Collaborative Content Placement Among Wireless Edge Caching Stations With Time-to-Live Cache

Lixing Chen, Linqi Song, *Member, IEEE*, Jacob Chakareski, *Senior Member, IEEE*, and Jie Xu, *Member, IEEE*

***Abstract*—Content caching at the Internet edge using a network of wireless edge caching stations (ECSs) is recently considered as a key solution to alleviating the backhaul traffic burden and improving the quality of experience in 5G networks. This paper studies wireless edge caching systems with the following features: first, content files can be partitioned into many coded packets, which then can be cached in multiple ECSs for collaborative content delivery; second, the service provider (SP) deploys time-to-live cache at ECSs and each cached content file has an occupancy time that needs to be guaranteed; third, the content-to-be-cached arrives at the caching system following a stochastic process as users request new content over time. Unlike existing works that determine which content to cache, this paper focuses on how to distribute the coded packets of content-to-be-cached among the network of ECSs in order to reduce the content downloading time. A novel content placement strategy, called stochastic collaborative content placement is proposed based on Lyapunov techniques. The proposed algorithm makes content placement decisions using only currently available information without foreseeing future content arrivals, takes advantage of the spatial content popularity variation with coded caching, and achieves the provable close-to-optimal long-term caching performance. Simulations are carried out on a real-world YouTube video request trace and the results demonstrate a tremendous caching performance improvement against a variety of benchmark schemes.**

***Index Terms*—Content placement, coded caching, wireless network, online decision-making.**

I. Introduction

DUE to the prevalence of smart mobile devices with advanced multimedia capabilities and the trend towards high data rate applications, wireless networks have been experiencing a tremendous increase in data traffic, especially multimedia data traffic, in recent years. To keep up with the massive growth of mobile data demand, there is a growing interest in *caching at the edge* of the Internet. Wireless Edge Caching Stations (ECSs) (e.g. caching-enabled small cell base stations or wireless infostations [1], [2]) are being deployed as a supplement to the existing cellular architecture and content delivery networks (CDNs) for caching popular content (e.g. videos) in close proximity to end users, thereby reducing the content downloading time and alleviating the backhaul traffic burden. Wireless environment is very different from wired networks due to its broadcasting nature and limited transmission range. Thanks to the recently developed *coded caching* approach [3] and network densification in 5G [4], collaborative caching among a network of ECSs further improves the caching performance: content files can be partitioned into many coded packets, which then are cached in multiple collaborative ECSs. In this way, ECSs collaborate in terms of not only sharing cache spaces but also the physical transmission for content delivery.

Network service providers (SPs) offer caching-as-a-service to content providers (CPs). Recent studies [5] have shown that without explicit monetary compensation or incentive structures, such CDNs do not operate effectively. However, standard cache management policies such as Least Frequently Used (LFU) and Least Recently Used (LRU) make it difficult for SPs to design individualized contracts and for CPs to account for their valuation of their content when contracting with SPs because these policies treat different content in a strongly coupled manner. This recently motivates the design of Time-to-Live (TTL) cache management policies, which aim to ensure the *occupancy time* of content and decouple the dependency of different content. Specifically, the CP can negotiate with the SP to determine the price and occupancy time of a content file when a caching request is issued. The SP then will ensure that this content file will be cached for at least a certain amount of time [6], [7].

In this paper, we investigate content placement among a network of wireless ECSs under a TTL cache management policy. Different from many existing works that consider *which content* to cache, we focus on *how to place the content-to-be-cached among the ECS network* to maximize the caching performance, taking into account the content occupancy time requirements and the limited storage/cache space in ECSs. The content occupancy time requirements in TTL cache make the collaborative content placement problem significantly different and more difficult: once a content file is chosen to be cached, it has to stay in the cache for at least the negotiated occupancy time, which may reduce the chance of caching a more popular content file

Manuscript received August 23, 2018; revised February 4, 2019; accepted June 25, 2019. Date of publication July 16, 2019; date of current version January 24, 2020. The work of L. Chen and J. Xu was supported in part by the U.S. Army Research Office under Grant W911NF1810343. The work of L. Song was supported by the City University of Hong Kong under Grant 7200594. The work of J. Chakareski was supported in part by the National Science Foundation (NSF) under Award CCF-1528030, Award ECCS-1711592, Award CNS-1836909, and Award CNS-1821875, and in part by research gifts and an Adobe Data Science Award from Adobe Systems. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Christian Timmerer. *(Corresponding authors: Lixing Chen; Linqi Song; Jie Xu.)*

L. Chen and J. Xu are with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146 USA (e-mail: lx.chen@miami.edu; jiexu@miami.edu).

L. Song is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: linqi.song@cityu.edu.hk).

J. Chakareski is with the Ying Wu College of Computing, New Jersey Institute of Technology, Newark, NJ 07103 (e-mail: jacob@ua.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2019.2929004

forthcoming in the future. This is very different from existing cache management policies where a content file can be replaced by a new content file at any time. The fact that new content files, which vary considerably in size and popularity, arrive at the network (i.e. are requested by users for the first time) sequentially in realistic scenarios makes collaborative content placement particularly challenging, as any content placement decisions must be made without foreseeing the size and popularity of new content files forthcoming in the future. This thus calls for an efficient online algorithm that makes content placement decisions on-the-fly. The main contributions of this paper are summarized as follows:

1) We propose a stochastic collaborative content placement (SCCP) framework. SCCP adopts the coded caching approach which allows ECSs to cache a portion of coded packets of a content file depending on the content popularity among their serving regions as well as their spare caching spaces. SCCP aims to optimize the long-term caching performance, explicitly taking into account the fact that new content-to-be-cached arrives to the system according to a stochastic process and the content occupancy time requirements due to TTL cache.

2) The proposed content placement algorithm performs in an online fashion without requiring the future information of forthcoming new content files by leveraging and extending the *perturbed Lyapunov optimization* [8] technique. We prove that the proposed algorithm achieves within a bounded deviation from the optimal caching performance (in terms of minimizing content downloading time) that can be achieved by an oracle algorithm that knows the complete future information.

3) As an integral part of the SCCP framework, a content coding strategy for distributed ECSs, namely *spatial coded packets reuse*, is designed to minimize the coding overhead. The key idea is to allow ECSs sufficiently apart from each other to cache the same coded packets, thereby reducing the number of required distinct coded packets. The proposed coding strategy is proved to achieve optimal coding efficiency.

4) Simulations on real-world YouTube video request traces are carried out to validate our analytical results and evaluate the performance of the proposed algorithms. The results confirm that our method significantly improves caching performance in terms of downloading time against state-of-the-art benchmark algorithms.

The rest of this paper is organized as follows. Section II discusses related work. Section III presents the system model and formulates the stochastic collaborative content placement problem. Section IV develops a novel content placement algorithm based on the perturbed Lyapunov technique and proves its performance guarantee. Section V designs a coding strategy to minimize the coding overhead. Simulations on real-world data are carried out in Section VI, followed by the conclusion in Section VII.

II. Related Work

Recent effort has been made on caching policy design in wireless CDNs to improve quality of experience for mobile users. Authors in [9], [10] introduce the concept of *FemtoCaching*/*EdgeCaching* and study content placement at *cache helpers* at the network edge to minimize the content downloading delay, utilizing coded caching. Geographical caching is investigated in [11] to maximize the probability of serving a user using the stochastic geometry model. The idea of using caching to support mobility is investigated in [12]. Authors in [13] study context-aware resource allocation for energy-efficient caching and [14] designs a caching policy for base stations with energy harvesting. A common simplification for content caching assumes that the content popularity follows the Zipf distribution [13]. More complicated models employ user-specific context and social information for popularity prediction [15]. However, most of these works do not consider collaboration among multiple ECSs.

Due to the trend of the network densification in 5G, collaborative caching among multiple ECSs is gaining increasing attention. For example, [16] considers a hierarchical structure where the core caching network and a set of (possibly) interconnected base stations work collaboratively to cache content. Authors in [17] consider the energy consumption of Small-cell Base Stations (SBSs) when performing the collaborative caching [18]. Investigates the collaborative caching for multiple CPs such that the payment of a CP for caching a number of content files is minimized. However, these works only consider caching schemes under static offline settings where the content-to-be-cached is *already-known*. Although authors of [18] also propose an online algorithm for collaborative caching under unlimited cache space, it is very different from our work since we consider the limited cache capacities of ECSs, that new content files are requested by users in a stochastic manner and that the system employs TTL caches.

Many recent works also consider dynamic cache-enabled systems. For instance, [19] proposes an online proactive caching strategy concerning the uncertainty of user activities. [20] jointly optimizes content placement and request redirection with dynamic user request distribution. [21] designs a multicast scheduling policy based on Markov Decision Process (MDP). [22] designs a caching strategy based on Lyapunov criterion to stabilize the service request queues thereby ensuring finite delay. However, these works focus on the stochasticity of users' content requests. In contrast, we focus on the stochastic content arrival process and decision making, i.e. the content placement decision is made when a new content file is to be cached under the TTL cache management policy. Because the arrival of new content-to-be-cached follows a stochastic process, the content placement problem is a stochastic optimization problem where information about future content-to-be-cached is unknown at the current decision time. The Lyapunov technique is a widely used stochastic optimization technique. Originally proposed for establishing control system stability, it was later extended to achieve long-term queuing stability in networks [8], with a salient feature that it does not require future information when making control decisions. However, standard Lyapunov technique cannot handle the causality constraint induced by the cache dynamics and the occupancy time requirements. In this paper, we develop our content placement algorithm based on the *perturbed* Lyapunov optimization technique [23], which was applied to address similar causality challenges in other applications [24], [25]. We introduce the concept of "free cache space", a type of virtual queue, and derive application-specific performance bounds.

