signal processing, design, modeling, and analysis in wireless systems and networks. He is especially working on topics relevant to wireless caching networks in recent years.



HAO FENG (Member, IEEE) received the B.S. and M.S. degrees from the Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China, in 2006 and 2008, respectively, and the Ph.D. degree from the Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA, in 2017. He joined Intel Labs, Intel Corporation, Hillsboro, OR, USA, in 2018, where he is currently a Research Scientist. His research interests include stochastic network optimization with their applications in cloud and computing networks, caching networks, wireless D2D and ad hoc networks, and information centric networks. He received the Best Paper Award from the IEEE ICC 2016 Conference, and the Annenberg Graduate Fellowship, from 2009 to 2013, for his Ph.D. study.

ANDREAS F. MOLISCH (Fellow, IEEE) received the Dipl. Ing., Ph.D., and habilitation degrees from the Technical University of Vienna, Vienna, Austria, in 1990, 1994, and 1999, respectively.

He subsequently was with FTW, Austria, AT&T Bell Laboratories Research, USA, Lund University, Lund, Sweden, and Mitsubishi Electric Research Labs, USA. He is currently a Professor and the Solomon Golomb–Andrew and Erna Viterbi Chair at the University of Southern California, Los Angeles, CA, USA. His current research interests are measurement and modeling of mobile radio channels, multiantenna systems, wireless video distribution, ultrawideband communications and localization, and novel modulation formats. He has authored, coauthored, or edited four books, among them *Wireless Communications* (Wiley-IEEE Press), 19 book chapters, more than 250 journal papers, more than 350 conference papers, as well as holds more than 80 patents, and 70 standards contributions.

Dr. Molisch has been an Editor of a number of journals and special issues, the General Chair, the Technical Program Committee Chair, or the Symposium Chair of multiple international conferences, as well as the Chairman of various international standardization groups. He is a Fellow of National Academy of Inventors, a Fellow of AAAS, Fellow of IET, an IEEE Distinguished Lecturer, and a member of Austrian Academy of Sciences. He has received numerous awards, such as the Donald Fink Prize from the IEEE, the IET Achievement Medal, the Armstrong Achievement Award from the IEEE Communications Society, and the Eric Sumner Award from the IEEE.