TABLE I
COMPARISON WITH EXISTING WORKS

	[9] [27]	[29]	[16], [17]	[18]	[21], [22]	[19]	[5], [30]	This paper
Collaboration	Yes	No	Yes	Yes	Yes	No	No	Yes
Online	No	No	No	Yes	Yes	Yes	Yes	Yes
Stochasticity	No	No	No	No	User request	User request	-	Content arrival
Limited capacity	Yes	Yes	Yes	No	Yes	No	Yes	Yes

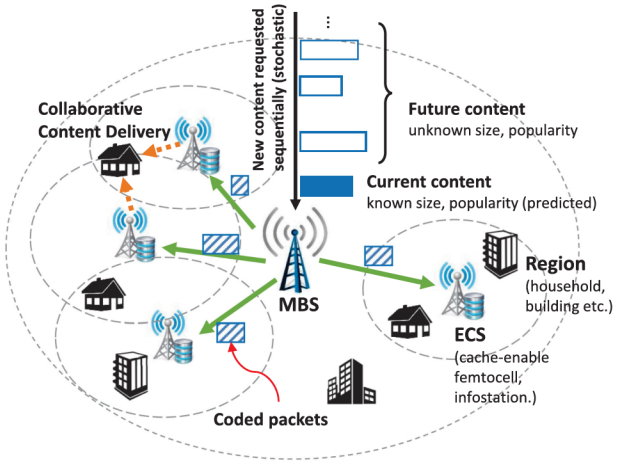


Fig. 1. Stochastic collaborative caching in heterogeneous small cell networks.

Coded caching [3] is a fundamental technique that enables collaboration among multiple ECSs. It has been shown that coded caching can effectively enhance caching efficiency by partitioning files into packets and caching them in collaborative ECSs. The authors in [26] combine the coded/uncoded caching strategy in disjoint cluster-centric small cell networks. In [27], [28], the maximum distance separable (MDS) coded caching is considered where the optimal MDS-coded cache placement problems are formulated to minimize the downloading time and the backhaul rate, respectively. Our paper does not develop new codes for collaborative caching. Instead, we use existing coding methods to enable ECS collaboration and focus on developing online content placement schemes for stochastic content arrivals. Table I summarizes the differences of the proposed scheme from the existing works.

III. System Model

A. Network Model

We consider a heterogeneous wireless network (see Fig. 1 for illustration) consisting of one macro base station (MBS) and N wireless Edge Caching Stations (ECSs), indexed by $\mathcal{N} = \{1, 2, \ldots, N\}$. The MBS provides coverage with a large cell radius. The ECSs are densely deployed in hotspots for caching capacity enhancement (e.g. SBSs). We consider that the network is divided into M disjoint regions (or hotspots), indexed by $\mathcal{M} = \{1, 2, \ldots, M\}$. Due to the dense deployment of ECSs, users in a region are in the transmission range of (possibly) multiple ECSs. Let $\mathcal{B}_m \in \mathcal{N}$ be the set of ECSs that can serve region m and $\mathcal{M}_n$ be the set of regions that are served by ECS n. By making each region small, we consider the expected transmission rate between ECS n and region m, denoted by $r_{m,n}$. We further assume that the inter-ECS interference is handled by state-of-the-art interference mitigation solutions, e.g. successive interference cancellation [31]. Depending on the locations of the ECSs and regions, the expected transmission rate differs across ECS-region pairs. Specifically, the expected downlink transmission rate between ECS n and region m is

$$r_{m,n} = \mathbb{E}_{h_{m,n}} \left[W_n \log \left(1 + \frac{h_{m,n} P_n}{N_0 + I} \right) \right], \tag{1}$$

where W_n is the channel bandwidth of ECS n, $h_{m,n}$ is the channel gain, P_n is the transmission power of ECS n, N_0 is the noise level, and I is the interference from other ECSs. At the content placement decision time, the channel condition under which users will download the content is not realized. Therefore, we consider the expected rate rather than instantaneous rate.

If the content file requested by a user is cached at the nearby ECSs, then the user can access the content quickly, bearing only the ECS-to-user transmission delay. Otherwise, the user has to retrieve the content by turning to the cache entities at the MBS (if cached at the MBS) or the content provider (if not cached at the MBS), which incurs the larger transmission delay. Let $r_{m,0}$ denote the data rate for content retrieval via the MBS, which is assumed to be lower than $r_{m,n}$. This is because the MBS often can only provide limited radio resources to each user and hence, the MBS transmission rate is usually lower than the ECS transmission rate [32].

B. Content Arrival and Popularity Model

From an operational perspective, when a user requests some content, if the content is not in the cache, then the SP needs to fetch the content from the CP and forward it to the user. Therefore, this is a natural time for the SP to negotiate contracts with the CP and determine caching decisions for future requests. This leads to an event-driven system in which content caching and placement decisions have to be made sequentially over time. In this paper, we consider only content files that the SP decides to cache in MBS/ECSs according to some contract-based caching mechanism and focus on *how to place the content packets among the MBS/ECSs*. Hence, we are not deciding *whether or not* to cache content but rather where to place the content-to-be-cached, which is substantially different from many existing works. To better see the link between content placement and traditional caching problems, consider scenarios with and without ECSs. In the scenario without ECSs, caching occurs only on the MBS and existing TTL-based caching policies can be directly applied. In the scenario with ECSs, caching can occur on both the MBS and ECSs, which play as caching helpers to move cached content even closer to the end users. The decision of whether or not

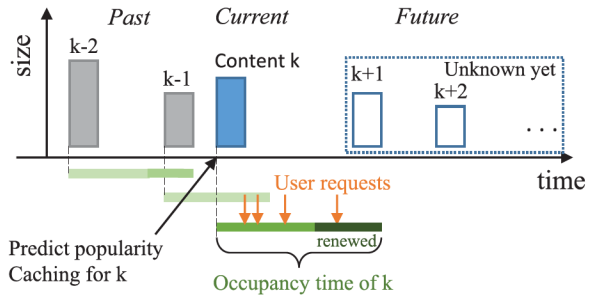


Fig. 2. Illustration of content arrival, occupancy time and popularity.

to cache is the same as before and hence the same TTL-based caching policy can be used. The difference is that where to place the content is not a simple decision of putting the content in the MBS, but rather involves a more complex decision of distributing content packets among the ECSs given their capacity and location constraints.

Let $k = 0, 1, 2, \ldots$ be the sequence of new content files that the caching policy decides to cache over time. The size of content file k is denoted by $s^k \in (0, s_{\max}]$ (measured in packets of the same size), which differ across content files. Each content file to be cached has an occupancy time requirement as a result of some contract-based mechanism determined by the CP and the SP. The occupancy time does not need to be a predetermined constant. It can also be variable (renewed/prolonged) as in [6]. When the content has been cached by the SP for the corresponding occupancy time, the content expires so the SP is allowed to remove it from the caches of the MBS/ECSs, thereby making room for new content. Let g_n^k denote the number of expired packets in the cache of ECS n when a new content file k arrives. We assume that each ECS n has a limited cache capacity, denoted by C_n (measured in packets), and the MBS has a sufficiently large cache capacity to support all content files that are to be cached. Note that the MBS does not cache the whole content catalog; it only needs to cache content that the TTL caching policy decides to cache. However, due to the limited ECS cache capacity and the required content occupancy time, the system has to judiciously decide how to place each content file in the ECS network based on the available cache spaces, the content size as well as the popularity of the content among users in order to minimize the expected downloading time.

To facilitate the content placement decisions among the ECSs, the region-wise popularity of each content file k will have to be predicted. Here, we assume that an accurate enough prediction algorithm [15], [33] is used and do not develop new prediction algorithms. In this paper, the popularity of content file k is described by $(p_1^k, \ldots, p_M^k)$ where $p_m^k \in [0, p_{\max}]$ is the probability that content k will be requested by users in region m within its occupancy time. However, we note that the popularity and the size of forthcoming content files (i.e. content $k+1$, $k+2, \ldots$) are not known and cannot be predicted by the system. The fact that new content files become available sequentially makes the problem very challenging since *content placement decisions must be made without foreseeing forthcoming new content files*.

Fig. 2 illustrates the content arrival process. When content k arrives to the system (namely it is requested by some user in the cell for the first time and the caching policy decides to cache it), its region-wise popularity is predicted. At this point, the system does not know what content file $k+1, k+2, \ldots$ are, when they will arrive or what their sizes are and hence also cannot predict their popularity.

C. Coded Caching and Content Downloading Model

While many codes can be used for collaborative content placement, we use Maximum Distance Separable (MDS) code as a specific example, which is also widely used in the literature [17], [34]. Specifically, a content file k is divided into s^k small packets and then $Q^k \geq s^k$ coded packets are generated using the MDS code at the MBS. Then the coded packets will be distributed among the network of ECSs. The MDS code allows a user to recover the original content file by downloading any s^k distinct coded packets. In general, we would like the coding overhead, defined as Q^k/s^k, to be small in order to reduce the coding complexity. In the following sections, we first determine how many coded packets each ECS should cache for a particular content file. In Section V, we design a coding scheme to minimize the coding overhead.

Let $x_n^k \leq s^k$ denote the number of coded packets cached in ECS n for content file k. We let $x_0^k = s^k$ so that the users can always retrieve the whole content from the MBS. A user may have to access multiple ECSs to acquire a total of s^k distinct packets for content recovery. We assume that the user device is equipped with a single radio and hence, it cannot download from multiple ECSs at the same time. To examine the total downloading time for a typical user in region m when it requests file k, let $a_{m,n}^k \leq x_n^k$ denote the number of coded packets downloaded from ECS n by the user. Then, the user can recover the original content k by following the MDS decoding rule if the total number of downloaded coded packets is greater than s^k, i.e.

$$a_{m,0}^k + \sum_{n \in \mathcal{B}_m} a_{m,n}^k \geq s^k \tag{2}$$

where $a_{m,0}^k$ denotes the number of coded packets received from the MBS. Depending on the transmission rates between ECSs and region m, the total downloading time of content k for a user in region m is thus

$$\begin{aligned} d(\boldsymbol{a}_m^k) &= \sum_{n \in \mathcal{B}_m} \frac{a_{m,n}^k}{r_{m,n}} + \frac{s^k - \sum_{n \in \mathcal{B}_m} a_{m,n}^k}{r_{m.0}} \\ &= \frac{s^k}{r_{m,0}} - \sum_{n \in \mathcal{B}_m} \left(\frac{1}{r_{m,0}} - \frac{1}{r_{m,n}} \right) a_{m,n}^k \end{aligned} \tag{3}$$

The optimal downloading strategy that minimizes the downloading time by user in region m given the caching strategy $\boldsymbol{x}^k = (x_1^k, \ldots, x_N^k)$ can therefore be obtained by solving the following linear program:

$$\min_{\boldsymbol{a}_m^k} \quad d(\boldsymbol{a}_m^k) \tag{4a}$$

$$\text{s.t.} \quad a_{m,n}^k \leq x_n^k, \qquad \forall m, n \tag{4b}$$

$$\sum_{n \in \mathcal{B}_m} a_{m,n}^k \leq s^k, \qquad \forall m, n \tag{4c}$$

The solution to the above linear program is intuitive: the user ranks its reachable ECSs according to the transmission

rate in the descending order. Then, it downloads as many coded packets as possible following this ranking until s^k distinct coded packets have been downloaded. If the number of coded packets cached at all reachable ECSs is smaller than s^k, then the user turns to the MBS to download the remaining coded packets. We denote $d_m(\boldsymbol{x}^k)$ as its optimal downloading time of content file k for users in region m achieved by the optimal association/downloading strategy given the content placement strategy $\boldsymbol{x}^k$. Since content popularity differs across regions, the expected downloading time of all users for a given collaborative content placement strategy $\boldsymbol{x}^k$ is $d^k(\boldsymbol{x}^k) = \sum_m p_m^k d_m(\boldsymbol{x}^k)$. In this paper, we use the content downloading time as the performance metric.

D. Model Assumptions and Justifications

We summarize the assumptions made in the above model and their justifications. 1) *Inter-ECS interference is handled by state-of-the-art interference mitigation solutions*: A simple interference mitigation scheme is to assign orthogonal frequency bands to adjacent ECSs. A more advanced (but more complicated) solution can be successive interference cancellation. 2) *Data rate for content retrieval via MBS is lower than ECS*: This is a standard assumption and what motivates caching on ECSs. 3) *Each ECS has a limited cache capacity and the MBS has a sufficiently large cache capacity to support all content-to-be-cached*: It is a natural assumption that an ECS has a limited cache capacity. For the second part, note that the MBS caches only content that the TTL caching policy decides to cache. 4) *An accurate enough prediction algorithm is used to predict content popularity of the current received content*: Many existing content popularity prediction algorithms have been developed in the past recent years, e.g. [15], [33]. Our algorithm takes the output of these algorithms to make content placement decisions. Developing prediction algorithms is orthogonal to this effort. 5) *The user device is equipped with a single radio so it cannot download from multiple ECSs at the same time*: Most current commercial mobile devices have a single radio that can only communicate with one other device at a time. The mobile device may have multiple antennas to leverage multiple-input multiple-output (MIMO), but the communication is still between one transmitter and one receiver at a time.

E. Problem Formulation

As the ECSs have limited storage capacity, the number of content packets that can be cached in an ECSs is limited by the current free cache space at the placement decision time. We define c_n^k as the *free cache space* of ECS n at the decision time for content k, which evolves as old content packets expire and new content packets are pushed into the cache over time. Note that c_n^k is *not* the content size already cached in ECS n; rather, it is the space available for caching new content. To understand how c_n^k evolves over time, we illustrate the events occurred between arrivals of content k and content $k+1$.

1) Content-to-be-cached k arrives to the system. The current free cache space of ECS n is c_n^k.
2) Each ECS n pushes $x_n^k \le s^k$ packets into its local cache where we must also have $x_n^k \le c_n^k$ satisfied.

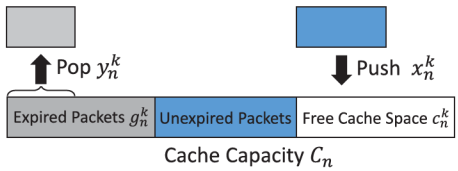


Fig. 3. Pop, push and cache space evolution.

3) Users request and download unexpired content up to k.
4) By the time when content-to-be-cached $k+1$ arrives, g_n^k packets in ECS n's cache become expired and y_n^k of them are removed from the cache of ECS n. Thus, the free cache space evolves to $c_n^{k+1} = c_n^k - x_n^k + y_n^k$. Fig. 3 illustrates the evolution of the free cache space.

For each newly arrived content k, each ECS n makes two caching decisions: the **push** decision x_n^k and the **pop** decision y_n^k. The objective of our system is to minimize the average content downloading time by judiciously making content placement decisions (push/pop) for the network of ECSs. Formally, the stochastic collaborative content placement (SCCP) problem is formulated as follows:

$$\textbf{SCCP} \quad \min_{\boldsymbol{x}^k, \boldsymbol{y}^k, \forall k} \quad \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} d^k(\boldsymbol{x}^k) \tag{5a}$$

$$\text{s.t.} \quad x_n^k \le s^k, y_n^k \le g_n^k, \forall n, \forall k \tag{5b}$$

$$x_n^k \le c_n^k, \forall n, \forall k \tag{5c}$$

$$c_n^{k+1} = c_n^k - x_n^k + y_n^k, \forall n, \forall k \tag{5d}$$

The SCCP problem aims to minimize the average content downloading time, i.e., $\frac{1}{K}\sum_{k=1}^{K} d^k(\boldsymbol{x}^k)$, for a total of K content files arriving sequentially. For the decision problem with respect to content k, the decision variables are push decision $\boldsymbol{x}^k = (x_1^k, \ldots, x_N^k)$ and pop decision $\boldsymbol{y}^k = (y_1^k, \ldots, y_N^k)$. Although the pop decision $\boldsymbol{y}^k$ is not explicitly reflected in the objective function, it indirectly affects the maximal number of packets that can be put in the ECSs through the free caching space dynamics. We use $K \to \infty$ to model a long period of time during which there are many content arrivals. Our model also captures the dynamic system where individual users can move because we consider content download between ECSs and user regions. The **SCCP** problem is a stochastic optimization problem because information about content $k+1, k+2, \ldots$ is unavailable at decision time for content k. Therefore, it cannot be solved using a conventional offline optimization approach. Let d^* be the infimum average downloading time achievable by any content placement policy that meets the required constraints, possibly by an oracle policy that have complete information of future content. We will compare the performance achieved by our algorithm with d^* in the next section. Before proceeding with the algorithm, we make several remarks on this formulation:

Causality Constraint: Constraints (5c) and (5d) impose a free cache space causality constraint on the feasible cache push actions. This constraint makes **SCCP** a particularly challenging problem to solve since the system's decisions are now intricately intertwined across different content k due to the free caching

space dynamics (5d). Even if the information of future content size and popularity were known, solving **SCCP** involves $2NK$ decision variables coupled via (5d) for a number of K content files. Therefore, solving **SCCP** can be intractable when K is large. The problem becomes even more complicated due to the fact that content file $k+1, k+2, \ldots$ are unknown by the arrival of content k, and hence their content sizes and popularity are unknown.

Content Expiration Process: The number of expired packets g_n^k in ECS n during decision cycle k is a random variable depending on content $1, \ldots, k$'s popularity changes and their associated contract renewal processes (if the TTL-based caching policy allows renewal). Given the current free cache space c_n^k, it is clear that g_n^k must be no more than $C_n - c_n^k$. To enable our subsequent analysis, we make the following assumption.

Assumption 1: For each ECS n, there exists a random process $\tilde{g}_n^k$ so that (1) $\tilde{g}_n^k \in [0, g_{max}]$ is i.i.d. over k, (2) $g_n^k = \tilde{g}_n^k$ if $\tilde{g}_n^k \leq C_n - c_n^k$, and (3) $g_n^k = C_n - c_n^k$ if $\tilde{g}_n^k > C_n - c_n^k$.

This assumption states that g_n^k is a truncated version (truncated by $C_n - c_n^k$) of an i.i.d. random variable $\tilde{g}_n^k$. We note that this assumption is required to perform rigorous performance analysis of our algorithm, but our algorithm can still run without this assumption. Moreover, we show that the i.i.d. assumption roughly holds by using real-world data traces. Simulations in Section VI-E show that g_n^k itself is already i.i.d. and hence, simply taking $\tilde{g}_n^k = g_n^k$ justifies this assumption.

Pop Action: In addition to the push decision, each ECS also has to make a pop decision which removes part of the expired packets from the cache. In practice, however, all expired packets can be removed to make more room for newly arrived content. The pop decision is only introduced and needed to facilitate the analysis. In particular, by appropriately deciding the number of packets to remove from the cache, the free cache space c_n^k can be kept bounded in our problem (which will be shown later). Therefore, the total required cache capacity can be made finite, thereby enabling practical implementation. Apparently, removing all expired packets will require a smaller cache capacity to implement the algorithm.

Objective Function: we consider content downloading time as the objective function. Nevertheless, our framework can be easily adapted to handle different objective functions. For instance, instead of the downloading time, we can assign other types of costs (e.g. energy, bandwidth or monetary cost) to downloading the content from the network edge and from the core network (i.e. wired Internet backbone and the MBS). Moreover, instead of minimizing cost, our framework can also be used to maximize user satisfaction or the SP's profit.

IV. Stochastic Collaborative Content Placement

In this section, we develop the stochastic collaborative content placement algorithm for ECSs based on the perturbed Lyapunov optimization technique. We will have to solve the **SCCP** problem involving both the spatial correlation among ECSs due to coded caching and the temporal correlation due to the free cache space causality constraint, yet the placement decision of a content file has to be made without foreseeing the forthcoming content files in the future.

Instead of solving the original **SCCP** problem, we will first investigate a relaxed problem (**R-SCCP**) below:

$$\textbf{R-SCCP} \quad \min_{\boldsymbol{x}^k, \boldsymbol{y}^k, \forall k} \lim_{K\to\infty} \frac{1}{K} \sum_k d^k(\boldsymbol{x}^k) \tag{6a}$$

$$\text{s.t.} \quad x_n^k \leq s^k, y_n^k \leq \tilde{g}_n^k, \qquad \forall n, \forall k \tag{6b}$$

$$x_n^k \leq c_n^k, \qquad \forall n, \forall k \tag{6c}$$

$$\lim_{K\to\infty} \frac{1}{K} \mathbb{E}\left[\sum_k (x_n^k - y_n^k)\right] = 0, \quad \forall n \tag{6d}$$

R-SCCP is a relaxed problem of **SCCP** since the constraint (5b) is relaxed due to $y_n^k \leq \tilde{g}_n^k$ and the causality constraint (5d) is relaxed to (6d). Denote $d^\dagger$ as the infimum average downloading time achievable by any content placement policy that meets the required constraints in **R-SCCP**. Therefore, we must have $d^\dagger \leq d^*$. Because $\tilde{g}_n^k$ may be larger than $C_n - c_n^k$, the cache capacity constraint may be violated. Therefore, we assume $C_n = \infty, \forall n$ for now but we will show later that our algorithm only requires a finite cache capacity while meeting all constraints in **SCCP**. As a result, our algorithm will also be a feasible content placement policy for the original problem. Similar techniques are also adopted in [25]. We define *perturbed* free cache space for each ECS n as $\tilde{c}_n^k = c_n^k - \theta_n$ where θ_n is a perturbation term for each ECS. We choose θ_n as follows:

$$\theta_n \triangleq V p_{\max} \sum_{m \in \mathcal{M}_n} \left(\frac{1}{r_{m,0}} - \frac{1}{r_{m,n}}\right) + s_{\max} \tag{7}$$

where V is a control parameter, and the exact rationale behind this choice will be explained later. The main purpose of the perturbation parameter is to stabilize the occupied caching space and meanwhile minimizing the average downloading time. The required caching capacity can be determined based on the perturbation. The quadratic Lyapunov function associated with the perturbed fr ee cache space is defined as:

$$\Psi^k = \frac{1}{2} \sum_n \left(c_n^k - \theta_n\right)^2. \tag{8}$$

We now examine the Lyapunov drift Δ^k which represents the expected change in the Lyapunov function:

$$\Delta^k \triangleq \mathbb{E}[\Psi^{k+1} - \Psi^k | \boldsymbol{c}^k] \tag{9}$$

where the expectation is with respect to the random process es associated with the system, given the free cache space $\boldsymbol{c}^k$. Moreover, due to the free cache dynamics (5d), we also have $c_n^{k+1} - \theta_n = c_n^k - \theta_n - x_n^k + y_n^k$. Squaring both sides of the above equation, we obtain $(c_n^{k+1} - \theta_n)^2 = (c_n^k - \theta_n)^2 + (x_n^k - y_n^k)^2 - 2(c_n^k - \theta_n)(x_n^k - y_n^k)$. The term $(x_n^k - y_n^k)^2 \leq s_{\max}^2 + g_{\max}^2$ is upper-bounded by a constant. Using this bound and rearranging the above equation, we have $(c^{k+1} - \theta)^2 - (c^k - \theta)^2 \leq s_{\max}^2 + g_{\max}^2 - 2(c^k - \theta)(x^k - y^k)$. Using this inequality and the definition of Δ^k, we have

$$\Delta^k \leq D - \mathbb{E}\left[\sum_n \tilde{c}_n^k \left(x_n^k - y_n^k\right) \mid \boldsymbol{c}^k\right] \tag{10}$$

where $D \triangleq N(s_{\max}^2 + g_{\max}^2)/2$ is a constant. We then add the term $V\mathbb{E}\left[d^k(\boldsymbol{x}^k) \mid \boldsymbol{c}^k\right]$ to both sides and denote

$$\Delta_V^k = \Delta^k + V\mathbb{E}\left[d^k(\boldsymbol{x}^k) \mid \boldsymbol{c}^k\right], \tag{11}$$

which is known as the *drift-plus-penalty*. We further have

$$\Delta_V^k \le D + \mathbb{E}\left[Vd^k(\boldsymbol{x}^k) - \sum_n \tilde{c}_n^k\left(x_n^k - y_n^k\right) \mid \boldsymbol{c}^k\right] \tag{12}$$

Following the theory of Lyapunov optimization (drift-plus-penalty method) [8], the control actions are chosen for each content k to minimize the bound on the *drift-plus-penalty*. Therefore, for each content k, we solve the problem below:

$$\min_{\boldsymbol{x}^k, \boldsymbol{y}^k} \quad Vd^k(\boldsymbol{x}^k) - \sum_n \tilde{c}_n^k\left(x_n^k - y_n^k\right) \tag{13a}$$

$$\text{s.t.} \quad x_n^k \le s^k, \qquad y_n^k \le \tilde{g}_n^k, \forall n \tag{13b}$$

$$x_n^k \le c_n^k, \forall n, \quad \forall k \tag{13c}$$

where V is a positive control parameter to adjust the trade-off between the downloading time and the cache capacity requirement. This problem can be converted to a linear program taking into account the optimal downloading strategies of individual users in (4a) as follows

$$\min_{\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{a}^k} -V\sum_m p_m^k\left(\sum_{n\in\mathcal{B}_m}\left(\frac{1}{r_{m,0}} - \frac{1}{r_{m,n}}\right)a_{m,n}^k\right) - \sum_n \tilde{c}_n^k\left(x_n^k - y_n^k\right) \tag{14a}$$

$$\text{s.t.} \quad x_n^k \le s^k, y_n^k \le \tilde{g}_n^k, \qquad \forall n \tag{14b}$$

$$a_{m,n}^k - x_n^k \le 0, \qquad \forall m, n \tag{14c}$$

$$\sum_{n\in\mathcal{B}_m} a_{m,n}^k \le s^k, \qquad \forall m \tag{14d}$$

The objective function can be further simplified to

$$-\sum_n\left(V\sum_{m\in\mathcal{M}_n}\tilde{p}_{m,n}^k a_{m,n}^k + \tilde{c}_n^k x_n^k\right) + \sum_n \tilde{c}_n^k y_n^k \tag{15}$$

where we define $\tilde{p}_{m,n}^k \triangleq p_m^k\left(1/r_{m,0} - 1/r_{m,n}\right)$. Clearly, the cache pop actions $\boldsymbol{y}^k$ can be decoupled from the cache push actions $\boldsymbol{x}^k$ and the downloading actions $\boldsymbol{a}^k$. Next, we discuss the solutions for $\boldsymbol{y}^k$ and $\boldsymbol{x}^k$ separately.

Solution of the Pop actions $\boldsymbol{y}^k$. The optimal pop actions can be obtained by solving

$$\min_{\boldsymbol{y}^k} \sum_n \tilde{c}_n^k y_n^k \qquad \text{s.t. } y_n^k \le \tilde{g}_n^k, \forall n \tag{16}$$

Depending on the signs of $\tilde{c}_n^k$, the solution is

$$y_n^{k,*} = \begin{cases} \tilde{g}_n^k, & \text{if } c_n^k < \theta_n \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

That is, ECS n removes all expired packets from the cache if the free cache space c_n^k is smaller than a threshold θ_n. Otherwise, it simply leaves these packets in the cache temporally. Again, we note that in actual implementation, all expired packets can be removed from the cache.

Algorithm 1: SCCP

Input: Current free cache space $(c_1^k, \ldots, c_N^k)$, content size s^k, transmission rates $r_{m,n}, \forall m, n$;

1: **for** each new content file k **do**
2: Predict content popularity $p_{m,n}^k, \forall m, n$;
3: Each ECS n pops $y_n^k = \tilde{g}^k$ packets if $c_n^k < \theta_n$ and $y_n^k = 0$ otherwise;
4: Obtain the push decision $x_n^k, \forall n$ and downloading strategies $a_{m,n}^k, \forall m, n$ by solving the linear program (18);
5: Update $c_n^{k+1} = c_n^k - x_n^k + y_n^k, \forall n$;
6: **end for**

Solution of the Push actions $\boldsymbol{x}^k$. The optimal push actions can be obtained by solving the following linear program

$$\textbf{LP} \quad \max_{\boldsymbol{x}^k, \boldsymbol{a}^k} \quad \sum_n\left(V\sum_{m\in\mathcal{M}_n}\tilde{p}_{m,n}^k a_{m,n}^k + \tilde{c}_n^k x_n^k\right) \tag{18a}$$

$$\text{s.t.} \quad x_n^k - s^k \le 0, \qquad \forall n \tag{18b}$$

$$a_{m,n}^k - x_n^k \le 0, \qquad \forall m, n \tag{18c}$$

$$\sum_n a_{m,n}^k - s^k \le 0, \qquad \forall m \tag{18d}$$

In particular, because $V\sum_{m\in\mathcal{M}_n}\tilde{p}_{m,n}^k a_{m,n}^k + \tilde{c}_n^k x_n^k \le \left(V\sum_{m\in\mathcal{M}_n}\tilde{p}_{m,n}^k + \tilde{c}_n^k\right)x_n^k$, we must have $x_n^k = 0$ if $V\sum_{m\in\mathcal{M}_n}\tilde{p}_{m,n}^k + \tilde{c}_n^k < 0$. In other words, ECS n does not push anything into the cache if the current free cache space c^k is smaller than a threshold $\theta_n^{'k}$ defined as $\theta_n^{'k} \triangleq \theta - V\sum_{m\in\mathcal{M}_n} p_m^k\left(1/r_{m,0} - 1/r_{m,n}\right)$. Note that $\theta_n^{'k}$ is not a constant but is changing depending on the popularity of content k among the users served by ECS n. If the content is sufficiently unpopular (in the sense of weighted average popularity among all user regions where the weight for region m is $\frac{1}{r_{m,0}} - \frac{1}{r_{m,n}}$), then ECS n will not cache the content at all to save cache space for future (possibly more popular) content. However, even if the content is sufficiently popular, ECSs may not want to push the entire content in the cache. By collaborating with other ECSs, each ECS only needs to cache a portion of the content, and the the exact number of packets to be cached is determined by **LP**. We summarize the proposed SCCP algorithm in Algorithm 1.

A. Performance Analysis

First, we show that by employing the proposed algorithm, g_n^k in fact equals $\tilde{g}_n^k$ for all k by choosing an appropriate V provided that the cache capacity is moderately large.

Lemma 1: For any $C_n > Vp_{\max}\sum_{m\in\mathcal{M}_n}(1/r_{m,0} - 1/r_{m,n}) + s_{\max} + 2g_{\max}$, we have $g_n^k = \tilde{g}_n^k, \forall k, n$ by choosing

$$0 \le V \le \min_n \frac{C - s_{\max} - 2g_{\max}}{p_{\max}\sum_{m\in\mathcal{M}_n}(1/r_{m,0} - 1/r_{m,n})} \tag{19}$$

$$\theta_n = Vp_{\max}\sum_{m\in\mathcal{M}_n}\left(\frac{1}{r_{m,0}} - \frac{1}{r_{m,n}}\right) + s_{\max}, \quad \forall n \tag{20}$$

Proof: See in online Appendix A [35].

Lemma 1 is a crucial result because it shows that our algorithm not only leads to a feasible content placement policy for the relaxed **R-SCCP** problem but also a feasible policy for the original **SCCP** problem. Based on this result, Theorem 1 below proves a performance guarantee for our algorithm.

Theorem 1: If the cache capacity satisfies

$$C_n \geq V p_{\max} \sum_{m \in \mathcal{M}_n} \left(\frac{1}{r_{m,0}} - \frac{1}{r_{m,n}} \right) + s_{\max} + 2g_{\max} \quad (21)$$

then **SCCP** yields a feasible content placement policy. Moreover, the achievable average downloading time satisfies

$$\lim_{K \to \infty} \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}[d^k] \leq d^* + \frac{\tilde{D}}{V} \quad (22)$$

where d^* is the optimal average downloading time that can be achieved by solving the original **SCCP** problem and $\tilde{D}$ is a constant less than D.

Proof: See in online Appendix B [35].

Theorem 1 provides a strong performance guarantee for our proposed algorithm. It proves that the achievable average downloading time is within a bounded deviation of the optimal performance achieved by a policy that possibly knows the information on future content. Moreover, by tuning the control parameter V, SCCP can easily make a $[O(1/V), O(V)]$ tradeoff between the content downloading time and the required cache space, i.e. average downloading time is inversely proportional to the control parameter V and the required cache capacity of ECS is proportional to the control parameter V. The result can be intuitively understood: a stronger performance guarantee (i.e. a tighter bound) can be obtained if the cache capacity is larger. In particular, when we have an unlimited cache space, SCCP achieves the optimal performance and the policy becomes simply to cache the entire content file k on every ECS, which is intuitively optimal.

V. Spatial Coded Packets Reuse

The SCCP algorithm developed in the previous section determines the number of coded packets that should be cached in the ECSs for each content k. For a user to successfully decode to obtain the original content, the coded packets downloaded from possibly multiple ECSs must be *distinct*. To ensure that the user can download distinct coded packets even from multiple ECSs, careful MDS code design is essential. In this section, we describe an MDS code design that fulfills this purpose while minimizing the coding overhead.

The MDS code is described by a tuple (s, Q) where s is the number of original packets and Q is the number of coded packets. This code allows the original s packets to be recovered using any s distinct coded packets from the Q coded packets, and Q/s represents the coding overhead. A straightforward MDS code design is to make $Q = \sum_{n=1}^{N} x_n + s$ distinct coded packets. In this way, each ECS n has x_n coded packets and the MBS has s coded packets, which are all distinct from each other. However, this straightforward design is inefficient: when the number of ECSs N is large, the overhead Q/s can also be large. To address this issue, we propose "spatial coded packets reuse", similar to "spatial spectrum reuse" in wireless communications, and provide the optimal MDS code design that minimizes Q for given s and $x_1, \ldots, x_N$. The key idea is that ECSs apart from each other far enough do not have common service regions and hence, the same coded packets can be cached on these ECSs, thereby reducing the total number of required coded packets. Nevertheless, the spatially complex network structure still demands a careful design of the coded packets allocation among the ECSs.

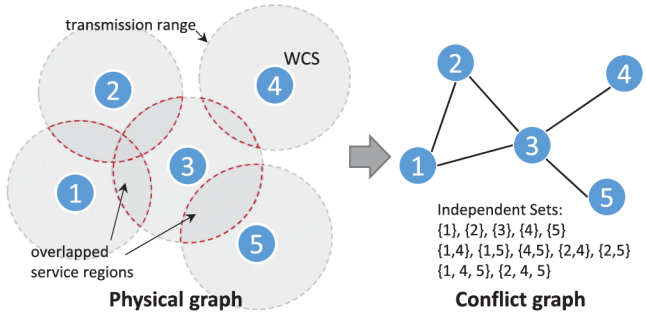


Fig. 4. Conflict graph of collaborative content placement

A. Weighted Conflict Graph for Collaborative Caching

Depending on the locations of the ECSs and their common service regions, we construct a weighted conflict graph $G = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$ for ECSs and the MBS for each content k, where the elements are defined as follows:

- **Vertices**: each caching entity (ECS or MBS) corresponds to a vertex $n \in \mathcal{V}$
- **Edges**: for every pair of vertices, add an edge between them if and only if there exist common service regions that can access both of them. Clearly, there is an edge between MBS and every ECS.
- **Weights**: for each vertex, assign a weight $w(n)$ as the solution of the content placement decision for this content derived in SCCP. In particular, $w(n) = x_n$ if $n \in \mathcal{N}$ is an ECS and $w(n) = s$ if $n = 0$ is the MBS.

Fig. 4 illustrates an example of the weighted conflict graph of collaborative content placement. The weighted conflict graph is different for different content because of the weights. However, the connection relationship remains the same for all content files. The following concepts in graph theory are important for our MDS code design.

Definition 1: An independent set (IS) is a set of vertices in which no pair is connected by an edge.

Let $\mathcal{I}(G)$ denote the set of all ISs of graph G, and $\mathcal{I}(G, n)$ denote the set of ISs of G that contain the vertex n. A fractional coloring of G is a function $f : \mathcal{I}(G) \to \mathbb{R}_+$ that assigns each independent set a non-negative real number such that for any vertex n of G, the sum of real numbers assigned to it is no less than its weight, namely $\sum_{I \in \mathcal{I}(G,n)} f(I) \geq w(n), \forall n$. Apparently, there are many feasible fractional coloring functions. The minimum possible sum $\sum_{I \in \mathcal{I}(G)} f(I)$ over all ISs of a fractional coloring is called the fractional chromatic number $\chi_f(G)$,

Algorithm 2: Spatial Coded Packets Reuse

1: Construct the weighted conflict graph $G = \langle \mathcal{V}, \mathcal{E}, \mathcal{W} \rangle$
2: Determine the set of ISs $\mathcal{I}(G)$ and $\mathcal{I}(G, n), \forall n$
3: **for** each content k **do**
4: Determine $\boldsymbol{x}^k$ by running SCCP
5: Determine f^* and $\chi_f(G)$ by solving (23a)
6: Construct MDS code $(s^k, \chi_f(G))$
7: Distribute coded packets among ISs according to f^*
8: Each ECS/MBS n picks x_n^k coded packets from its associated ISs to push into its cache
9: **end for**

which can be obtained by solving the following minimization problem:

$$\min_{f} \quad \sum_{I \in \mathcal{I}(G)} f(I) \tag{23a}$$

$$\text{s.t.} \quad \sum_{I \in \mathcal{I}(G,n)} f(I) \geq w(n), \quad \forall n \in \mathcal{V} \tag{23b}$$

$$f(I) \geq 0, \qquad \forall I \in \mathcal{I}(G) \tag{23c}$$

B. Optimal MDS Coding via Spatial Coded Packets Reuse

Now we are ready to present our MDS code design via spatial coded packets reuse. For each content, we compute the fractional coloring function f^* that achieves the fractional chromatic number $\chi_f(G)$ by solving (23a). Then we construct a MDS code (s, Q) such that $Q = \chi_f(G)$ following the standard MDS construction process. Next, we distribute the $\chi_f(G)$ coded packets among the ISs without overlapping so that each IS $I \in \mathcal{I}(G)$ gets $f^*(I)$ coded packets. Then each ECS n picks any $w(n)$ coded packets out of the total number of $\sum_{I \in \mathcal{I}(G,n)} f^*(I)$ coded packets that it can use to push into its cache. The whole process is summarized in Algorithm 2.

We discuss a couple of implementation issues regarding the above algorithm as follows. First, Algorithm 2 is run for each content arrival, which involves solving the **SCCP** problem and (23a). Both problems are linear programs so easy to solve. In order to solve (23a), the set of ISs needs to be determined first, which can be complex if the number of ECSs are large. However, computing the ISs is a one-time task that can be carried out offline before the system starts given the coverage relationships of the ECSs and the user regions. Moreover, there exist various graph coloring approaches that can be used to obtain an approximate solution of $\chi_f(G)$ with low complexity. Second, since the content is cached into the ECSs, broadcasting can be utilized to reduce duplicated transmissions. Specifically, the MBS broadcasts a stream of $\chi_f(G)$ coded packets to all ECSs, and only when the coded packets are for the ISs that ECS n belongs to does this ECS push these packets into its cache. This can be done by simply adding a few bits to the broadcasted stream, which indicate the IS that each coded packet belongs to.

We now prove the optimality of the proposed MDS coding scheme in terms of minimizing the coding overhead Q/s.

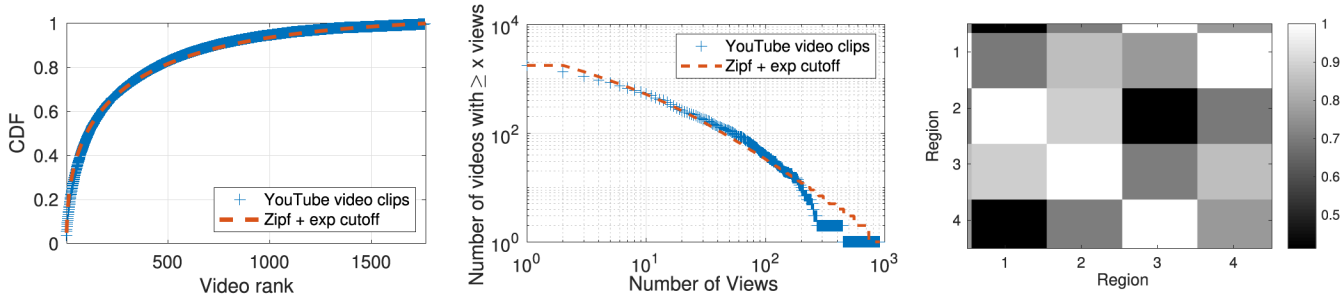
Theorem 2: The proposed MDS coding scheme minimizes the coding overhead Q/s.

Proof: See in online Appendix C [35].

VI. Simulation

We carry out simulations on real-world YouTube video request traces [36] to evaluate the performance of the proposed algorithm (SCCP). We simulate a 500 m $\times$ 500 m area served by 5 ECSs whose locations are randomly chosen. We split the video requests into 4 groups according to their IP addresses. To be specific, the requests are grouped by Network ID (i.e., the first 16 bits of IP address). For example, given three users' IP addresses 254.212.25.141, 254.212.31.36, and 63.22.67.111, the first and second users belong to the same group since they have the same Network ID 254.212. The rationale behind this grouping scheme is that users with the same Network ID belong to the same LAN and therefore are geographically closer to each other (e.g. employees of the same company). Therefore, each group can be abstracted as a region. We further deploy an MBS whose location is randomly generated and is at least 1Km away from the ECSs. The MBS and ECSs work at a fixed transmission power and the downlink channel condition $h_{m,n}$ is calculated by the path-loss model with log-normal shadowing: $P_L(\text{dist})[\text{dB}] = \bar{P}_L(\text{dist}_0) + 10\gamma \log(\text{dist}/\text{dist}_0) + X_\delta$, where *dist* is the average distance between a region and an ECS and $\text{dist}_0 = 1$km, $X_\delta \sim N(0, \delta^2)$ is the random shadowing effects, $\gamma = 2$ is the path-loss exponent, $\bar{P}_L(\text{dist}_0) = 28$ dB. Other parameters for wireless communication model in (1) are: noise power $N_0 = -174$ dBm/Hz, transmission power of ECS n is $P_n = 20$ dBm, and channel bandwidth $W_n = 20$ MHz. We employ the policy in [37] to determine the occupancy time of content files. The occupancy time L_k for content file k is calculated as $L^k = \exp(250 \cdot p^k - 1)/p^k$, where popular content files tend to have a longer occupancy time. However, SCCP is also compatible with other strategies that determine the occupancy time of the content in different ways. We compare the performance of SCCP with the following benchmarks:

- First In First Out (FIFO) [38]: Each ECS pops out the earliest arrived content and pushes as many coded packets as possible into the cache upon arrival of a new content file.
- Least Frequently Used (LFU) [39]: Each ECS has an ordered list to track the number of accesses of cached content files. LFU pushes as many coded packets as possible into the cache upon arrival of new content files. The least frequently used one is replaced when the cache is full.
- Life-time Based Caching (LBC) [37]: Each content file is associated with an expected occupancy time $L^k = (\exp(250 \cdot p^k) - 1)/p^k$ determined by its popularity p_k. Upon the arrival of a content file, LBC pushes as many packets as possible into each ECS and ensures their occupancy time. LBC pops out packets when they expire.
- Myopic SCCP (SCCP-M): SCCP-M is a myopic version of SCCP where ECSs collaboratively cache coded packets to minimize the content downloading time but does not concern the forthcoming content. Specifically, upon the arrival of content file k, ECSs solve a myopic optimization problem $\min_{\boldsymbol{x}^k} d^k(\boldsymbol{x}^k) + \boldsymbol{w}^\top \boldsymbol{x}^k$ subject to current cache space constraints. The term $\boldsymbol{w}^\top \boldsymbol{x}^k$ is added to avoid excessive pushing, without which the problem degenerates to LBC. The weight is set to $\boldsymbol{w} = [3, 3, \ldots, 3]^\top$ which is empirically optimal.



(a) CDF of content popularity (b) Distribution of user requests (c) Popularity correlation across regions

Fig. 5. Statistics of YouTube data.

- Non-collaborative SCCP (SCCP-NC): SCCP-NC is a non-cooperative version of SCCP where each ECS decides online content placement decisions independently using the Lyapunov technique. Moreover, the users can only request content from one ECS or the MBS.
- Non-caching: The users directly request content files via MBS without exploiting ECSs.

In our simulation, FIFO, LFU, LBC are semi-collaborative caching schemes in the sense that users are allowed to download from multiple ECSs fbut the content placement decisions of ECSs are not jointly optimized.

A. YouTube Data

We use the YouTube data from the study conducted at the University of Massachusetts' Amherst campus [36]. The study records YouTube requests arising from the campus network for several days. We use the data recorded on 09/15/07. The request traces contain 928 unique users (source IP address); every user request has a start timestamp, duration (in second), requested content (content server IP), and the size of the requested content. The data contains 1,766 video files across the operational timeline which is equally divided into 20,000 time slots. Each content placement decision cycle begins with a request for a new video. The distribution of the content files is presented in Fig. 5. As can be seen from Fig. 5(a) and Fig. 5(b), the popularity distribution of the adopted YouTube video data follows a Zipf's distribution with an exponential cutoff. Based on the content requests of users in four regions, we analyze the content popularity for each region and depict the correlation of content popularity across four regions in Fig. 5(c). For example, the color block at the grid (1,3) denotes the correlation of content popularity vectors $(p_1^1, p_1^2, \ldots, p_1^K)$ and $(p_3^1, p_3^2, \ldots, p_3^K)$. We see that the correlation value at grid (1,3) is very small, which means that the content popularities of region 1 and region 3 are very different. It can be observed in Fig. 5(c) that these four regions have noticeable differences in content popularity as has been considered in our model.

B. Performance Comparison

Fig. 6 shows the average downloading time (the objective of the SCCP problem in (5)) of SCCP and 6 benchmarks on the YouTube data upon the arrival of each content file k (i.e., $\frac{1}{k}\sum_{i=1}^{k} d^i(\boldsymbol{x}^i)$). The cache capacity of each ECS is set

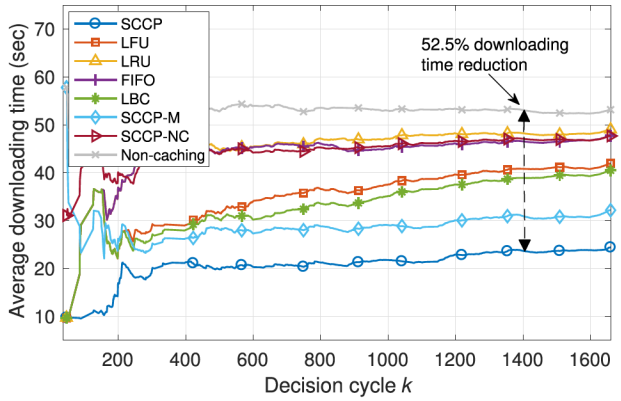


Fig. 6. Performance comparison.

to 600 packets. It can be observed from Fig. 6 that SCCP significantly outperforms other benchmark schemes, providing a 52.5% downloading time reduction compared to the *Non-caching* scheme. In general, traditional caching schemes, i.e. LFU, LRU, and FIFO, incur large downloading time, since these schemes simply cache content files as they arrive without analyzing the popularity of content. As a result, the cache space at ECSs is more likely to be occupied by unpopular content due to occupancy time in TTL setting. By comparing SCCP and SCCP-NC, we see that enabling the collaboration among ECSs dramatically reduces the content downloading time. Comparing SCCP and SCCP-M, we see that the content downloading time can be further reduced by proactively considering forthcoming content while making content placement decision for the current content file.

Besides the content downloading time, we also report an MBS offloading performance of SCCP and other benchmarks in Fig. 7. Specifically, the MBS offloading performance shows the fraction of content packets downloaded from ECSs and the fraction of content packets downloaded from the MBS. Fig. 7 shows that SCCP has the highest fraction of packets downloaded from ECSs.

C. Impact of Control Parameter V

We are also interested in whether the theoretical performance guarantee of SCCP still holds in a realistic scenario. We vary

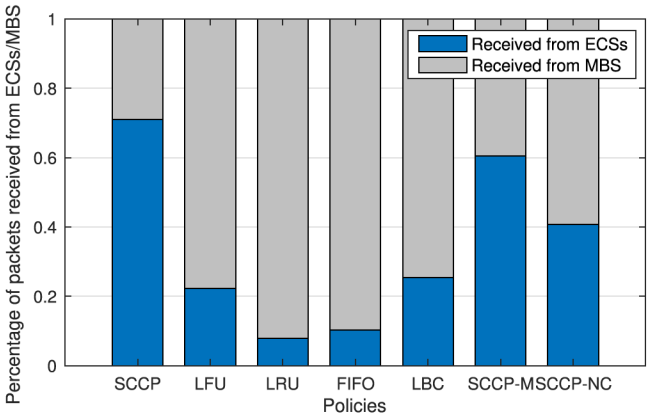


Fig. 7. Content packet MBS offloading.

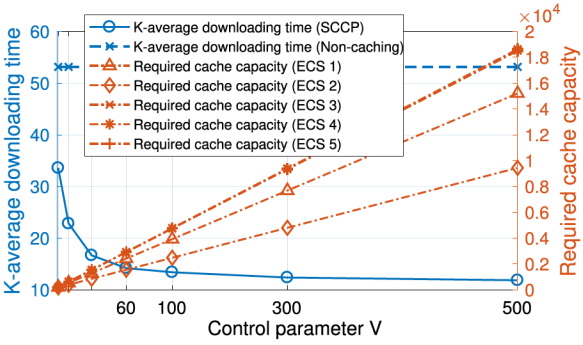


Fig. 8. Impact of control parameter$V$.

the control parameter V instead of setting a fix cache capacity C. Fig. 8 shows the K-average downloading time (i.e., $\frac{1}{K}\sum_{k=1}^{K} d^k(\boldsymbol{x}^k)$, where K is the total number of content files in YouTube data) and required cache capacity under different values of control parameter V. The result shows clearly that the performance of SCCP follows a $[O(1/V), O(V)]$ tradeoff between the content downloading time and the required cache capacity as characterized in Theorem 1. With a larger V, SCCP emphasizes more on minimizing the downloading time and a large cache capacity is required to implement the algorithm.

D. Impact of Cache Capacity

Fig. 9 shows the K-average downloading time (i.e., $\frac{1}{K}\sum_{k=1}^{K} d^k(\boldsymbol{x}^k)$, where K is the total number of content files in YouTube data) of SCCP and other benchmarks with different cache capacities. Fig. 9 shows a general trend that the downloading time decreases as the cache capacity increases for all content placement schemes. The reason is intuitive: with larger caches at ECS, more coded packets can be downloaded at a higher transmission rate. In addition, it is worth emphasizing that SCCP is more effective when the cache capacity is small: when the cache capacity is around 200, the downloading time of SCCP is much lower than other benchmarks; however, when the capacity is increased to 1600, many benchmarks can achieve similar downloading time as SCCP does.

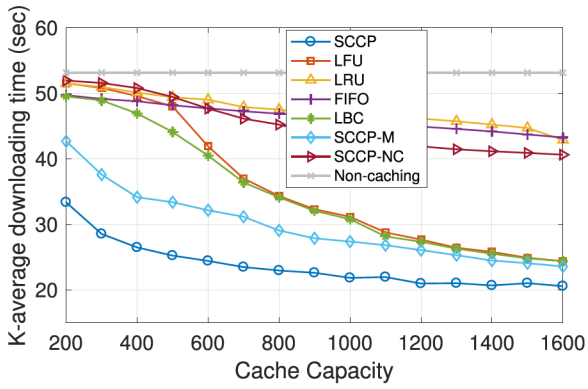


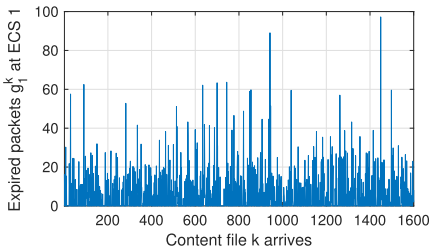
Fig. 9. Impact of cache capacity on downloading time.

E. Analysis of Expiration Process

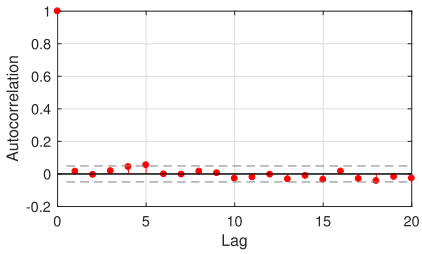
The content expiration process g_n^k is assumed to be i.i.d. previously for ease of performance analysis of SCCP. Here, we analyze the content expiration process when running the proposed algorithm on the YouTube data to see if this i.i.d. assumption holds true. Fig. 10(a) depicts the number of expired packets at ECS 1 upon the arrival of content file k. To check whether the expiration process in Fig. 10(a) is i.i.d., we should ask two questions: 1) are the observations independent? 2) do they all have the same distribution? As for the first question, the autocorrelation function can be used to investigate the independence of the expiration process. The independence is ascertained by computing autocorrelations for the expiration process at varying time lags. If independent, such autocorrelations should be near zero for any and all time-lag separations. Fig. 10(b) gives a lag plot of autocorrelation function of expiration process. We can see that the value of autocorrelation function is effectively 0 for all lag separations, which means the observations are independent. For the second question, we do a stationary test for the expiration process. If the expiration process is stationary (i.e., the distribution of the expiration process does not change over time), it means that the observations are from an identical distribution. We utilize a moving-window to show the changes in the statistic parameters, namely mean and standard deviation, of the expiration process over time. The length of the moving window is set to 400 and the step is 1. The result is given in Fig. 10(c) and it shows that the mean and standard deviation stay almost the same over time, which indicates that the expiration process is stationary. With the above analysis, we can conclude that the expiration process of SCCP is i.i.d.

F. Impact of Popularity Prediction Accuracy

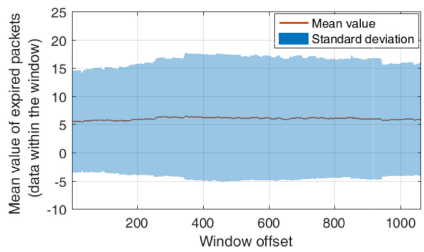
One assumption of our algorithm is that the content popularity can be accurately predicted upon their arrival. However, this assumption may not be true in real-world applications. Therefore, we run SCCP with popularity prediction errors to see the impact of prediction accuracy. Specifically, we add random errors (chosen from a normal distribution) on the ground-truth content popularity to simulate the inaccurate prediction. Fig. 11 shows the performances of SCCP when running with different prediction errors. We can see clearly that SCCP provides better



(a) Content expiration process at ECS 1



(b) Autocorrelation of expiration process



(c) Stationary test for expiration process

Fig. 10. Analysis of content expiration process.

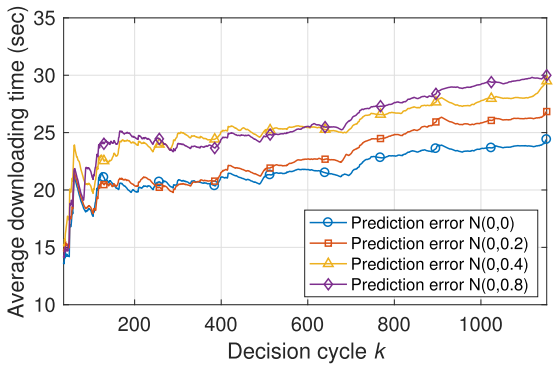


Fig. 11. Impact of prediction accuracy.

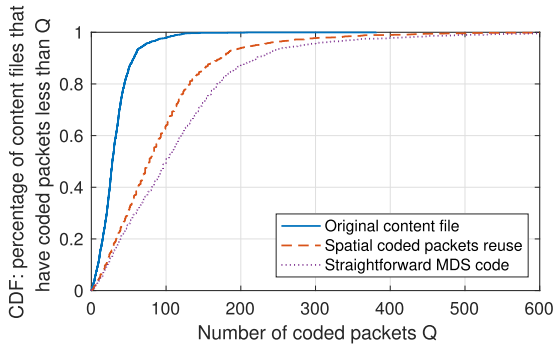


Fig. 12. CDF of content files on the number of coded packets.

performance (i.e., lower content downloading time) if the popularity prediction is more accurate. By comparing Fig. 11 and Fig. 6, we see that SCCP still has lower downloading time compared to other benchmarks even when the popularity prediction is inaccurate.

G. Spatial Coded Packets Reuse

Fig. 12 compares the coding overhead of the *spatial coded packets reuse* and the straightforward MDS. It depicts a CDF of content files on the number of created coded packets. We see that with the proposed *spatial coded packets reuse*, the system requires much fewer coded packets compared to the straightforward MDS coding, thereby reducing the coding overhead and complexity.

VII. Conclusions

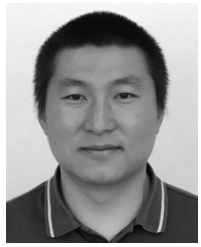
In this paper, we investigated collaborative caching in ECS networks, explicitly considering the stochastic nature of content arrivals and the content occupancy time requirement. An online algorithm is developed by leveraging the *Lyapunov optimization* with perturbation. The key idea of our online algorithm is that the ECSs make caching decisions collaboratively based on the current free cache spaces in the hope of saving proper cache spaces for the potentially more popular future content, thereby improving the caching performance in the long run. The proposed algorithm provides provable performance guarantee that achieves within a bounded deviation from the optimal caching performance. We evaluated our algorithm on a real-world YouTube video request trace and our simulation results show that the proposed collaborative ECS caching system reduces the average downloading time by more than 50% compared to the non-caching scenario.

References

[1] A. Iacono and C. Rose, "Infostations: New perspectives on wireless data networks," *Next Gener. Wireless Netw.*, vol. 598, pp. 3–63, 2000.
[2] P. Blasco and D. Gunduz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *Proc. IEEE Int. Symp. Inf. Theory*, 2014, pp. 51–55.
[3] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1029–1040, Aug. 2015.
[4] N. Bhushan *et al.*, "Network densification: The dominant theme for wireless evolution into 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 82–89, Feb. 2014.
[5] P. K. Agyapong and M. Sirbu, "Economic incentives in information-centric networking: Implications for protocol design and public policy," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 18–26, Dec. 2012.
[6] R. T. Ma and D. Towsley, "Cashing in on caching: On-demand contract design with linear pricing," in *Proc. 11th ACM Conf. Emerg. Netw. Exp. Technologies*, 2015, pp. 1–6.
[7] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. Tay, "A utility optimization approach to network cache design," in *Proc. Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[8] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[9] 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.
[10] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-driven mobile edge caching placement for adaptive video streaming," *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 965–984, Apr. 2018.
[11] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 3358–3363.
[12] T. Wang, L. Song, and Z. Han, "Dynamic femtocaching for mobile users," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2015, pp. 861–865.

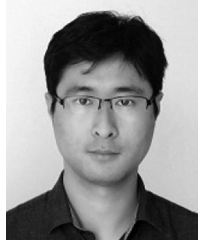
[13] Z. Zhou, M. Dong, K. Ota, and Z. Chang, "Energy-efficient context-aware matching for resource allocation in ultra-dense small cells," *IEEE Access*, vol. 3, pp. 1849–1860, 2015.
[14] S. Zhou, J. Gong, Z. Zhou, W. Chen, and Z. Niu, "Greendelivery: Proactive content caching and push with energy-harvesting-based small cells," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 142–149, Apr. 2015.
[15] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.
[16] X. Wang, X. Li, V. C. Leung, and P. Nasiopoulos, "A framework of cooperative cell caching for the future mobile networks," in *Proc. Hawaii Int. Conf. Syst. Sci.*, 2015, pp. 5404–5413.
[17] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2275–2284, Aug. 2016.
[18] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, Aug. 2016.
[19] J. Tadrous and A. Eryilmaz, "On optimal proactive caching for mobile networks with demand uncertainties," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2715–2727, Oct. 2016.
[20] J. Liu, Q. Yang, and G. Simon, "Joint optimization of content placement and request redirection in mobile-CDN," in *Proc. FIP/IEEE Int. Symp. Integr. Netw. Manage.*, 2017, pp. 169–176.
[21] B. Zhou, Y. Cui, and M. Tao, "Stochastic content-centric multicast scheduling for cache-enabled heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6284–6297, Sep. 2016.
[22] 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.
[23] L. Huang and M. J. Neely, "Utility optimal scheduling in energy-harvesting networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1117–1130, Aug. 2013.
[24] Y. Mao, J. Zhang, and K. B. Letaief, "A Lyapunov optimization approach for green cellular networks with hybrid energy supplies," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2463–2477, Dec. 2015.
[25] S. Lakshminarayana, T. Q. Quek, and H. V. Poor, "Cooperation and storage tradeoffs in power grids with renewable energy resources," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 7, pp. 1386–1397, Jul. 2014.
[26] Z. Chen, J. Lee, T. Q. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3401–3415, May 2017.
[27] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.
[28] V. Bioglio, F. Gabry, and I. Land, "Optimizing MDS codes for caching at the edge," in *Proc. IEEE Global Commun. Conf.*, 2015, pp. 1–6.
[29] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.
[30] F. Kocak, G. Kesidis, T.-M. Pham, and S. Fdida, "The effect of caching on a model of content and access provider revenues in information-centric networks," in *Proc. Int. Conf. Social Comput.*, 2013, pp. 45–50.
[31] M. Wildemeersch, T. Q. Quek, M. Kountouris, A. Rabbachin, and C. H. Slump, "Successive interference cancellation in heterogeneous networks," *IEEE Trans. Commun.*, vol. 62, no. 12, pp. 4440–4453, Dec. 2014.
[32] S. Zhang, N. Zhang, X. Fang, P. Yang, and X. Shen, "Self-sustaining caching stations: Toward cost-effective 5G-enabled vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 202–208, Nov. 2017.
[33] J. Xu, M. van der Schaar, J. Liu, and H. Li, "Forecasting popularity of videos using social media," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 2, pp. 330–343, Mar. 2015.
[34] E. Ozfatura and D. Gündüz, "Mobility and popularity-aware coded small-cell caching," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 288–291, Feb. 2018.
[35] L. Chen, L. Song, J. Chakareski, and J. Xu, Online appendix. 2019. [Online]. Available: https://www.dropbox.com/sh/hmiexb15ioz01u2/AABGvGkwEJ3IHxhB95HpUH3Va?dl=0
[36] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network–measurements, models, and implications," *Comput. Netw.*, vol. 53, no. 4, pp. 501–514, 2009.
[37] H. Qian, W. Muqing, W. Dongyang, and G. Song, "Lifetime-based greedy caching approach for content-centric networking," in *Proc. Int. Conf. Telecommun.*, 2014, pp. 426–430.
[38] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Telecom Paris-Tech*, Paris, France, Tech. Rep. 1, 2011.
[39] D. Lee *et al.*, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.

Lixing Chen received the B.S. and M.S. degrees from the College of Information and Control Engineering, China University of Petroleum, Qingdao, China, in 2013 and 2016, respectively. He is currently working toward the Ph.D. degree at the College of Engineering, University of Miami, Coral Gables, FL, USA. His primary research interests include mobile edge computing, game theory, and machine learning for networks.



Linqi Song (M'17) received the B.S. and M.S. degrees in electronic engineering, Tsinghua University, Beijing, China, and the Ph.D. degree in electrical engineering from University of California, Los Angeles (UCLA), Los Angeles, CA, USA. He is currently an Assistant Professor with the Computer Science Department, City University of Hong Kong, Hong Kong. Prior to that, he was a Postdoctoral Scholar with the Electrical and Computer Engineering Department, UCLA. His research interests include content-type coding, index coding, network coding, algorithms, big data, and machine learning. He was the recipient of the UCLA Fellowship for his graduate studies.

Jacob Chakareski (SM'14) trained as a Ph.D. student at Rice University, Houston, TX, USA, and Stanford University, Stanford, CA, USA. He is currently an Associate Professor with the Ying Wu College of Computing, New Jersey Institute of Technology, Newark, NJ, USA, where he leads the Laboratory for VR/AR Immersive Communication. His research interests span networked virtual and augmented reality, UAV IoT sensing and networking, fast online machine learning, 5G wireless edge computing/caching, ubiquitous immersive communication, and societal applications. He was the recipient of the Adobe Data Science Faculty Research Award in 2017 and 2018, the Swiss NSF Career Award Ambizione in 2009, the AFOSR Faculty Fellowship in 2016 and 2017, and Best/Fast Track Paper Awards at the IEEE International Conference on Communications in 2017 and the IEEE Global Communications Conference in 2016. He is the organizer of the first NSF Visioning Workshop on networked VR/AR communications. He held research appointments with Microsoft, HP Labs, and EPFL, and sits on the advisory board of Frame, Inc. His research was supported by the NSF, AFOSR, Adobe, Tencent Research, NVIDIA, and Microsoft.



Jie Xu (S'09–M'15) received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2008 and 2010, respectively and the Ph.D. degree in electrical engineering from the University of California, Los Angeles, Los Angeles, CA, USA in 2015. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, University of Miami, Coral Gables, FL, USA. His primary research interests include mobile edge computing, machine learning for networks, and network security.